EXPERTGEN: Scalable Sim-to-Real Expert Policy Learning from Imperfect Behavior Priors

Zifan Xu[1,2], Ran Gong[1], Maria Vittoria Minniti[1], Ahmet Salih Gundogdu[1], Eric Rosen[1], Kausik Sivakumar[1], Riedana Yan[1], Zixing Wang[1], Di Deng[1], Peter Stone[2,3], Xiaohan Zhang[*,1], Karl Schmeckpeper[*,1]

[1]Robotics and AI Institute, [2]University of Texas at Austin, [3]Sony AI, [*]Equal Advising

bdaiinstitute.github.io/expertgen

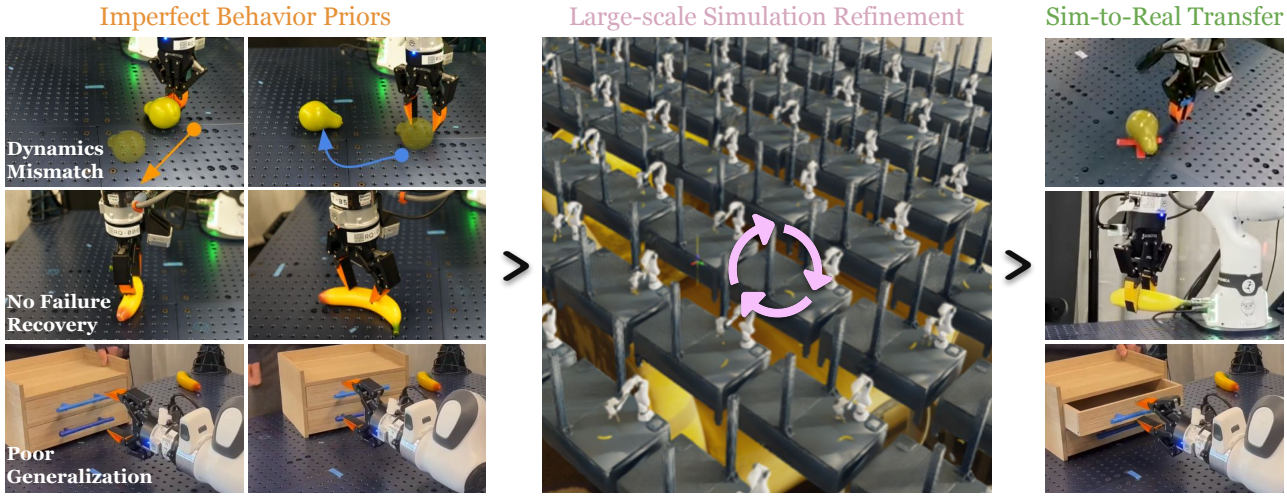

Fig. 1: EXPERTGEN pipeline: (left) generative modeling of imperfect behavior priors; (middle) steering prior model in massively parallel simulation using reinforcement learning; and (right) visual distillation in simulation with DAgger for zero-shot sim-to-real transfer.

***Abstract*—Learning generalizable and robust behavior cloning policies requires large volumes of high-quality robotics data. While human demonstrations (e.g., through teleoperation) serve as the standard source for expert behaviors, acquiring such data at scale in the real world is prohibitively expensive. This paper introduces EXPERTGEN, a framework that automates expert policy learning in simulation to enable scalable sim-to-real transfer. EXPERTGEN first initializes a behavior prior using a diffusion policy trained on imperfect demonstrations, which may be synthesized by large language models or provided by humans. Reinforcement learning is then used to steer this prior toward high task success by optimizing the diffusion model's initial noise while keep original policy frozen. By keeping the pretrained diffusion policy frozen, EXPERTGEN regularizes exploration to remain within safe, human-like behavior manifolds, while also enabling effective learning with only sparse rewards. Empirical evaluations on challenging manipulation benchmarks demonstrate that EXPERTGEN reliably produces high-quality expert policies with no reward engineering. On industrial assembly tasks, EXPERTGEN achieves a 90.5% overall success rate, while on long-horizon manipulation tasks it attains 85% overall success, outperforming all baseline methods. The resulting policies exhibit dexterous control and remain robust across diverse initial configurations and failure states. To validate sim-to-real transfer, the learned state-based expert policies are further distilled into visuomotor policies via DAgger and successfully deployed on real robotic hardware.**

I. INTRODUCTION

The success of deep learning has been driven by access to large-scale, high-quality data [1–3], as evidenced across domains such as natural language processing [4–7], visual understanding [8–10], and multimodal learning [11–13]. Similar trends have emerged in robotics with the rise of vision–language–action (VLA) models [14–18], which aim to unify perception, language understanding, and control within a single framework. Trained on internet-scale image–language corpora, these models exhibit strong semantic generalization, such as instruction following and spatial reasoning capabilities. However, due to the scarcity of high-quality robotics data, translating semantic competence into reliable physical execution remains a major challenge [19]. In contrast to static visual or linguistic data, robotics data must be temporally aligned perception-action sequences that support closed-loop control, long-horizon skill chaining, and recovery from failures. The scarcity of such expert-level robotics data has become a primary bottleneck for real-world execution of VLA models.

Simulation offers a scalable alternative by enabling large-scale data generation through massive GPU parallelism. Prior work has explored simulation-based strategies such as teleop-

eration, which still requires substantial human effort, and predefined skill libraries or scripted policies [20], which produce limited behavioral diversity and are brittle to distribution shift and failure states. In principle, reinforcement learning (RL) provides a promising solution by directly optimizing closed-loop task success and producing expert policies that generalize across scene configurations, exhibit behavioral diversity [21], and demonstrate robustness to perturbations [22–24]. In practice, however, scaling RL-based data generation remains challenging, as training high-performing expert policies typically depends on carefully engineered reward functions. Designing such rewards requires significant domain expertise in both robotics and RL, limiting scalability across tasks and domains.

To address these challenges, this paper focuses on *scalable learning of sim-to-real expert policies* that satisfy four key properties: (1) spatial generalization beyond demonstrated trajectories, (2) robust failure recovery, (3) transferability to real-world systems, and (4) learning under sparse task-level rewards without any reward engineering.

To this end, we introduce EXPERTGEN, which starts from a small set of human- or LLM-generated *imperfect* demonstrations in simulation that may exhibit limited state coverage, incomplete recovery behaviors, or embodiment and dynamics mismatch. These demonstrations are used to pretrain a lightweight state-based diffusion policy, serving as *imperfect behavior priors*, which is then refined in massively parallel simulation (e.g, IsaacLab [25]) via diffusion steering reinforcement learning (DSRL) [26]. DSRL optimizes for task success by steering only the initial noise of the diffusion model, while preserving the generative denoising process to maintain in-distribution, human-like behaviors. EXPERTGEN instantiates this process using FastTD3 [27] enabling efficient learning under sparse, task-level rewards without any reward engineering, and facilitating policy learning across large simulation batches. Finally, the resulting state-based expert policies are leveraged as teachers in a DAgger-style [28] distillation stage, producing visuomotor policies that inherit the expert's robustness, spatial generalization, and failure recovery, while being deployable on real hardware.

The major contributions of the paper are

- Scaling diffusion steering to massively parallel robotics simulation, demonstrating that it preserves the natural motion manifold of diffusion models while significantly boosting success rates under sparse rewards.
- Introducing EXPERTGEN, an end-to-end framework that transforms a handful of imperfect demonstrations into sim-to-real-ready expert policies, bypassing the need for manual reward engineering.
- Robust zero-shot sim-to-real transfer of visuomotor policies via large-scale DAgger-based distillation, identifying critical bottlenecks encountered when learning from simulated expert demonstrations.

II. RELATED WORK

This section reviews related work along two complementary directions: offline-to-online reinforcement learning, and synthetic data generation for scalable robot learning.

A. Offline-to-Online Reinforcement Learning

A growing body of work studies how to refine policies learned from offline data using limited online interaction, with the goal of improving generalization and closed-loop performance while preserving strong demonstration priors. Residual reinforcement learning (RL) [29] methods exemplify this paradigm by constraining online learning to an additive correction space. Residual RL has also shown promise in improving precision for dexterous assembly [30] and robustness for whole-body loco-manipulation [31]. However, these approaches typically require careful tuning of residual action bounds [32] and exploration schedules [32, 33], making them non-trivial to scale across domains and task families.

In parallel, offline-to-online RL methods based on pretrained value functions [34] have gained traction due to their strong sample efficiency, particularly in real-world settings where online interaction is expensive [21]. Separately, diffusion policies (DP) [35] have emerged as a powerful framework for offline imitation learning, motivating specialized offline-to-online refinement strategies. DPPO [36] fine-tunes diffusion policies by treating the denoising process as a Markov decision process and optimizing task rewards via RL. In contrast, diffusion steering reinforcement learning (DSRL) [26] performs online refinement by learning to steer the initial diffusion noise using a lightweight RL adaptor, without updating the pretrained policy weights. We build on DSRL for offline-to-online refinement, due to its simplicity in manipulating the diffusion noise space and preserving the learned data manifold. We demonstrate that combining this approach with large-scale simulation allows for constraining exploration to human-like behaviors and enables robust sim-to-real transfer.

B. Synthetic Robotics Data Generation

Another complementary line of work addresses the scalability bottleneck of robotics data collection through *synthetic demonstration generation and data augmentation*. For data augmentation, a common strategy is to algorithmically expand a small number of human demonstrations into large, task-consistent datasets by exploiting structure in task geometry, kinematics, and skill composition. Systems such as MimicGen [37] and its extensions—including SkillMimicGen [38], DexMimicGen [39], DemoGen [40], and ReinforceGen [41]—formalize this approach by programmatically perturbing demonstrations or composing reusable skill primitives, enabling scalable data generation for long-horizon, contact-rich, and bimanual dexterous manipulation. However, these augmentation strategies primarily exploit task structure around successful executions and do not guarantee sufficient coverage of failure modes or recovery behaviors.

For synthetic demonstration generation, the ManiSkill frameworks [42, 43] utilize RL for large-scale automatic data collection. However, these systems still requires substantial human effort for manual task design and reward engineering. To address this, more recent work [20, 44–47] employs

scripted policies refined by LLMs to generate high-quality datasets. While efficient, these scripted strategies often lack behavioral diversity; consequently, the resulting policies suffer from limited state-space coverage and lack robust recovery capabilities when faced with out-of-distribution scenarios.

In contrast, EXPERTGEN adopts a multi-stage approach that begins with LLM-generated scripted trajectories or human teleop trajectories to train a state-space diffusion policy. We then employ DSRL to fine-tune this policy for large-scale data collection or visul distillation. By leveraging the diffusion model as a prior, DSRL ensures that the resulting motions remain within the natural motion data manifold specified by the data, significantly improving sim-to-real transfer compared to unconstrained RL. Furthermore, this steering mechanism allows the policy to develop robust recovery behaviors using only sparse rewards, maintaining high collection efficiency across diverse object states with no reward engineering.

III. PRELIMINARIES

This section formalizes the learning problem and introduces the notation used throughout the paper.

A. Constrained Markov Decision Process (CMDP)

To model sim-to-real policy learning under real-world deployment constraints, we formulate the learning problem as a constrained Markov decision process (CMDP). In addition to maximizing task return, the policy is required to select actions that are feasible for execution on physical hardware.

Formally, the CMDP is defined by the tuple

$$\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \mathcal{F}), \tag{1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1]$ is the discount factor, and $\mathcal{F} \subseteq \mathcal{S} \times \mathcal{A}$ denotes the feasible state–action set capturing the constraints during real-world deployment.

At each time step t, the agent selects an action a_t such that $(s_t, a_t) \in \mathcal{F}$. The objective is to learn an expert policy π_E that maximizes the expected discounted return subject to these feasibility constraints:

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right] \\ \text{s.t.} \quad & (s_t, a_t) \in \mathcal{F}, \quad \forall t. \end{aligned} \tag{2}$$

Here $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ denotes a trajectory induced by policy π interacting with the environment dynamics under the feasibility constraints.

In this work, the reward function is defined as a binary task success signal. Let $\mathbb{I}_{\text{succ}}(s_t)$ denote an indicator function that evaluates to 1 if the task is successfully completed at state s_t and 0 otherwise. The reward is given by

$$r(s_t, a_t) = \mathbb{I}_{\text{succ}}(s_t), \tag{3}$$

which provides sparse supervision by assigning nonzero reward only upon task success.

B. Diffusion Policy

This paper considers diffusion policy for all imitation learning training. Specifically, diffusion policy considers demonstrations as samples from a diffusion denoising process. Given a state s_t at time step t, the policy represents the conditional distribution of an action chunk [48] $\mathbf{a}_t = a_{t:t+H}$ of length H via a diffusion model [35]. During training, Gaussian noise is progressively added to expert trajectories according to a forward process

$$q(\mathbf{a}_t^k \mid \mathbf{a}_t^0) = \mathcal{N}\big(\mathbf{a}_t^k; \sqrt{\bar{\alpha}_k}\,\mathbf{a}_t^0,\ (1-\bar{\alpha}_k)I\big)\,, \tag{4}$$

where $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$ defines the noise schedule. The diffusion policy is trained to predict the injected noise using a denoising network ϵ_θ, by minimizing the standard diffusion objective:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{a}_t^0, s_t,\, k,\, \epsilon\sim\mathcal{N}(0,I)}\left[\left\|\epsilon - \epsilon_\theta(\mathbf{a}_t^k, s_t, k)\right\|^2\right]. \tag{5}$$

At inference time, action trajectories are generated by iteratively applying the reverse diffusion process starting from Gaussian noise $\mathbf{a}_t^K \sim \mathcal{N}(0, I)$. To reduce denoising steps, we adopt Denoising Diffusion Implicit Models (DDIM) [49] sampling, which replaces the stochastic reverse process with a deterministic update:

$$\mathbf{a}_t^{k-1} = \sqrt{\bar{\alpha}_{k-1}}\,\hat{\mathbf{a}}_t^0 + \sqrt{1-\bar{\alpha}_{k-1}}\,\epsilon_\theta(\mathbf{a}_t^k, s_t, k), \tag{6}$$

where $\hat{\mathbf{a}}_t^0$ denotes the prediction of the clean action chunk.

IV. EXPERTGEN

This section details the three major components of EXPERTGEN, with an illustration shown in Fig. 2.

A. Generative Behavior Prior Modeling

While the feasible set $\mathcal{F}$ is not assumed to be explicitly known during learning, we assume access to a collection of demonstration trajectories from an *imperfect behavior prior*.

$$\mathcal{D}_P = \{\tau_P^i\}_{i=1}^N, \qquad \tau_P^i \sim \pi_P, \tag{7}$$

where π_P denotes a prior policy that is not optimal for a given task, but is feasible for real-world deployment.

Specifically, the prior policy π_P is assumed to satisfy two properties. First, all state–action pairs generated by π_P are feasible under real-world deployment constraints:

$$(s_t, a_t) \in \mathcal{F}, \quad \forall (s_t, a_t) \in \tau_P^i,\ \forall i. \tag{8}$$

Second, π_P achieves nontrivial task performance, in the sense that it attains positive expected return under the binary success reward:

$$\mathbb{E}_{\tau\sim\pi_P}\left[\sum_{t=0}^{T} \gamma^t \mathbb{I}_{\text{succ}}(s_t)\right] > 0. \tag{9}$$

Despite satisfying these properties, the behavior prior is imperfect. Imperfectness may arise from one or more of the following sources:

- **Incomplete state or geometry coverage**, where demonstrations span only a subset of object poses, contact configurations, or scene layouts;

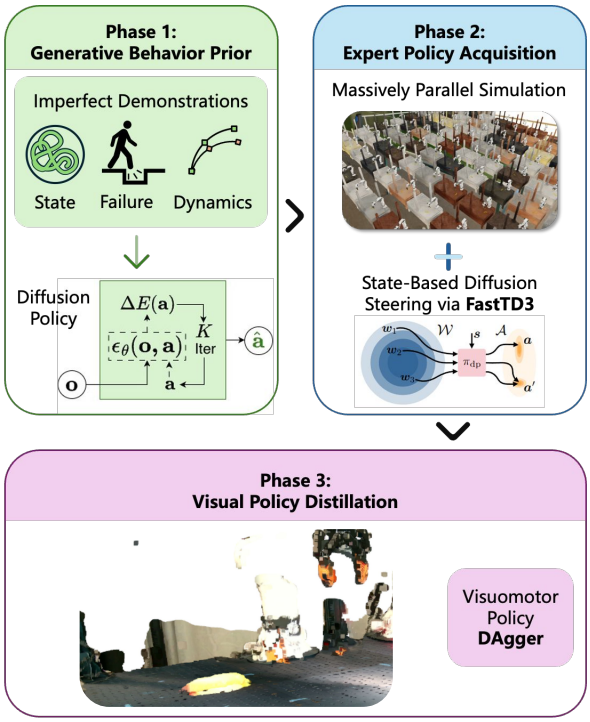

Fig. 2: ExpertGen training pipeline: generative modeling of imperfect behavior priors using a state-based diffusion policy (Phase 1); steering diffusion policy in massive parallel simulation using FastTD3 (phase 2); visual policy distillation using DAgger from expert teachers after steering (phase 3).

- **Limited recovery behaviors**, where demonstrations succeed only under nominal conditions and do not cover off-nominal or failure states; and
- **Dynamics or embodiment mismatch**, where demonstrations are generated under different physical parameters, simplified dynamics, or approximate controllers (e.g., teleoperation, scripted policies, or LLM-generated plans).

As a result, the *imperfect behavior prior* distribution captures feasible and structured motions but does not reliably achieve task success across the full state distribution. EXPERTGEN employs a diffusion policy (Section III-B) trained on imperfect demonstrations to represent the behavior priors.

B. Expert Policy Acquisition

This subsection describes how EXPERTGEN acquires task-specialized expert policies by combining *diffusion steering reinforcement learning* (DSRL) with a massively parallel, off-policy reinforcement learning algorithm based on FastTD3. Starting from the pretrained diffusion policy described in the previous subsection, EXPERTGEN treats the diffusion model as a strong behavior prior and refines it toward task success using only sparse, task-level rewards.

a) Diffusion Steering Reinforcement Learning: EXPERTGEN adopts DSRL to optimize task performance while preserving the structure of behaviors captured by the diffusion prior. Unlike residual or policy fine-tuning approaches, DSRL does *not* modify the diffusion model parameters nor reduce the dimensionality of the action space. Instead, it learns a steering policy that operates in the same space as the diffusion model's chunked action output.

b) Off-Policy RL with FastTD3: While the original DSRL formulation focuses on Soft Actor–Critic (SAC) [50] to support sample-efficient learning on real robot or small-scale simulation benchmarks, EXPERTGEN targets a fundamentally different regime: massively parallel, large-scale simulation training to amortize the cost of long-horizon action chunk rollouts. Accordingly, the steering policy π_ϕ is optimized using FastTD3 [27], a scalable variant of TD3 designed for high-throughput robotic simulation. Compared to standard TD3 [51], FastTD3 introduces several key modifications to TD3 including large-batch training, distributional critics [52], and mixed exploration noise to stabilize TD3 training in a massively parallel environment. As a result, FastTD3 demonstrates superior efficiency compared to PPO in sparse reward massively parallel environment settings. This parallel training paradigm is critical for scaling ExpertGen to diverse scene configurations and long-horizon tasks without sophisticated shaping rewards.

C. Visuomotor Policy Distillation through DAgger

In our setting, expert policies are learned in simulation with access to the full environment state, which may include privileged information unavailable on real hardware. To enable deployment, we assume access to an observation function $f : \mathcal{S} \to \mathcal{O}$ that maps the privileged state to an observation space $\mathcal{O}$, such as RGB images or point clouds. Deployable policies are restricted to operate solely on observations.

Let $\pi_E(a \mid s)$ denote an expert policy trained in simulation using privileged state information. Rolling out π_E in simulation and applying the observation function f induces a dataset of observation–action trajectories

$$\mathcal{D}_E = \{\tau_E^i\}_{i=1}^N, \qquad \tau_E^i = \{(o_t, a_t)\}_{t=0}^T, \quad o_t = f(s_t), \tag{10}$$

where $a_t \sim \pi_E(\cdot \mid s_t)$. The goal of imitation learning is to train a deployable policy $\pi(a \mid o)$ that matches the expert's behavior using only observations.

A standard approach is behavior cloning, which learns π by minimizing the negative log-likelihood of expert actions conditioned on observations:

$$J_{\mathrm{IL}}(\pi) = \mathbb{E}_{(o_t,a_t)\sim\mathcal{D}_E}\left[-\log \pi(a_t \mid o_t)\right]. \tag{11}$$

While behavior cloning provides stable supervised training, it suffers from distribution shift, as the learned policy may encounter states at test time that are not covered by the expert dataset.

Dataset Aggregation (DAgger) [28] addresses this limitation by iteratively collecting data under the learner's induced state distribution. At iteration k, the current policy π_k is executed in simulation to generate states $\{s_t\}$, which are mapped to observations $o_t = f(s_t)$. The expert policy π_E then provides

corrective actions $a_t = \pi_E(s_t)$, and the aggregated dataset is updated as

$$\mathcal{D}_k = \mathcal{D}_{k-1} \cup \{(o_t, a_t)\}. \tag{12}$$

The policy is subsequently updated by minimizing the imitation loss over the aggregated dataset:

$$\pi_{k+1} = \arg\min_{\pi} \mathbb{E}_{(o_t,a_t)\sim\mathcal{D}_k} \left[-\log \pi(a_t \mid o_t) \right]. \tag{13}$$

Through this procedure, imitation learning distills a simulated, state-based expert policy into an observation-based policy that can be deployed in the real world.

Concretely, the expert is executed in simulation to label actions at the student's visited states, allowing the visual policy to iteratively correct compounding errors and learn robust closed-loop behavior. During distillation, extensive visual domain randomization is applied—including variations in textures, lighting, camera poses, and object appearances—to improve generalization and sim-to-real transfer. This combination of DAgger-style online supervision and aggressive visual randomization yields visual policies that faithfully reproduce expert behaviors while remaining robust to visual uncertainty at deployment time.

V. Experiment Setup

A. Benchmarks (Figure 4)

a) AnyTask: AnyTask [20] is an automated framework for task design and large-scale synthetic data generation in robotic manipulation. For benchmarking, we select eight tabletop manipulation tasks spanning a broad range of behaviors, including lifting, pushing, stacking, pick-and-place, and drawer opening. These tasks cover both short-horizon and long-horizon interactions, as well as contact-rich manipulation scenarios.

Imperfect behavior priors are some scripted policies [53] synthesized by an LLM that invokes an existing skill library. These policies provide incomplete state coverage and lack explicit failure recovery behaviors, which constitutes the source of imperfectness. We collect 1000 demonstrations for each task from the scripted policies.

b) AutoMate: AutoMate [54] is a simulation benchmark for robotic dexterous manipulation in industrial assembly, with a particular emphasis on high-precision peg-insertion tasks. The benchmark consists of a collection of insertion scenarios with tight geometric tolerances, demanding accurate pose alignment, fine-grained contact reasoning, and force-sensitive control during execution.

In this benchmark, imperfect behavior priors are generated from a scripted policy that performs peg disassembly starting from an already assembled configuration. Imperfect demonstrations are obtained by reversing these disassembly trajectories to approximate the assembly process. Because assembly and disassembly dynamics are inherently asymmetric, this inversion induces a dynamics mismatch, resulting in demonstrations with imperfect contact interactions. We collect 500 demonstrations for each task from the scripted policy.

B. Baselines - AnyTask

a) Scripted policy: This baseline directly executes the LLM-synthesized scripted policies provided for each individual task in AnyTask.

b) Diffusion policy: This baseline evaluates diffusion policies trained purely via imitation learning on the imperfect demonstrations, without any RL refinement.

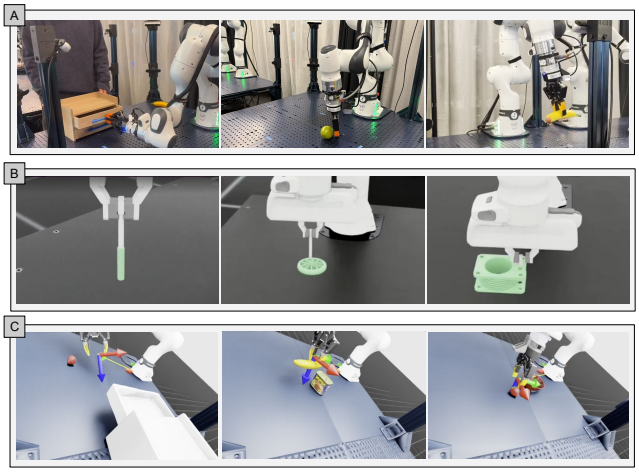

Fig. 4: Illustrations of the tasks in our experiments. (A) Real-world manipulation tasks. (B) Industrial assemble tasks from AutoMate. (C) Long-horizon tasks from AnyTask.

c) Offline-to-Online Refinement: This category includes residual RL [32] and SMP [55]. Both methods initialize from a pretrained diffusion policy and subsequently improve task performance using online reinforcement learning. Residual RL keeps the diffusion model fixed and learns a residual action correction on top of the diffusion output. SMP leverages pretrained motion diffusion models together with score distillation sampling (SDS) to construct reusable motion priors, which are incorporated as implicit style rewards for downstream tasks.

d) ExpertGen-PPO: This ablation replaces FastTD3 with PPO to evaluate the effectiveness of FastTD3 in the massively parallel simulation regime.

e) FastTD3: This baseline applies FastTD3 with no access to demonstrations or pretrained priors. It serves to quantify the difficulty of learning these tasks from sparse rewards alone.

C. Baselines - AutoMate

a) Diffusion policy: Because the scripted policies provided by AutoMate solve disassembly rather than assembly tasks, scripted-policy baselines are omitted. For diffusion policies, two variants are evaluated depending on whether small noise is injected in the x–y plane during the lifting phase of the reversed demonstrations. This noise introduces variability around the detachment plane between the peg and socket, increasing demonstration diversity near the insertion phase. These variants are denoted as *Diffusion policy with x–y noise* and *Diffusion policy without x–y noise*, respectively.

Methods \ Tasks	*Lift Banana*	*Lift Brick*	*Lift Peach*	*Open Drawer*	*Push Pear to Center*	*Stack Banana on Can*	*Put Object In Closed Drawer*	*Place Strawberry In Bowl*
Scripted Policy	56.3 ± 3.1	66.4 ± 3.0	60.9 ± 3.1	28.1 ± 2.8	18.0 ± 2.4	26.6 ± 2.7	9.4 ± 1.8	43.0 ± 2.9
Diffusion Policy	69.8 ± 2.8	72.6 ± 2.7	28.5 ± 2.8	77.6 ± 2.6	50.5 ± 3.1	16.3 ± 2.3	0.5 ± 0.5	6.6 ± 1.5
Residual RL	84.3 ± 2.0	98.4 ± 0.8	75.9 ± 2.6	**99.5 ± 0.5**	56.5 ± 3.0	0.0 ± 0.2	13.6 ± 2.1	1.3 ± 0.7
SMP	0.0 ± 0.2	0.0 ± 0.2	0.0 ± 0.2	**99.9 ± 0.3**	2.1 ± 0.2	0.0 ± 0.2	0.0 ± 0.2	1.2 ± 0.7
ExpertGen-PPO	89.6 ± 1.9	90.0 ± 1.8	81.9 ± 2.4	96.7 ± 1.1	69.8 ± 2.8	57.0 ± 3.0	54.9 ± 3.0	44.3 ± 3.0
FastTD3	0.0 ± 0.2	0.0 ± 0.2	0.0 ± 0.2	0.0 ± 0.2	0.0 ± 0.2	0.4 ± 0.4	0.0 ± 0.2	1.5 ± 0.8
ExpertGen (ours)	**99.8 ± 0.3**	**99.7 ± 0.4**	**99.3 ± 0.0**	**100 ± 0.1**	**83.3 ± 2.0**	**67.2 ± 2.9**	**80.7 ± 2.4**	**52.1 ± 3.1**

TABLE I: The success rates (%) of the evaluated state-based policies on ANYTASK benchmark. Bold number indicates the best number across all the approaches.

Methods \ Tasks	*Lift Banana*		*Lift Brick*		*Lift Peach*		*Open Drawer*		*Push Pear to Center*		*Stack Banana on Can*		*Put Object In Closed Drawer*		*Place Strawberry In Bowl*		*Average*	
	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓
Diffusion Policy	**0.114**	8.14	**0.111**	7.49	**0.112**	**3.67**	**0.087**	6.56	**0.109**	17.2	**0.122**	2.46	0.238	10.9	**0.118**	**2.04**	**0.126**	7.31
Residual RL (0.02)	0.142	6.21	0.132	**1.89**	0.120	3.71	0.131	**5.00**	0.136	**3.84**	0.152	2.26	0.264	8.17	0.163	6.12	0.155	**4.65**
Residual RL (0.1)	0.149	25.86	0.158	31.68	0.507	53.28	0.258	35.26	0.141	21.71	0.776	47.25	0.308	49.52	0.203	17.95	0.313	35.3
ExpertGen (ours)	0.134	**2.80**	0.146	6.84	0.116	7.25	0.137	7.21	0.113	5.24	0.130	**1.38**	**0.201**	**7.12**	0.135	9.92	0.139	5.97

TABLE II: The smoothness and feasibility measurement of the evaluated state-based policies on ANYTASK benchmark. In each cell, the numbers stand for normalized open-ended DTW (left) and jerk cost (right), respectively. The numbers in parenthesis after Residual RL indicate the action magnitudes. We include Residual RL training with action magnitudes of 0.02 and 0.1.

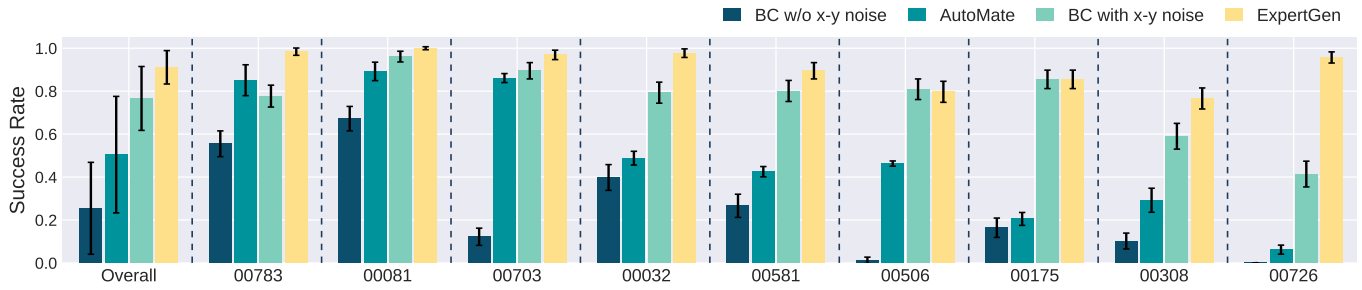

Fig. 3: The success rates (%) of the evaluated approaches on selected assets from AutoMate benchmark. EXPERTGEN outperforms all other baselines with an overall success of 90.5%. By introducing x-y noise, the BC policy demonstrates better state coverage and higher success rates compared to no x-y noise counterpart.

b) AutoMate: This baseline exactly replicates the single-task expert policies introduced in AutoMate [54].

D. Evaluation Metrics

a) Success Rate: For the ANYTASK and AutoMate benchmark, we report the success rate evaluated over 1024 and 256 rollout trajectories respectively from each baseline approach, with error bars denoting 95% Wilson binomial confidence intervals [56].

b) Open-Ended Dynamic Time Warping: To evaluate the feasibility of actions produced by the expert policies, we measure the similarity between action trajectories sampled from the learned expert policies and those from the imperfect behavior priors, which are assumed to be physically feasible and transferable to the real world. We quantify this similarity using open-ended Dynamic Time Warping (DTW) [57], which allows partial and temporally misaligned trajectory matching. This evaluation is conducted on the ANYTASK benchmark using 100 rollout trajectories.

c) Joint-Space Jerk Cost: We also evaluate the jerk cost for expert trajectories from the ANYTASK benchmark. Given a joint-space trajectory $q(t) \in \mathbb{R}^n$, where n is the number of joints, we define the joint-space jerk cost as

$$J_{\text{jerk}} = \frac{1}{T} \int_0^T \| \dddot{q}(t) \|_2^2 \, dt, \tag{14}$$

where $\dddot{q}(t)$ denotes the third-order time derivative of the joint positions. J_{jerk} is evaluated against 100 rollout trajectories.

VI. EXPERIMENTAL RESULTS

In this section, we discuss the major results including the expert performance, recovery capacity, trajectory smoothness, and visuomotor policy performance after transferring to the real world.

A. Expert Policy Acquisition

Starting from imperfect behavior priors, EXPERTGEN consistently learns near-perfect expert policies on both benchmarks. On the ANYTASK benchmark (Table I), EXPERTGEN

achieves the highest success rates across all eight evaluation tasks. For short-horizon tasks such as *Lift*, *Open Drawer*, and *Push*, success rates approach 100%. Performance on long-horizon tasks is comparatively lower, which is primarily attributed to inherently difficult or unsolvable configurations in the benchmark. Representative failure cases include objects being placed too close to the drawer frame, causing blockage during opening, or the banana knocks down the meat can when dropped during stacking. Despite these challenges, EXPERTGEN consistently outperforms all baseline methods, achieving the best overall success rates across both short- and long-horizon tasks. Beyond tabletop manipulation, as shown in Fig. 3, EXPERTGEN attains an average success rate of 90.5% across eight high-precision industrial assembly tasks in AutoMate, demonstrating its ability to solve dexterous manipulation tasks from behavior priors with inaccurate contact dynamics.

B. Failure Recovery Capacity

To evaluate failure recovery capacity, we introduce targeted perturbations to tasks from ANYTASK, including random gripper opening events and random external forces applied to the end-effector in the x–y plane. Under these perturbations, EXPERTGEN exhibits only minor performance degradation (Table III), with average success-rate drops of 0.5% and 28.6% across three tasks for gripper-opening and external-force perturbations, respectively. In contrast, the base diffusion policy is highly sensitive to such disturbances, with success rates dropping to nearly zero when random external forces are applied. These results demonstrate that EXPERTGEN policies possess strong failure recovery capabilities and can provide robust, high-quality supervision for downstream visuomotor policy learning.

C. Smoothness and Feasibility

Before transferring these expert policies to real world, Table II further analyzes the quality of the learned behaviors in terms of smoothness and feasibility. We report normalized open-ended DTW to measure trajectory similarity to the imperfect behavior priors, and joint-space jerk cost to quantify motion smoothness. For comparison, we additionally train Residual RL baselines with two action scales (0.02 and 0.1). Residual RL is sensitive to the residual action scale. A smaller action scale helps bootstrap training with higher initial success rates, but often leads to lower asymptotic performance. In contrast, a larger action scale allows better convergence but can result in unnatural and jittery behaviors.

The base diffusion policy consistently achieves the lowest DTW across most tasks, indicating strong adherence to the demonstrated motion manifold. By contrast, Residual RL (0.02) often deviates more substantially from the prior, despite occasionally producing smooth motions. Residual RL (0.1) produces noticeably jittery behaviors with significantly higher jerk cost (35.3). EXPERTGEN achieves a favorable balance between these metrics: while slightly increasing DTW relative to the diffusion policy, it substantially reduces jerk on several tasks and avoids the large distributional shifts observed in residual RL.

Methods	*Lift Banana*		*Open Drawer*		*Push Pear*	
	Open	Force	Open	Force	Open	Force
Diffusion Policy	72.5 (7.6↓)	1.9 (79.9↓)	78.2 (8.7↓)	58.0 (28.9↓)	47.1 (1.9↓)	15.8 (33.1↓)
EXPERTGEN	99.4 (0.3↓)	56.6 (43.1↓)	99.9 (0.0↓)	99.7 (0.2↓)	85.4 (1.2↑)	38.7 (45.3↓)

TABLE III: Success rates (%) with two perturbations: random gripper opening and random external force applied to the end-effector. Numbers in parentheses indicate the performance drop(↓) / increase(↑) compared to the evaluation without any perturbation.

TABLE IV: Real-world evaluation results of EXPERTGEN across manipulation tasks.

Method	Pointcloud			RGB
	Lift Banana	*Push Pear to Center*	*Open Drawer*	*Lift Banana*
EXPERTGEN	75.0%	65.0%	85.0%	–
ANYTASK [20]	73.3%	16.7%	42.5%	80%

D. Sim-to-Real Transfer of the Visuomotor Policies

In this section, we present real-world results for EXPERTGEN. Our sim-to-real policies are deployed on a single-arm Franka robot equipped with a Robotiq 2F-85 gripper. Perception is provided by three calibrated Intel RealSense D435i cameras, which yield a merged point-cloud and RGB observations. All policy inference is performed on a local workstation with an AMD Ryzen Threadripper PRO 5955WX CPU and an NVIDIA RTX 6000 Ada GPU.

a) Point-Cloud-based Policy: EXPERTGEN teacher policies are distilled into point-cloud-based student policies using DAgger and compared against standard behavioral cloning (BC) policies trained from scripted-policy demonstrations on three real-world manipulation tasks, as shown in Table IV. Both methods use the same policy architecture and visual domain randomization strategy as Gong et al. [20] to ensure a fair comparison.

While the two approaches achieve comparable performance on simple pick-and-place tasks, EXPERTGEN substantially outperforms the BC baseline on tasks involving articulation

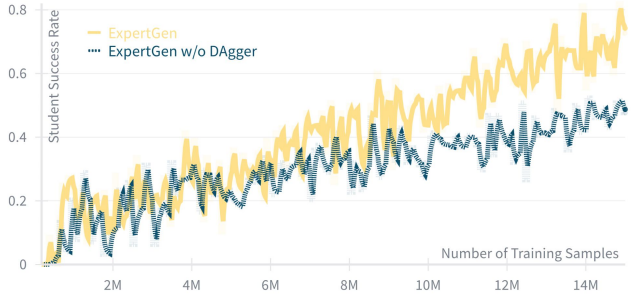

Fig. 5: Ablation of DAgger vs. BC in simulation. We plot the student policy's evaluation success rate against the number of online training samples. Training with DAgger is important for efficiently achieving higher success rates.

TABLE V: Ablation of α annealing in DAgger on the real-world *Lift Banana* task. α is linearly decayed from 1.0 (standard BC on teacher rollouts) to the specified final value between 50k and 100k global training steps, and the policy continues training until 200k global steps.

Final α	0.0	0.2	0.4	1.0
Real-world Success Rate	65.0%	75.0%	63.3%	55.0%

and more complex dynamics. This result suggests that RL-based teacher policies are critical for discovering robust strategies for contact-rich interactions, which are difficult to capture using LLM-synthesized scripted policies alone.

To verify the necessity of on-policy data collection and training using DAgger, we compare against a BC distillation baseline by monitoring student success rates during training (Figure 5). Utilizing DAgger consistently leads to faster convergence and a higher success rate compared to the BC baseline. We further investigate the sensitivity of the DAgger-BC mixture by ablating the schedule of the teacher rollout ratio α, as shown in Table V. While simulation evaluations showed negligible differences, real-world experiments reveal a sweet spot at a low, non-zero α. This suggests that retaining a minimal level of teacher correction stabilizes the policy against real-world noise better than fully decaying to the pure student policy or retaining high teacher influence.

b) RGB-based olicy: We also evaluate the transfer of EXPERTGEN to pixel-based control on *Lift Banana* task. In this setting, student policies take as input of RGB images with a resolution of 108×192. To facilitate sim-to-real transfer, we adopt domain randomization techniques from Singh et al. [58], including randomization of camera extrinsics, dome light textures, object textures, and HSV jitter. Additionally, object scaling is randomized within $\pm 10\%$ to improve robustness to variations in physical object dimensions.

In real-world zero-shot trials, the distilled policy achieves 80% task success rate and demonstrates remarkable robustness to visual distractors–even in cluttered workspace containing multiple objects with similar shape and color. In contrast, a standard behavioral cloning (BC) policy trained directly on scripted-policy demonstrations fails to achieve any successful trials in the real world. This comparison highlights an important limitation of scripted policies: although they provide a nominal sequence of actions, they lack the reactive behaviors necessary to handle real-world sensing noise and visual variation. In contrast, the supervision provided by an expert teacher produces significantly stronger training signals, enabling the student policy to learn robust visuomotor behaviors that transfer reliably from simulation to the physical robot.

VII. ADDITIONAL ANALYSIS

A. Imperfect Behavior Priors

The scripted policies provide incomplete state coverage, which directly limits the quality of the learned diffusion policy. Consequently, the diffusion policy is imperfect and

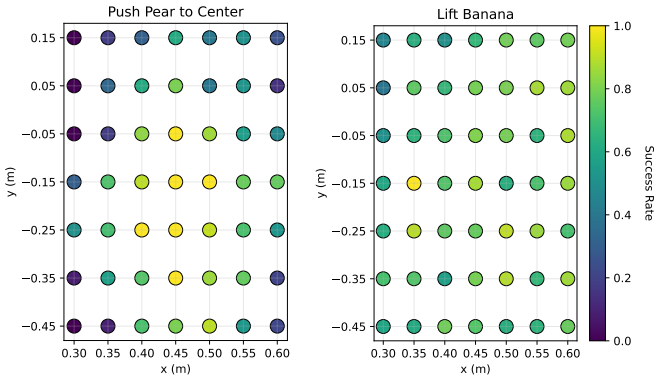

Fig. 6: Success rate (%) distribution of imperfect prior diffusion policy over different initial object configurations for *Push Pear to Center* (left) and *Lift Banana* (right).

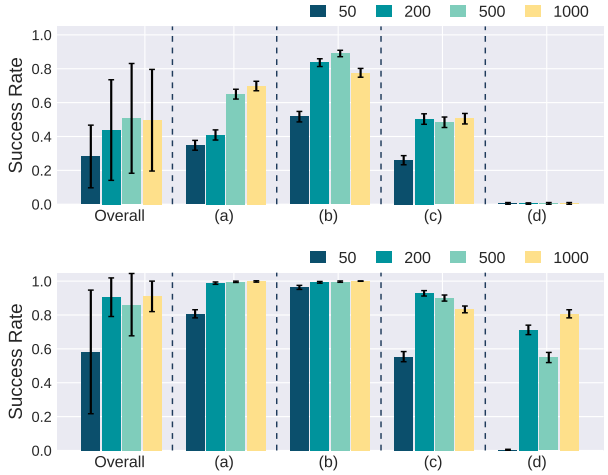

Fig. 7: Success rates of base diffusion policies (top) and the EXPERTGEN policies (bottom) trained with 50, 200, 500, and 1000 imperfect demonstrations on (a) *Lift Banana*, (b) *Open Drawer*, (c) *Push Pear to Center*, and (d) *Put Object in Closed Drawer*.

exhibits low success rates for certain initial configurations, as shown in Fig. 6. This trend is consistent on the AutoMate benchmark, where the diffusion policy without added x-y noise achieves the lowest success rates among all evaluated methods. Injecting x-y noise improves spatial coverage and partially mitigates this issue; however, the resulting policy remains suboptimal. By contrast, the EXPERTGEN pipeline consistently improves performance by refining the behavior prior through reinforcement learning. Furthermore, Table III shows that the diffusion policy lacks failure recovery capability, as evidenced by the substantial performance degradation under external perturbations.

B. Number of Imperfect Demonstrations

We further examine how many demonstrations are required to learn a behavior prior that is sufficiently diverse to support effective refinement in simulation. Figure 7 reports the success rates of both the base diffusion policies and the

EXPERTGEN policies trained with 50, 200, 500, and 1000 imperfect demonstrations on four ANYTASK tasks. Performance remains largely unchanged once the dataset contains more than 200 demonstrations. A noticeable performance drop occurs only when the dataset is reduced to 50 demonstrations, where the average success rate decreases to 58.2%. These results suggest that EXPERTGEN does not require large-scale demonstration datasets; instead, a relatively small number of imperfect demonstrations (approximately 200) is sufficient to initialize a behavior prior that can be effectively refined through reinforcement learning.

C. Choice of RL algorithms

We adopt FastTD3 as the underlying reinforcement learning algorithm for expert policy acquisition. As shown in Table I, while ExpertGen-PPO achieves reasonable success rates on several tasks, it consistently underperforms FastTD3 across all eight tasks in the ANYTASK benchmark. FastTD3's off-policy formulation enables more efficient reuse of expensive action-chunked rollout and more stable optimization under sparse, task-level rewards, which are essential to the success of EXPERTGEN.

D. Human Motions as Behavior Priors

While the main results use scripted policies as imperfect behavior priors, human demonstrations may provide richer motion diversity and more adaptive behaviors. To evaluate this hypothesis, we consider the task *Stack Banana on Can*, on which EXPERTGEN trained with scripted policies achieves a relatively low success rate (67% after RL training). Specifically, six human demonstrations are collected in IsaacLab and subsequently augmented to 1,000 demonstrations using SkillMimicGen [38]. A single-task diffusion policy is then trained on this dataset, while all other components of the EXPERTGEN pipeline remain unchanged. This variant is referred to as EXPERTGEN w/ SkillMimicGen.

The results, summarized in Fig. 8, show that both the diffusion policy and EXPERTGEN trained with SkillMimicGen data substantially outperform their counterparts trained using scripted policy demonstrations. These results suggest that human motion priors provide more diverse and adaptable behavioral patterns, which improve both imitation learning and subsequent reinforcement learning refinement.

VIII. CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

This paper presented EXPERTGEN, a scalable framework for sim-to-real expert policy acquisition that combines diffusion-based behavior priors trained on *imperfect demonstrations* with diffusion steering RL in massively parallel simulation. By steering only the initial noise of a pretrained diffusion policy, EXPERTGEN preserves in-distribution, human-like behaviors while enabling effective optimization with sparse task-level rewards. Experiments on the ANYTASK and AutoMate benchmarks demonstrate that EXPERTGEN reliably produces high-quality expert policies for both long-horizon and dexterous manipulation tasks, and large-scale DAgger distillation further validates successful transfer to real-world visuomotor policies. Overall, these results show that learning from imperfect demonstrations and refining behaviors by massive simulation training provides a scalable and practical path toward high-quality synthetic robotics data generation for sim-to-real learning.

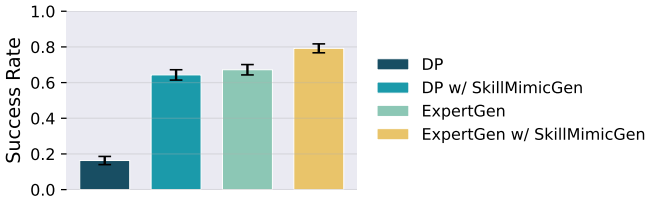

Fig. 8: The success rates (%) of the evaluated state-based policies on the *Stack Banana on Can* task. Diffusion Policy w/ SkillMimicGen trains a diffusion policy on human demonstrations augmented with SkillMimicGen [38], while EXPERTGEN w/ SkillMimicGen refines the diffusion policy trained with SkillMimicGen data. Using human motions as behavior priors significantly improves the performance of both diffusion policy and EXPERTGEN teacher performances.

One limitation of EXPERTGEN is that the learned expert policies remain constrained by the coverage of the provided demonstrations or imperfect behavior priors. When the prior lacks certain behaviors, performance degrades in scenarios that require qualitatively new skills—for example, flipping a meat can back upright after it has been knocked over. Future work could explore more flexible generative priors, such as classifier-free guidance with weak or no state conditioning, to synthesize novel motions beyond the demonstration distribution. This may enable more effective behavior transfer across tasks while reducing reliance on task-specific demonstrations.

REFERENCES

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

[3] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[4] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal

Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[5] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[6] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[7] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[8] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. *arXiv preprint arXiv:2407.20179*, 2024.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[13] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[14] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[15] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[17] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[18] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[19] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11142–11152, 2025.

[20] Ran Gong*, Xiaohan Zhang*, Jinghuan Shang*, Maria Vittoria Minniti*, Jigarkumar Patel, Valerio Pepe, Riedana Yan, Ahmet Gundogdu, Ivan Kapelyukh, Ali Abbas, Xiaoqiang Yan, Harsh Patel, Laura Herlant, and Karl Schmeckpeper. Anytask: an automated task and data generation framework for advancing sim-to-real policy learning. *arXiv preprint arXiv:2512.17853*, 2025.

[21] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.

[22] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pages 91–100. PMLR, 2022.

[23] Tairan He, Zi Wang, Haoru Xue, Qingwei Ben, Zhengyi Luo, Wenli Xiao, Ye Yuan, Xingye Da, Fernando Castañeda, Shankar Sastry, et al. Viral: Visual sim-to-real at scale for humanoid loco-manipulation. *arXiv preprint arXiv:2511.15200*, 2025.

[24] Haoru Xue, Tairan He, Zi Wang, Qingwei Ben, Wenli Xiao, Zhengyi Luo, Xingye Da, Fernando Castañeda, Guanya Shi, Shankar Sastry, et al. Opening the sim-to-real door for humanoid pixel-to-action policy transfer. *arXiv preprint arXiv:2512.01061*, 2025.

[25] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li,

Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[26] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[27] Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

[28] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[29] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[30] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025.

[31] Siheng Zhao, Yanjie Ze, Yue Wang, C Karen Liu, Pieter Abbeel, Guanya Shi, and Rocky Duan. Resmimic: From general motion tracking to humanoid whole-body loco-manipulation via residual learning. *arXiv preprint arXiv:2510.05070*, 2025.

[32] Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[33] Lakshita Dodeja, Karl Schmeckpeper, Shivam Vats, Thomas Weng, Mingxi Jia, George Konidaris, and Stefanie Tellex. Accelerating residual reinforcement learning with uncertainty estimation. *arXiv preprint arXiv:2506.17564*, 2025.

[34] Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. *arXiv preprint arXiv:2412.07762*, 2024.

[35] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[36] Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[37] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[38] Caelan Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. *arXiv preprint arXiv:2410.18907*, 2024.

[39] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16923–16930. IEEE, 2025.

[40] Zhengrong Xue, Shuying Deng, Zhenyang Chen, Yixuan Wang, Zhecheng Yuan, and Huazhe Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *arXiv preprint arXiv:2502.16932*, 2025.

[41] Zihan Zhou, Animesh Garg, Ajay Mandlekar, and Caelan Garrett. Reinforcegen: Hybrid skill policies with automated data generation and reinforcement learning. *arXiv preprint arXiv:2512.16861*, 2025.

[42] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, and Tse kai Chan et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

[43] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A uni-

fied benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

[44] Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.

[45] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

[46] Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. *arXiv preprint arXiv:2511.16651*, 2025.

[47] Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. Gensim2: Scaling robot data generation with multi-modal and reasoning llms. In *8th Annual Conference on Robot Learning*, 2024.

[48] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[49] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[50] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[51] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[52] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[53] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[54] Bingjie Tang, Iretiayo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S Sukhatme, Fabio Ramos, and Yashraj Narang. Automate: Specialist and generalist assembly policies over diverse geometries. *arXiv preprint arXiv:2407.08028*, 1(2), 2024.

[55] Yuxuan Mu, Ziyu Zhang, Yi Shi, Minami Matsumoto, Kotaro Imamura, Guy Tevet, Chuan Guo, Michael Taylor, Chang Shu, Pengcheng Xi, et al. Smp: Reusable score-matching motion priors for physics-based character control. *arXiv preprint arXiv:2512.03028*, 2025.

[56] Wikipedia contributors. Binomial proportion confidence interval — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Binomial_proportion_confidence_interval&oldid=1333177192, 2026. [Online; accessed 31-January-2026].

[57] Paolo Tormene, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial intelligence in medicine*, 45(1):11–34, 2009.

[58] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands, 2025.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

APPENDIX A
HYPER-PARAMETERS

A. Diffusion Policy

This paper considers diffusion policy for all imitation learning training. Specifically, diffusion policy considers demonstrations as samples from a diffusion denoising process. Given a state s_t at time step t, the policy represents the conditional distribution of an action chunk [48] $\mathbf{a}_t = a_{t:t+H}$ of length H via a diffusion model [35]. During training, Gaussian noise is progressively added to expert trajectories according to a forward process

$$q(\mathbf{a}_t^k \mid \mathbf{a}_t^0) = \mathcal{N}\big(\mathbf{a}_t^k; \sqrt{\bar{\alpha}_k}\,\mathbf{a}_t^0,\ (1-\bar{\alpha}_k)I\big)\,, \tag{15}$$

where $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$ defines the noise schedule. The diffusion policy is trained to predict the injected noise using a denoising network ϵ_θ, by minimizing the standard diffusion objective:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{a}_t^0, s_t,\, k,\, \epsilon\sim\mathcal{N}(0,I)} \left[\left\| \epsilon - \epsilon_\theta(\mathbf{a}_t^k, s_t, k) \right\|^2 \right]. \tag{16}$$

At inference time, action trajectories are generated by iteratively applying the reverse diffusion process starting from Gaussian noise $\mathbf{a}_t^K \sim \mathcal{N}(0, I)$. To reduce denoising steps, we adopt Denoising Diffusion Implicit Models (DDIM) [49] sampling, which replaces the stochastic reverse process with a deterministic update:

$$\mathbf{a}_t^{k-1} = \sqrt{\bar{\alpha}_{k-1}}\,\hat{\mathbf{a}}_t^0 + \sqrt{1-\bar{\alpha}_{k-1}}\,\epsilon_\theta(\mathbf{a}_t^k, s_t, k), \tag{17}$$

where $\hat{\mathbf{a}}_t^0$ denotes the prediction of the clean action chunk.

B. EXPERTGEN

The set of hyper-parameters used in this paper is shown in Table VI

C. EXPERTGEN-*PPO*

EXPERTGEN-PPO uses PPO for expert policy Acquisition instead of FastTD3. We list the PPO related hyper-parameters in Table VII. We found it important to use a large initial log-standard deviation (e.g., 0.0) to matches with the training initial noise distribution to maintain a relative high initial task success.

D. Residual RL

Residual RL augments a fixed base prior policy π_P with a learnable residual policy π_θ that predicts corrective actions. Given a state s_t, the base policy outputs a nominal action $a_t^0 = \pi_0(s_t)$, while the residual policy predicts an additive correction conditioned on both the state and the base action $\delta a_t = \pi_\theta(s_t, a_t^0)$. The final action executed in the environment is given by $a_t = a_t^0 + \delta a_t$.

The critic evaluates the composed action under the environment dynamics. The state–action value function is defined as

$$Q_\phi(s_t, a_t^0, \delta a_t) \ \triangleq\ Q_\phi(s_t, a_t), \tag{18}$$

and is trained using standard temporal-difference learning with targets computed from the combined action at the next state.

Hyperparameters	ANYTASK	AutoMate
Generative Behavior Prior Modeling		
Architecture	U-Net [59]	U-Net
Number of groups	8	8
Kernel size	5	5
Number of channels	[128, 256]	[128, 256]
Diffusion step embedding dim.	16	16
Condition type	FiLM	FiLM
Action chunk length	16	8
Number of epochs	500	500
Batch size	256	256
Number of demonstrations	1000	500
Expert Policy Acquisition		
Receding horizon	8	8
Critic learning rate	3e-4	3e-4
Actor learning rate	3e-4	3e-4
Buffer size	8196 × 1024	8196 × 256
Batch size	8196	8196
Policy noise	0.001	0.001
std_min	0.005	0.005
std_max	0.4	0.4
Number of updates	8	8
Number of steps	4	4
Number of atoms	101	101
v_min	0.0	0.0
v_max	100.0	400.0
Discount factor	0.99	0.99

TABLE VI: EXPERTGEN hyperparameters. The right two columns indicate the hyerparameters for ANYTASK and AutoMate, respectively.

Hyperparameters	ANYTASK
Expert Policy Acquisition	
Receding horizon	8
Number of steps	64
Learning rate	5e-4
Learning rate schedule	fixed
Discount factor	0.995
e_clip	0.2
Entropy coefficent	-2e-4
Initial logstd	0.0

TABLE VII: EXPERTGEN-PPO hyperparameters.

The residual actor is optimized to maximize the critic while keeping the base policy fixed:

$$\max_\theta\ \mathbb{E}_{s_t\sim\mathcal{D}}\left[Q_\phi(s_t,\ \pi_P(s_t),\ \pi_\theta(s_t, \pi_P(s_t)))\right]. \tag{19}$$

We also adapt FastTD for learning the residual policy π_θ. The list of hyper-parameters used for residual RL is shown in Table VIII.

E. Score-Matching Motion Priors (SMP) [55]

Given a pretrained diffusion policy ϵ_θ that models the distribution of expert action chunks, Score-Matching Motion Priors (SMP) repurpose this frozen diffusion model as a task-agnostic motion prior via score distillation sampling (SDS). During policy learning, an action chunk $\tilde{\mathbf{a}}_t^0$ generated by the current policy is evaluated under the diffusion prior by

Hyperparameters	ANYTASK
Residual RL	
Warm-start steps	10240
Action scale	0.02
Receding horizon	8
FastTD3	
Critic learning rate	3e-4
Actor learning rate	3e-4
Buffer size	8196 × 1024
Batch size	8196
Policy noise	0.001
std_min	0.005
std_max	0.4
Number of updates	8
Number of steps	4
Number of atoms	101
v_min	0.0
v_max	100.0
Discount factor	0.995

TABLE VIII: Residual RL hyperparameters.

injecting Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ using the same forward diffusion process,

$$\tilde{\mathbf{a}}_t^k = \sqrt{\bar{\alpha}_k}\,\tilde{\mathbf{a}}_t^0 + \sqrt{1-\bar{\alpha}_k}\,\epsilon. \tag{20}$$

The denoising network then predicts the injected noise as $\hat{\epsilon} = \epsilon_\theta(\tilde{\mathbf{a}}_t^k, s_t, k)$. The squared error $\|\hat{\epsilon} - \epsilon\|_2^2$ provides a score-matching signal that measures how well the policy-generated action aligns with the expert demonstration distribution captured by the diffusion model.

To integrate this signal into reinforcement learning, SMP converts the SDS loss into a bounded prior reward,

$$r_{\text{smp}} = \exp\!\left(-w_s \,\|\hat{\epsilon} - \epsilon\|_2^2\right), \tag{21}$$

where w_s controls the strength of the motion prior. To reduce variance arising from a single diffusion timestep, the SDS loss is evaluated over a fixed set of diffusion steps and averaged before computing the reward. This prior reward is combined with a task-specific reward to train the policy using standard reinforcement learning algorithms, while keeping the diffusion model fixed throughout training. As a result, SMP provides a reusable and stationary behavior regularizer that encourages policy-generated actions to remain in-distribution with respect to expert demonstrations, without requiring adversarial training or access to the original dataset during policy optimization.

To maintain fair comparison, SMP also employs FastTD3 as the underlying RL algorithm. Table IX lists all the hyperparameters for our SMP implementation.

Hyperparameters	ANYTASK
Warm-start steps	10240
Action scale	0.02
Receding horizon	8

TABLE IX: SMP hyperparameters.

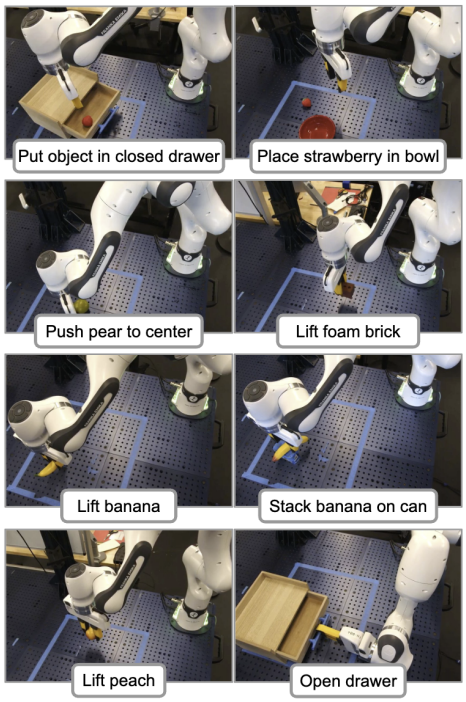

Fig. 9: An overview of the selected eight tasks from ANYTASK benchmark.

APPENDIX B
BENCHMARKS

A. ANYTASK

ANYTASK [20] is an automated framework for automatic task design and large-scale synthetic data generation in robotic manipulation. For benchmarking, eight tabletop manipulation tasks are selected (an overview is shown in Fig. 9), covering a diverse set of behaviors including lifting, pushing, stacking, pick-and-place, and drawer opening. Together, these tasks span short-horizon and long-horizon interactions, articulated objects, and contact-rich manipulation scenarios.

Imperfect behavior priors are some scripted policies [53] synthesized by an LLM that invokes an existing skill library. These policies provide incomplete state coverage and lack explicit failure recovery behaviors, which constitutes the source of imperfectness. All scripted policies are shown in Fig. 20. We collect 1000 demonstrations for each task from the scripted policies.

A state-based diffusion policy is trained to generate action chunks with a horizon of 16, conditioned on a structured state vector. The definitions of the state and action representations are summarized in Table X. For tasks involving fewer than two rigid objects or no articulated objects, the corresponding state dimensions are zero-masked. Task specification is encoded via a learned task embedding, obtained as the output of a lightweight MLP applied to a one-hot task index.

State	Dimension
End-effector pose	16
robot arm joint positions	9
Two object positions	6
Two object 6D rotations	12
Articulated object position	3
Articulated object 6D rotation	6
Articulated joint position	1
Task embedding	16
Action	**Dimension**
End-effector position	3
End-effector 6D rotation	6
Gripper action	1

TABLE X: State condition and action definitions for state-based diffusion policy for the ANYTASK benchmark.

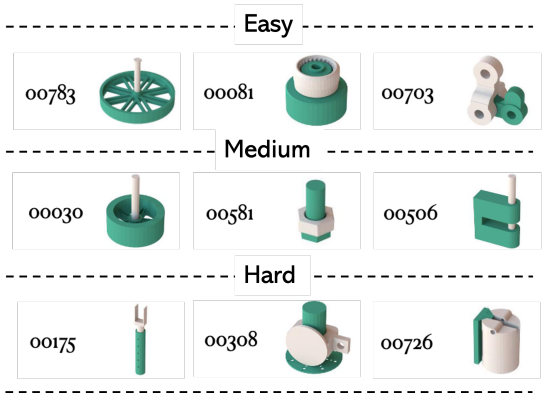

Fig. 10: Visualization of the selected nine assets from AutoMate benchmark covering three difficulty levels (easy, medium, and hard).

B. AutoMate

AutoMate [54] is a simulation benchmark for robotic dexterous manipulation in industrial assembly, with a particular emphasis on high-precision peg-insertion tasks. The benchmark consists of a collection of insertion scenarios with tight geometric tolerances, demanding accurate pose alignment, fine-grained contact reasoning, and force-sensitive control during execution. For evaluation, nine assets are selected from AutoMate, evenly spanning three difficulty levels: easy (success rate $>$ 80%), medium (success rate between 40% and 80%), and hard (success rate $<$ 40%), where difficulty is defined based on the performance of single-task expert policies provided by AutoMate. An overview of the selected assets is shown in Fig. 10.

In this benchmark, imperfect behavior priors are generated from a scripted policy that performs peg disassembly starting from an already assembled configuration. Imperfect demonstrations are obtained by reversing these disassembly trajectories to approximate the assembly process. Because assembly and disassembly dynamics are inherently asymmetric, this inversion induces a dynamics mismatch, resulting in demonstrations with imperfect contact interactions. We collect 500 demonstrations for each task from the scripted policy. The scripted policy for disassembly is shown in Fig. 21 including three skill functions, `_disassemble_plug_from socket`, `_lift_gripper`, and `_randomize_gripper pose`.

A state-based diffusion policy is trained to generate action chunks with a horizon of 8, conditioned on a structured state vector. The definitions of the state and action representations are summarized in Table XI. The 32-dimensional task embedding is computed from a pretrained point-cloud autoencoder similar to AutoMate [54].

State	Dimension
End-effector pose	16
Held positions	3
Held 6D rotation	6
Task embedding	32
Action	**Dimension**
End-effector position	3
End-effector 6D rotation	6

TABLE XI: State condition and action definitions for state-based diffusion policy for the AutoMate benchmark.

APPENDIX C
ADDITIONAL RESULTS

A. Sim-to-Real Transfer of the Visuomotor Policies

We present the full list of sim-to-real evaluation results including *Lift Peach* task in Table XII.

TABLE XII: Real-world evaluation results of EXPERTGEN across four manipulation tasks.

Method	*Lift Banana*	*Push Pear to Center*	*Open Drawer*	*Lift Peach*
EXPERTGEN	75.0%	65.0%	85.0%	80.0%
ANYTASK [20]	73.3%	16.7%	42.5%	62.1%

B. Additional AutoMate Results

We present the full list of AutoMate evluation results in Fig. 11 including an additional hard assembly task (ID. 00726). On this hard task, on which AutoMate can only achieve a success rate of 6.2%, EXPERTGEN still achieves the near 100% success.

C. Failure recovery capacity

We present the full failure recovery capacity results including the Residual RL in Table XIII including residual RL. Residual RL demonstrates similar robustness as EXPERTGEN under mild perturbation such as randomly open the gripper, however, its success rates drastically drop when more aggressive perturbations are added such as randomly applying a force to the end-effector.

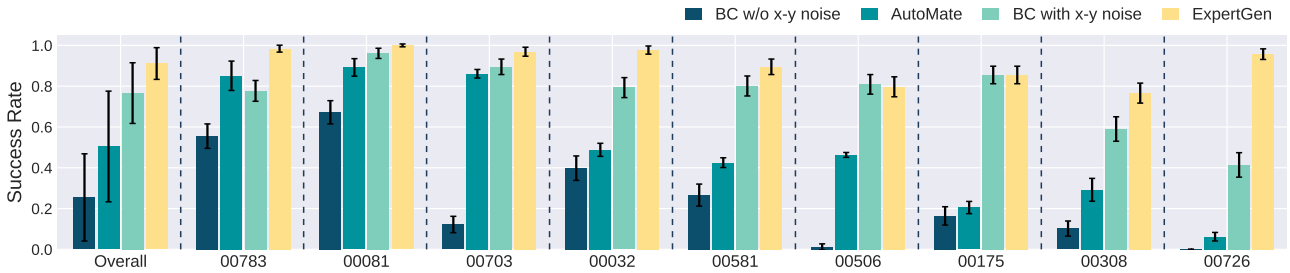

Fig. 11: he success rates (%) of the evaluated approaches on selected assets from AutoMate benchmark. EXPERTGEN outperforms all other baselines with an overall success of 91.1%. By introducing x-y noise, the BC policy demonstrates better state coverage and higher success rates compared to no x-y noise counterpart.

Methods	*Lift Banana*		*Open Drawer*		*Push Pear*	
	Open	Force	Open	Force	Open	Force
Diffusion Policy	72.5 (7.6↓)	1.9 (79.9↓)	78.2 (8.7↓)	58.0 (28.9↓)	47.1 (1.9↓)	15.8 (33.1↓)
Residual RL	97.6 (0.5↓)	3.5 (94.5↓)	99.5 (0.0↓)	92.4 (7.1↓)	56.4 (3.4↑)	15.6 (37.4↓)
EXPERTGEN	99.4 (0.3↓)	56.6 (43.1↓)	99.9 (0.0↓)	99.7 (0.2↓)	85.4 (1.2↑)	38.7 (45.3↓)

TABLE XIII: Success rates (%) with two perturbations: random gripper opening and random external force applied to the end-effector. Numbers in parentheses indicate the performance drop(↓) / increase(↑) compared to the evaluation without any perturbation.

D. Smoothness and Feasibility

Table XIV reports the full evaluation of motion smoothness and feasibility, measured by jerkness and open-ended DTW, respectively. We additionally include Residual RL baselines with two action magnitudes (0.02 and 0.1). Although action-magnitude scheduling is not implemented here, it is common practice to begin training with a small action magnitude (e.g., 0.02) for stability and gradually increase it (e.g., to 0.1) to improve convergence. To isolate the effect of a larger action magnitude, we evaluate the smoothness and feasibility of policies trained directly with an action magnitude of 0.1. As shown in Table XIV, Residual RL with action magnitude 0.1 exhibits substantially higher jerkness (35.3) and increased DTW distance (0.3125), indicating jerky and less feasible action trajectories. An overall open-ended DTW and jeckness cost averaged across eight tasks are shown in Fig. 12.

E. Human Motions as Behavior Priors

We select the *Stack Banana on Can task*, on which EXPERTGEN attains a relatively low success rate (67.2%), to evaluate the effectiveness of using human motion data as the source of the behavior prior. Specifically, six human demonstrations are collected in IsaacLab and augmented to 1,000 demonstrations using SkillMimicGen [38]. A single-task diffusion policy is then trained using this dataset, while all remaining components of the EXPERTGEN pipeline are kept unchanged. This variant is referred to as EXPERTGEN w/ SkillMimicGen. The results, summarized in Table XV, show that both the diffusion policy and EXPERTGEN trained with SkillMimicGen data substantially outperform their counterparts trained using scripted policy demonstrations.

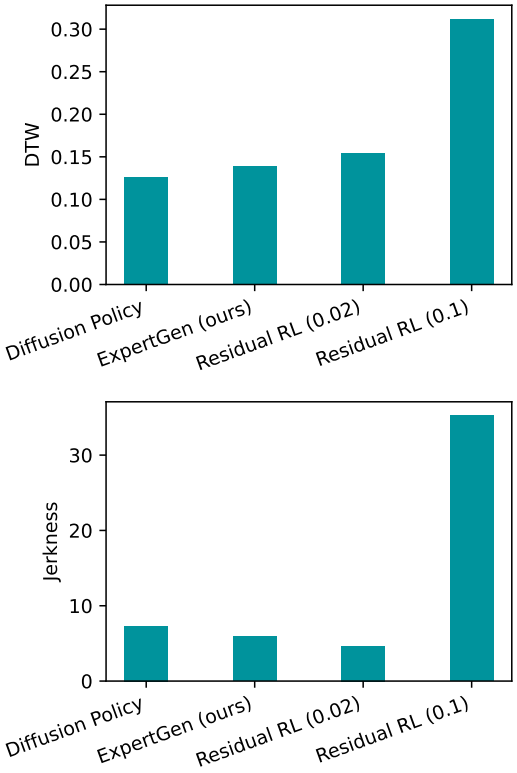

Fig. 12: Overall smoothness and feasibility evaluation of state-based policies on ANYTASK benchmark. The feasibility (top) is measured by normalized open-ended DTW and smoothness (bottom) is measured by jerk cost. Residual RL (0.1) exhibits substantially higher jerkness and DTW distance.

Methods \ Tasks	*Lift Banana*		*Lift Brick*		*Lift Peach*		*Open Drawer*		*Push Pear to Center*		*Stack Banana on Can*		*Put Object In Closed Drawer*		*Place Strawberry In Bowl*		*Average*	
	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓	DTW↓	Jerk.↓
Diffusion Policy	**0.114**	8.14	**0.111**	7.49	**0.112**	**3.67**	**0.087**	6.56	**0.109**	17.2	**0.122**	2.46	0.238	10.9	**0.118**	**2.04**	**0.126**	7.31
Residual RL (0.02)	0.142	6.21	0.132	**1.89**	0.120	3.71	0.131	**5.00**	0.136	**3.84**	0.152	2.26	0.264	8.17	0.163	6.12	0.155	**4.65**
Residual RL (0.1)	0.149	25.86	0.158	31.68	0.507	53.28	0.258	35.26	0.141	21.71	0.776	47.25	0.308	49.52	0.203	17.95	0.313	35.3
ExpertGen (ours)	0.134	**2.80**	0.146	6.84	0.116	7.25	0.137	7.21	0.113	5.24	0.130	**1.38**	**0.201**	**7.12**	0.135	9.92	0.139	5.97

TABLE XIV: The smoothness and feasibility measurement of the evaluated state-based policies on ANYTASK benchmark. In each cell, the numbers stand for normalized open-ended DTW (left) and jerk cost (right), respectively. The numbers in parenthesis after Residual RL indicate the action magnitudes. We include Residual RL training with action magnitudes of 0.02 and 0.1.

Methods	Diffusion Policy	Diffusion Policy w. SkillMimicGen	EXPERTGEN	EXPERTGEN w. SkillMimicGen
Success rate (%)	16.3 $\pm$ 2.3	64.3 $\pm$ 2.9	67.2 $\pm$ 2.9	**79.2** $\pm$ **2.5**

TABLE XV: The success rates (%) of the evaluated state-based policies on ANYTASK benchmark including Diffusion Policy w. SkillMimicGen that trains a diffusion policy on human demonstrations augmented with SkillMimicGen [38] and EXPERTGEN w/ SkillMimicGen that refines Diffusion Policy w/ SkillMimicGen. Bold number indicates the best number across all the approaches.

Scripted policy - *Lift Banana*

```
def scripted_policy(env):
   import torch

   # Object id for banana
   banana_id = 1

   # Step 1: Pick up the banana. This skill will grasp the banana and lift it with an internally
defined distance.
   pick(env, banana_id)

   # Step 2: Compute the target position for lifting.
   # We want to ensure the banana is lifted upward by at least 20 cm relative to its initial
position.
   # Get the initial position of the banana (batched tensor of shape (N, 3)).
   initial_banana_pos = get_object_initial_position(env, banana_id)

   # Define a lift offset of 0.25 m (25 cm) to ensure it meets the 20 cm requirement.
   lift_offset = torch.tensor([0.0, 0.0, 0.25], device=initial_banana_pos.device).reshape(1, 3)

   # Broadcast the lift offset and add to the initial banana position
   target_pos = initial_banana_pos + lift_offset

   # Step 3: Move the robot's end effector to the target position while keeping the gripper closed.
   move_to(env, target_pos, target_orientation=None, gripper_open=False)

   # Step 4: Ensure the gripper remains closed to keep a firm grasp.
   close_gripper(env)
```

Fig. 13: Scripted policy for *Lift Banana*

Scripted policy - *Lift Brick*

```
def scripted_policy(env):
   import torch

   # Object id for soft_brick
   soft_brick_id = 1

   # Step 1: Pick up the soft_brick. This skill will grasp the soft_brick and lift it with an
internally defined distance.
   pick(env, soft_brick_id)

   # Step 2: Compute the target position for lifting.
   # We want to ensure the soft_brick is lifted upward by at least 20 cm relative to its initial
position.
   # Get the initial position of the soft_brick (batched tensor of shape (N, 3)).
   initial_soft_brick_pos = get_object_initial_position(env, soft_brick_id)

   # Define a lift offset of 0.25 m (25 cm) to ensure it meets the 20 cm requirement.
   lift_offset = torch.tensor([0.0, 0.0, 0.25], device=initial_soft_brick_pos.device).reshape(1, 3)

   # Broadcast the lift offset and add to the initial soft_brick position
   target_pos = initial_soft_brick_pos + lift_offset

   # Step 3: Move the robotŏ027s end effector to the target position while keeping the gripper
closed.
   move_to(env, target_pos, target_orientation=None, gripper_open=False)

   # Step 4: Ensure the gripper remains closed to keep a firm grasp.
   close_gripper(env)
```

Fig. 14: Scripted policy for *Lift Brick*

Scripted policy - *Lift Peach*

```
def scripted_policy(env):
   import torch

   # Object id for peach
   peach_id = 1

   # Step 1: Pick up the peach. This skill will grasp the peach and lift it with an internally
defined distance.
   pick(env, peach_id)

   # Step 2: Compute the target position for lifting.
   # We want to ensure the peach is lifted upward by at least 20 cm relative to its initial position.
   # Get the initial position of the peach (batched tensor of shape (N, 3)).
   initial_peach_pos = get_object_initial_position(env, peach_id)

   # Define a lift offset of 0.25 m (25 cm) to ensure it meets the 20 cm requirement.
   lift_offset = torch.tensor([0.0, 0.0, 0.25], device=initial_peach_pos.device).reshape(1, 3)

   # Broadcast the lift offset and add to the initial peach position
   target_pos = initial_peach_pos + lift_offset

   # Step 3: Move the robotŏ027s end effector to the target position while keeping the gripper
closed.
   move_to(env, target_pos, target_orientation=None, gripper_open=False)

   # Step 4: Ensure the gripper remains closed to keep a firm grasp.
   close_gripper(env)
```

Fig. 15: Scripted policy for *Lift Peach*

Scripted policy - *Open Drawer*

```
def scripted_policy(env):
   obj_id = 1
   open_drawer(env, obj_id)
```

Fig. 16: Scripted policy for *Open Drawer*

Scripted policy - *Push Pear to Center*

```
def scripted_policy(env):
   import torch
   # Push the pear (object_id=1) to target xy (0.4, -0.5) using iterative, state-based pushes
   pear_id = 1
   target_xy = make_a_tensor(env, [0.4, -0.44])
   push_or_pull_object_to_xy(env, pear_id, target_xy)
```

Fig. 17: Scripted policy for *Push Pear to Center*

Scripted policy - *Stack Banana on Can*

```
def scripted_policy(env):
   import torch
   banana_id = 2
   can_id = 1

   # 1) Pick the banana
   pick(env, banana_id)

   # 2) Lift to 25 cm above initial height
   init_banana = get_object_initial_position(env, banana_id) # (N,3)
   lift_offset = torch.tensor([0.0, 0.0, 0.18], device=init_banana.device).reshape(1, 3)
   lifted_pos = init_banana + lift_offset
   move_to(env, lifted_pos, target_orientation=None, gripper_open=False)
   can_position = get_object_position(env, can_id)
   target_position = can_position.clone()
   place(env, target_position)
```

Fig. 18: Scripted policy for *Stack Banana on Can*

Scripted policy - *Put Object in Closed Drawer*

```
def scripted_policy(env):
   import torch
   obj_id = 1 # drawer
   strawberry_id = 2

   # 1) Open the upper drawer
   open_drawer(env, obj_id, reset_after_open=True)

   # 2) Pick the strawberry
   pick(env, strawberry_id)

   # 3) Lift strawberry  23 cm above its initial position
   init_straw = get_object_initial_position(env, strawberry_id) # (N,3)
   lift_offset = torch.tensor([0.0, 0.0, 0.23], device=init_straw.device).reshape(1, 3)
   lifted_pos = init_straw + lift_offset
   move_to(env, lifted_pos, target_orientation=None, gripper_open=False)

   # 4) Place location: use upper drawer's (x, y) and fixed z = 0.23
   drawer_pos = get_articulated_obj_part_pos(env, obj_id, 'upper_drawer', _use_bbox_center=True) #
(N,3)
   z_const = torch.full((drawer_pos.shape[0], 1), 0.23, device=drawer_pos.device)
   place_target = torch.cat([drawer_pos[:, :2], z_const], dim=1) # (N,3)

   # Move to place location and open gripper directly (no pre-place)
   move_to(env, place_target, target_orientation=None, gripper_open=False)
   open_gripper(env)
```

Fig. 19: Scripted policy for *Put Object in Closed Drawer*

Scripted policy - *Place Strawberry in Bowl*

```
def scripted_policy(env):
   import torch
   strawberry_id = 2
   bowl_id = 1

   # 1) Pick the strawberry
   pick(env, strawberry_id)

   # 2) Lift to 25 cm above initial height
   init_strawberry = get_object_initial_position(env, strawberry_id) # (N,3)
   lift_offset = torch.tensor([0.0, 0.0, 0.18], device=init_strawberry.device).reshape(1, 3)
   lifted_pos = init_strawberry + lift_offset
   move_to(env, lifted_pos, target_orientation=None, gripper_open=False)
   bowl_position = get_object_position(env, bowl_id)
   target_position = bowl_position.clone()
   place(env, target_position)
```

Fig. 20: Scripted policy for *Put Object in Closed Drawer*

Scripted policy - _disassemble_plug_from_socket

```
def _disassemble_plug_from_socket(self):
  """Lift plug from socket till disassembly and then randomize end-effector pose."""

  if_intersect = np.ones(self.num_envs, dtype=np.float32)

  env_ids = np.argwhere(if_intersect == 1).reshape(-1)
  lift_distance = self.disassembly_dists * 3.0
  self._lift_gripper(lift_distance, self.cfg_task.disassemble_sim_steps, env_ids)

  self.step_sim_no_action()

  if_intersect = (self.held_pos[:, 2] < self.fixed_pos[:, 2] + self.disassembly_dists).cpu().numpy()
  env_ids = np.argwhere(if_intersect == 0).reshape(-1)
  self._randomize_gripper_pose(self.cfg_task.move_gripper_sim_steps, env_ids)
```

Scripted policy - _lift_gripper

```
def _lift_gripper(self, lift_distance, sim_steps, env_ids=None):
  """Lift gripper by specified distance. Called outside RL loop (i.e., after last step of
  episode)."""

  ctrl_tgt_pos = torch.empty_like(self.fingertip_midpoint_pos).copy_(self.fingertip_midpoint_pos)
  ctrl_tgt_quat = torch.empty_like(self.fingertip_midpoint_quat).copy_(self.fingertip_midpoint_quat)
  ctrl_tgt_pos[:, 2] += lift_distance
  lifted = self.fixed_pos[:, 2] > self.disassembly_dists * 0.95
  ctrl_tgt_pos[lifted, :2] += torch.randn((self.num_envs, 2), dtype=torch.float32,
device=self.device) * 0.02
  if len(env_ids) == 0:
    env_ids = np.array(range(self.num_envs)).reshape(-1)

  self._move_gripper_to_eef_pose(env_ids, ctrl_tgt_pos, ctrl_tgt_quat, sim_steps, if_log=True)
```

Scripted policy - _randomize_gripper_pose

```
def _randomize_gripper_pose(self, sim_steps, env_ids):
  """Move gripper to random pose."""

  ctrl_tgt_pos = torch.empty_like(self.gripper_goal_pos).copy_(self.gripper_goal_pos)
  ctrl_tgt_pos[:, 2] += self.cfg_task.gripper_rand_z_offset

  # ctrl_tgt_pos = torch.empty_like(self.fingertip_midpoint_pos).copy_(self.fingertip_midpoint_pos)

  fingertip_centered_pos_noise = 2 * (
    torch.rand((self.num_envs, 3), dtype=torch.float32, device=self.device) - 0.5
  ) # [-1, 1]
  pos_noise = torch.tensor([
    0.03, 0.03, 0.02 + self.disassembly_dists[0]
  ], device=self.device)
  fingertip_centered_pos_noise = fingertip_centered_pos_noise @ torch.diag(pos_noise)
  fingertip_centered_pos_noise[:, 2] -= (0.02 + self.disassembly_dists[0]) / 2.0
  ctrl_tgt_pos += fingertip_centered_pos_noise

  # Set target rot
  ctrl_target_fingertip_centered_euler = (
     torch.tensor(self.cfg_task.fingertip_centered_rot_initial, device=self.device)
    .unsqueeze(0)
    .repeat(self.num_envs, 1)
  )

  fingertip_centered_rot_noise = 2 * (
    torch.rand((self.num_envs, 3), dtype=torch.float32, device=self.device) - 0.5
  ) # [-1, 1]
  fingertip_centered_rot_noise = fingertip_centered_rot_noise @ torch.diag(
     torch.tensor(self.cfg_task.gripper_rand_rot_noise, device=self.device)
  )
  ctrl_target_fingertip_centered_euler += fingertip_centered_rot_noise
  ctrl_tgt_quat = torch_utils.quat_from_euler_xyz(
     ctrl_target_fingertip_centered_euler[:, 0],
     ctrl_target_fingertip_centered_euler[:, 1],
     ctrl_target_fingertip_centered_euler[:, 2],
  )

  self._move_gripper_to_eef_pose(env_ids, ctrl_tgt_pos, ctrl_tgt_quat, sim_steps, if_log=True)
```

Fig. 21: Scripted policy for AutoMate disassembly tasks.