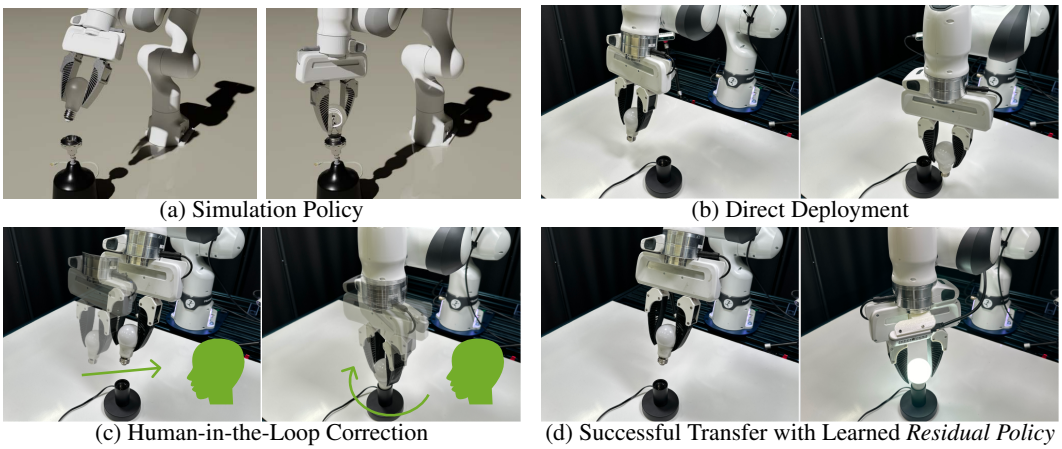
TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction

Yunfan Jiang[1], **Chen Wang**[1], **Ruohan Zhang**[1,2], **Jiajun Wu**[1,2], **Li Fei-Fei**[1,2]
[1]Department of Computer Science [2]Institute for Human-Centered AI (HAI)
Stanford University

Abstract: Learning in simulation and transferring the learned policy to the real world has the potential to enable generalist robots. The key challenge of this approach is to address simulation-to-reality (sim-to-real) gaps. Previous methods often require domain-specific knowledge *a priori*. We argue that a straightforward way to obtain such knowledge is by asking humans to observe and assist robot policy execution in the real world. The robots can then learn from humans to close various sim-to-real gaps. We propose TRANSIC, a data-driven approach to enable successful sim-to-real transfer based on a human-in-the-loop framework. TRANSIC allows humans to augment simulation policies to overcome various unmodeled sim-to-real gaps holistically through intervention and online correction. Residual policies can be learned from human corrections and integrated with simulation policies for autonomous execution. We show that our approach can achieve successful sim-to-real transfer in complex and contact-rich manipulation tasks such as furniture assembly. Through synergistic integration of policies learned in simulation and from humans, TRANSIC is effective as a holistic approach to addressing various, often coexisting sim-to-real gaps. It displays attractive properties such as scaling with human effort. Videos and code are available at `transic-robot.github.io`.

Keywords: Sim-to-Real Transfer, Human-in-the-Loop, Robot Manipulation



(a) Simulation Policy (b) Direct Deployment

(c) Human-in-the-Loop Correction (d) Successful Transfer with Learned *Residual Policy*

Figure 1: **TRANSIC for sim-to-real transfer in contact-rich robotic manipulation tasks. a) and b)** Naïvely deploying policies trained in simulation usually fails due to various sim-to-real gaps. Here, the robot attempts to first align the light bulb with the base and then insert and screw the light bulb into the base. **c)** A human operator monitors robot behaviors, intervenes, and provides online correction through teleoperation when necessary. Human data are collected to train a *residual policy* to tackle various sim-to-real gaps in a holistic manner. **d)** The simulation and the residual policies are integrated together during test time to achieve a successful sim-to-real transfer for contact-rich tasks, such as screwing a light bulb into the base.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

1 Introduction

Learning in simulation is a potential approach to the realization of generalist robots capable of solving sophisticated decision-making tasks [1, 2]. Learning to solve these tasks requires a large amount of training data [3–5]. Providing unlimited training supervision through state-of-the-art simulation [6–10] could alleviate the burden of collecting data in the real world with physical robots [11, 12]. Therefore, it is crucial to seamlessly transfer and deploy robot control policies acquired in simulation, usually through reinforcement learning (RL), to real-world hardware. Successful demonstrations of this simulation-to-reality (sim-to-real) approach have been shown in dexterous in-hand manipulation [13–17], locomotion [18–27], and quadrotor flight [28, 29].

Nevertheless, replicating similar success in manipulation tasks with robotic arms remains surprisingly challenging, with only a few cases in simple non-prehensile manipulation [30–33], industry assembly under restricted settings [34–38], and peg swinging [39]. The difficulty mainly stems from the unavoidable sim-to-real gaps [40], including but not limited to perception gap [41–43], embodiment mismatch [18, 44, 45], controller inaccuracy [46–48], and dynamics realism [49]. Traditionally, researchers tackle them through system identification [18, 30, 50, 51], domain randomization [13, 52–55], real-world adaptation [56, 57], and simulator augmentation [58–60]. Many of these approaches require explicit, domain-specific knowledge and expertise in tasks or simulators. Although for a particular simulation-reality pair, there may exist specific inductive biases that can be hand-crafted *post hoc* to close the sim-to-real gap [18], this knowledge is often not available *a priori*. Identifying its effects on task completion is also intractable.

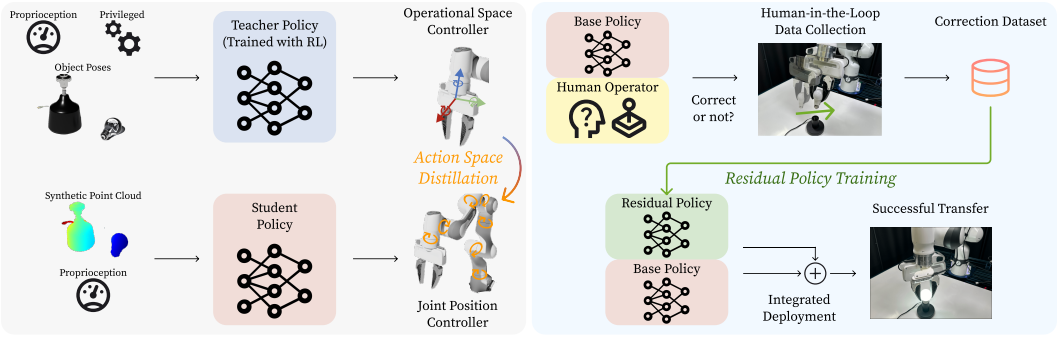
We argue that a straightforward and feasible way for humans to obtain such knowledge is to observe and assist policy execution in the real world. If humans can assist the robot to successfully accomplish the tasks in the real world, sim-to-real gaps are effectively addressed. This naturally leads to a generally applicable paradigm that can cover different priors across simulations and realities—human-in-the-loop learning [61–63] and shared-autonomy [64, 65].

Our key insight is that the human-in-the-loop framework is promising for addressing the sim-to-real gaps as a whole, in which humans directly assist the physical robots during policy execution by providing online correction signals. The knowledge required to close sim-to-real gaps can be learned from human signals. We present TRANSIC (**tran**sferring policies **si**m-to-real by learning from online **c**orrection, Fig. 1), a data-driven approach to enable the successful transfer of robot manipulation policies trained with RL in simulation to the real world. In TRANSIC, once the base robot policies are acquired from simulation training, they are deployed to real robots where human operators monitor the execution. When the robot makes mistakes or gets stuck, humans interrupt and assist robot policies through teleoperation. Such human intervention data are collected to train a *residual policy*, after which the base policy and the residual policy are combined to solve contact-rich manipulation tasks. Unlike previous approaches that heavily rely on domain knowledge, since humans can successfully assist the robot trained in silico to complete real-world tasks, sim-to-real gaps are implicitly handled and addressed by humans in a domain-agnostic manner.

To summarize, our key contribution is a **novel, holistic human-in-the-loop method** called TRANSIC to tackle sim-to-real policy transfer for manipulation tasks. Through extensive evaluation, we show that our method leads to **more effective sim-to-real transfer** compared to traditional methods [50, 52] and **requires less real-robot data** compared to the prevalent imitation learning and offline RL algorithms [66–69]. We demonstrate that successful sim-to-real transfer of short-horizon skills can solve **long-horizon, contact-rich manipulation** tasks, such as furniture assembly.

2 Sim-to-Real Policy Transfer by Learning from Online Correction

An overview of TRANSIC is shown in Fig. 2. This section starts with a brief preliminary review, followed by a description of simulation training. We then introduce residual policies learned from human intervention and online correction and present an integrated framework for deployment during testing. Lastly, we provide implementation details.



(a) Simulation Policy Training through *Action Space Distillation* (b) *Residual Policy* Learning from Human Correction

Figure 2: **TRANSIC method overview. a)** Base policies are first trained in simulation through *action space distillation* with demonstrations generated by RL teacher policies. Base policies take point cloud as input to reduce perception gap. **b)** The acquired base policies are first deployed with a human operator monitoring the execution. The human intervenes and corrects through teleoperation when necessary. Such correction data are collected to learn *residual policies*. Finally, both residual policies and base policies are integrated during test time to achieve a successful transfer.

2.1 Preliminaries

We formulate a robot manipulation task as an infinite-horizon discrete-time Markov Decision Process (MDP) $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. At time step t, a robot observes $s_t \in \mathcal{S}$, executes an action a_t, and receives a scalar reward r_t from the reward function $R(s_t, a_t)$. The environment progresses to the next state following the transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$. The robot learns a parameterized policy $\pi_\theta(\cdot|s)$ to maximize the expected discounted return $\mathcal{J} := \mathbb{E}_{\tau\sim p_{\pi_\theta}}[\sum_{t=0}^{\infty}\gamma^t r_t]$ over induced trajectory distribution $\tau := (s_0, a_0, r_0, ...) \sim p_{\pi_\theta}$, where $s_0 \sim \rho_0$ is sampled from the initial state distribution and $\gamma \in [0, 1)$ is a discount factor. We consider simulation and real environments as two different MDPs. We adopt an intervention-based learning framework [66, 67] where a human operator can intervene and take control during the execution of the robot policy.

2.2 Learning Base Policies in Simulation with RL

Policy Learning with 3D Representation Object geometry matters for contact-rich manipulation. For example, a robot should ideally insert a light bulb into the lamp base with the thread facing down. To retain such 3D information and facilitate sim-to-real transfer, we propose to use point cloud as the main visual modality. Typical RGB observation used in visuomotor policy training [70] suffers from several drawbacks that hinder successful transfer, such as vulnerability to different camera poses [71] and discrepancies between synthetic and real images [42]. Well-calibrated point cloud observation can bypass these issues and has been successfully demonstrated [14, 72]. During the simulation RL training phase, we synthesize point cloud observations for higher throughput. Concretely, given the synthetic point cloud of the m-th object $\mathbf{P}^{(m)} \in \mathbb{R}^{K\times 3}$, we transform it into the global frame through $\mathbf{P}_g^{(m)} = \mathbf{P}^{(m)}\left(\mathbf{R}^{(m)}\right)^\intercal + \left(\mathbf{p}^{(m)}\right)^\intercal$. Here, $\mathbf{R}^{(m)} \in \mathbb{R}^{3\times 3}$ and $\mathbf{p}^{(m)} \in \mathbb{R}^{3\times 1}$ denote the object's orientation and translation in the global frame. Further, the point cloud representation of a scene $\mathbf{S}$ with M objects is aggregated as $\mathbf{P}^{\mathbf{S}} = \bigcup_{m=1}^{M}\mathbf{P}_g^{(m)}$ and subsequently used as policy input.

Action Space Distillation A suitable action abstraction is critical for efficient learning [46, 47] as well as sim-to-real transfer [48]. A high-level controller such as the operational space controller (OSC) [73] facilitates RL exploration [46] but may hinder sim-to-real transfer because it requires accurate modeling of robot parameters, such as joint friction, mass, and inertia [74]; on the other hand, a low-level action space such as the joint position ensures consistent deployment in simulation and real hardware, but renders trial-and-error RL impractical. We draw inspiration from the teacher-student framework [15, 75–77] and propose to first train the teacher policy $\pi^{teacher}$ through RL with OSC and then distill successful trajectories into the student policy $\pi^{student}$ with joint position

control. Specifically, we roll out $\pi^{teacher}$ and record the robot's joint position at every simulated time interval δt to construct a dataset $\mathcal{D}^{teacher} = \{\tau^{(n)}\}_{n=1}^{N}$. We then relabel actions from the end-effector's poses to joint positions. Such a relabeled dataset is ready to train student policies through behavior cloning. We name this approach as *action space distillation* and find it crucial to overcome the sim-to-real controller gap. Furthermore, teacher policies directly receive privileged observations for ease of learning, while student policies learn on synthetic point-cloud inputs to match real-world measurements. The student policy parameterized by θ, $\pi_\theta^{student}$, is trained by minimizing the loss function: $\mathcal{L}^{student} = -\mathbb{E}_{\mathcal{D}^{teacher}}\left[\log \pi_\theta^{student}\right] + \beta \mathbb{E}_{\mathcal{D}^{pcd}}\left[\|\phi(\mathbf{P}^{real}) - \phi(\mathbf{P}^{sim})\|^2\right]$, where $\phi(\cdot)$ denotes the point cloud encoder of $\pi_\theta^{student}$ and $\mathcal{D}^{pcd} = \{(\mathbf{P}^{real}, \mathbf{P}^{sim})^{(i)}\}_{i=1}^{N}$ is a separate dataset that contains N pairs of matched point clouds in simulation and reality for regularization purpose. Further justifications of this distillation phase can be found in Appendix D.1.

2.3 Learning Residual Policies from Online Correction

Human-in-the-Loop Data Collection Once the student policy is obtained from simulation, it is used directly as the base policy π^B to bootstrap the data collection. Naïvely deploying the base policy on real robots usually results in inferior performance and unsafe motion due to various sim-to-real gaps. In TRANSIC, the base policy is instead deployed in a way that is fully synchronized with the human operator. Concretely, at time step t, once $a_t^B \sim \pi^B$ is deployed, a human operator needs to decide whether intervention is necessary, indicated as $\mathbb{1}_t^H$. Intervention is not necessary for most task execution when the robot is approaching objects of interest. However, when the robot tends to behave abnormally, the human operator intervenes and takes full control through teleoperation to correct robot errors. In these cases, the robot's pre- and post-intervention states, as well as intervention indicator $\mathbb{1}_t^H$, are collected to construct the online correction dataset $\mathcal{D}^H \leftarrow \mathcal{D}^H \cup \left(\mathbb{1}_t^H, \mathbf{q}_t^{pre}, \mathbf{q}_t^{post}\right)$. This procedure is illustrated in Appendix Algorithm 1.

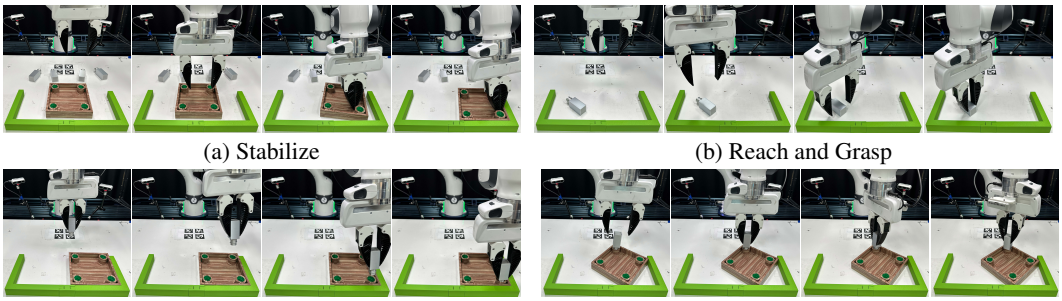
Human Correction as Residual Policies Properly modeling human correction can be challenging. This is because humans usually solve tasks not purely based on current observation, hence the non-Markovian decision process [68]. Therefore, directly fine-tuning the base policy π^B on human correction dataset $\mathcal{D}^H$ leads to large motions and even model collapse (Sec. 3). Inspired by prior work on learning residuals to compensate for unknown dynamics and noisy observations [78–80], we propose to incorporate human correction behaviors with *residual policies*. Concretely, at the time of intervention, a residual policy π_ψ^R parameterized by ψ learns to predict human intervention as the difference between post- and pre-intervention robot states: $a^R = \mathbf{q}^{post} \ominus \mathbf{q}^{pre}$, where $\ominus$ denotes generalized subtraction. For continuous variables such as joint position, it computes the numerical difference; for binary variables such as opening and closing the gripper, it computes exclusive nor. The residual policy is then trained to maximize the likelihood of human correction: $\mathcal{L}^{residual} = -\mathbb{E}_{\mathcal{D}^H}\left[\log \pi_\psi^R(a^R|\cdot)\right]$.

2.4 An Integrated Deployment Framework

In practice, we find that learning a gating function to control whether to apply residual actions or not leads to better empirical performance (Appendix D.3). We call this *learned gated residual policy*. Denote the gating function as $g_\psi(\cdot)$. It shares the same feature encoder with the residual policy π^R and is jointly learned through classification on the same correction dataset $\mathcal{D}^H$. At the inference time, we effectively use g_ψ as an indicator function $\mathbb{1}_g$ to determine whether to apply the predicted residual actions. The policy effectively being deployed to autonomously complete tasks is an integration of base policy π^B and residual policy π^R, gated by g: $\pi^{deployed} = \pi^B \oplus \mathbb{1}_g \pi^R$. A joint position controller is used during deployment.

2.5 Implementation Details

We use the Isaac Gym [9] simulator. Proximal policy optimization (PPO) [81] is used to train teacher policies from scratch with task-specific reward functions and curricula. Student policies



(a) Stabilize (b) Reach and Grasp

(c) Insert (d) Screw

Figure 3: **Four tasks benchmarked in this work.** They are fundamental skills required to assemble a square table from FurnitureBench [85]. The task definition can be found in Appendix C.1.

are parameterized as Gaussian Mixture Models (GMMs) [68]. See Appendix A for more details. During human-in-the-loop data collection, we use a 3Dconnexion SpaceMouse as the teleoperation interface. Residual policies use state-of-the-art point cloud encoders [82–84] and GMM as the action head. More training hyperparameters are provided in Appendix B.4.

3 Experiments

We answer the following research questions through experiments.

- $\mathcal{Q}\mathbf{1}$: Does TRANSIC lead to better transfer performance while requiring less real-world data?
- $\mathcal{Q}\mathbf{2}$: How effectively can TRANSIC address different types of sim-to-real gaps?
- $\mathcal{Q}\mathbf{3}$: How does TRANSIC scale with human effort?
- $\mathcal{Q}\mathbf{4}$: Does TRANSIC exhibit intriguing properties, such as generalization to unseen objects, policy robustness, ability to solve long-horizon tasks, and other emergent behaviors?

3.1 Tasks, Baselines, and Evaluation Protocol

As shown in Fig. 3, we consider complex contact-rich manipulation tasks (*Stabilize*, *Reach and Grasp*, *Insert*, and *Screw*) that require high precision in FurnitureBench [85]. These tasks are challenging and ideal for testing sim-to-real transfer, since perception, embodiment, controller, and dynamics gaps all need to be addressed. We collect 20, 100, 90, and 17 real-robot trajectories with human correction, respectively. These amount to 62, 434, 489, and 58 corrections for each task. See Appendix B.1 for the detailed system setup. We compare with three groups of baselines. **1) Traditional sim-to-real methods:** It includes domain randomization and data augmentation [52] ("DR. & Data Aug."), real-world fine-tuning through BC ("BC Fine-Tune") and implicit Q-learning [69] ("IQL Fine-Tune"). To estimate the performance lower bound, we also include "Direct Transfer" without any data augmentation or real-world fine-tuning. **2) Interactive imitation learning (IL):** It includes HG-Dagger [66] and IWR [67]. **3) Learning from real-robot data only:** It includes BC [86], BC-RNN [68], and IQL [69] that are trained on real-robot demonstrations only. *All* evaluations are conducted on the real robot and consist of 20 trials starting with different objects and robot poses. See Appendix C for details.

3.2 Results

TRANSIC is effective for sim-to-real transfer and requires significantly less real-world data ($\mathcal{Q}\mathbf{1}$). As shown in Fig. 4 and Table A.XI, TRANSIC achieves the best performance on average and in all four tasks with significant margins. What are the reasons for successful transfer? We observe that adding real-world human correction data does not guarantee improvement. For example, among traditional sim-to-real methods, the best baseline BC Fine-Tune outperforms DR. & Data Aug. by 7%, but IQL Fine-Tune leads to worse performance. In contrast, TRANSIC effectively uses human correction data, which boosts average performance by 1.24×. Not only does it achieve

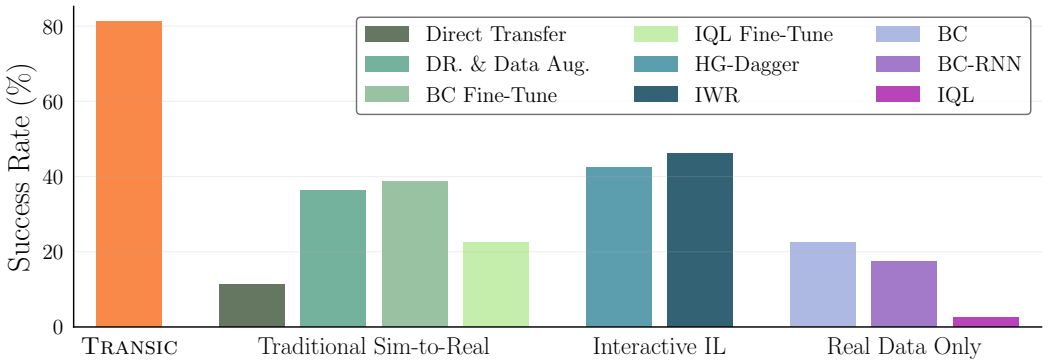


Figure 4: **Average success rates over four benchmarked tasks.** Numerical results in Table A.XI.

the best transfer performance, but it also improves simulation policies the most among various sim-to-real approaches.

Furthermore, TRANSIC outperforms interactive IL methods, including HG-Dagger and IWR, by 0.75× on average. Although both of them weigh the intervention data higher during training, we find that they tend to *erase* the original policy and lead to catastrophic forgetting. In contrast, by incorporating human correction with a separate residual policy and integrating both base and residual policies through gating, TRANSIC combines the best properties of both policies during deployment. It relies on the simulation policy for robust execution most of the time; when the base policy is likely to fail, it automatically applies the residual policy to prevent failures and correct mistakes.

Finally, TRANSIC only requires dozens of real-robot corrections to achieve superior performance. However, methods such as BC-RNN and IQL trained on such a limited number of trajectories suffer from overfitting and model collapse. TRANSIC achieves 3.6× better performance than them. This result highlights the importance of first training in simulation and then leveraging sim-to-real transfer for robot learning practitioners.

In summary, we show that in sim-to-real transfer, a good base policy learned from the simulation can be combined with limited real-world data to achieve success. However, effectively utilizing human correction data to address the sim-to-real gap is challenging, especially when we want to prevent catastrophic forgetting of the base policy.

TRANSIC is effective in addressing different sim-to-real gaps (Q2). We shed light on its ability to close each individual sim-to-real gap by creating five different simulation-reality pairs. For each of them, we intentionally create large gaps between the simulation and the real world. These gaps are applied to real-world settings, including *perception error*, *underactuated controller*, *embodiment mismatch*, *dynamics difference*, and *object asset mismatch*. Note that these are *artificial* settings for a controlled study. See Appendix C.3 for detailed setups.

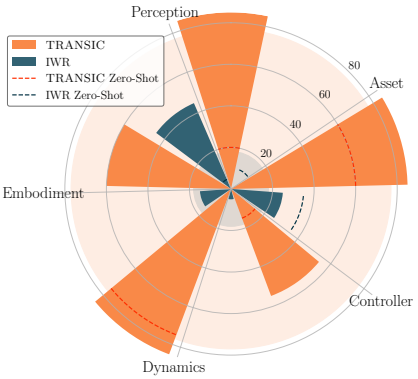


Figure 5: **Robustness to different sim-to-real gaps.** Numbers are averaged success rates (%). Polar bars represent performances after training with data collected specifically to address a particular gap. Dashed lines are zero-shot performances. Shaded circles show average performances.

As shown in Fig. 5, TRANSIC achieves an average success rate of 77% across five different simulation-reality pairs with deliberately exacerbated sim-to-real gaps. This indicates its remarkable ability to close these individual gaps. In contrast, the best baseline method, IWR, only achieves an average success rate of 18%. We attribute this effectiveness in addressing different sim-to-real gaps to the residual policy design. Zeng et al. [80] echos our finding that residual learning is an effective tool to compensate for domain factors that cannot be explicitly modeled. Furthermore, training with data specifically collected from a particular setting generally increases TRANSIC's performance. However, this is not the case for IWR, where fine-

tuning on new data can even lead to worse performance. These results show that TRANSIC is better not only in addressing multiple sim-to-real gaps as a whole but also in handling individual gaps of a very different nature.

TRANSIC scales with human effort ($\mathcal{Q}3$). We demonstrate that TRANSIC scales better with human data than the best baseline, IWR, as shown in Fig. 6 and Table A.XII. If we increase the dataset size from 25% to 75% of the full size, TRANSIC improves on average by 42%. In contrast, IWR only achieves a 23% relative improvement. Additionally, for tasks other than *Insert*, IWR performance plateaus at an early stage and even starts to decrease as more human data becomes available. We hypothesize that IWR suffers from catastrophic forgetting and struggles to properly model the behavioral modes of humans and trained robots. On the other hand, TRANSIC bypasses these issues by learning gated residual policies only from human correction.

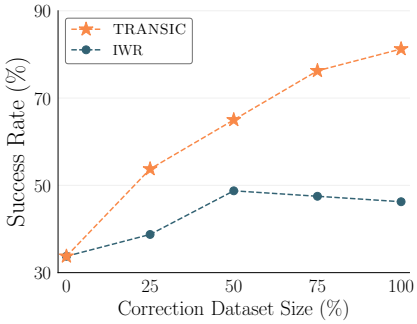


Figure 6: **Scalability with human correction data.** Per-task results are shown in Table A.XII.

Intriguing properties and emergent behaviors of TRANSIC ($\mathcal{Q}4$). As shown in Fig. 7, TRANSIC can achieve an average success rate of 75% when zero-shot evaluated on assembling a lamp. However, IWR can only succeed once every three attempts. This evidence suggests that TRANSIC is not overfitting to a particular object; instead, it has learned reusable skills for **category-level object generalization**. Further, with the ablation results shown in Table 1, TRANSIC exhibits intriguing properties including **effective gating**, **policy robustness** against reduced cameras and suboptimal correction data, and **consistency in learned visual features**. See Appendix C.5 for detailed setups and discussions. Qualitatively, TRANSIC shows several representative behaviors that resemble humans. For instance, they include error recovery, unsticking, safety-aware actions, and failure prevention, as shown in Fig. A.12. Finally, we demonstrate that successful sim-to-real transfer of individual skills can be effectively chained together to enable long-horizon contact-rich manipulation (Fig. 8). See videos at `transic-robot.github.io`.

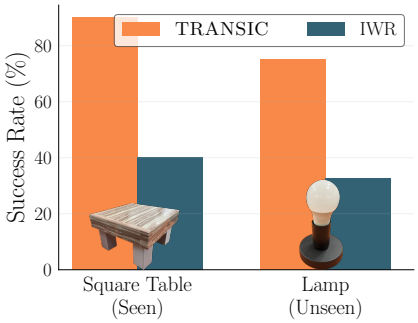


Figure 7: **Generalization to unseen objects from a new category.** Success rates are averaged over tasks *Reach and Grasp* and *Screw*.

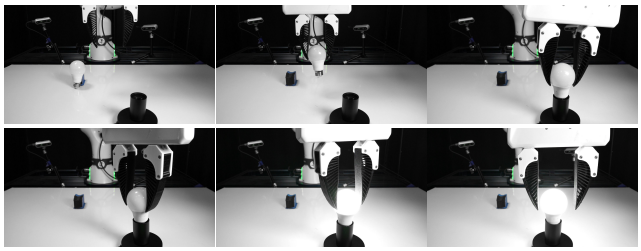
4 Related Work

Robot Learning via Sim-to-Real Transfer Physics-based simulations have become a driving force for developing robotic skills [6–10]. However, the domain gap between the simulators and reality is not negligible [40]. Successful sim-to-real transfer includes locomotion [18–27], dexterous in-hand manipulation [13–17], and simple non-prehensile manipulation [30–39]. In this work, we tackle more challenging sim-to-real transfer for complex whole-arm manipulation tasks and successfully demonstrate that our approach can solve sophisticated contact-rich tasks. More importantly, it requires significantly fewer real-robot data compared to the behavior cloning approach [68]. This makes solutions based on simulators and sim-to-real transfer more appealing to roboticists.

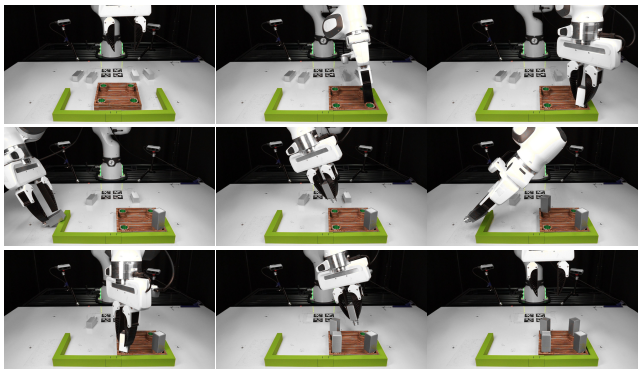
Table 1: **Ablation study results.** Numbers are success rates.

	Average	Stabilize	Reach and Grasp	Insert	Screw
Original	81%	100%	95%	45%	85%
w/ Human Gating	82%	100%	95%	50%	85%
Reduced Cameras	80%	95%	90%	40%	95%
Noisy Correction	76%	85%	80%	45%	95%
w/o Regularization	55%	85%	55%	20%	60%

Sim-to-Real Gaps in Manipulation Tasks The sim-to-real gaps can be coarsely categorized as follows: **a)** perception gap [41–43], where synthetic sensory observations differ from those measured in the real world; **b)** embodiment mismatch [18, 44, 45], where the robot models used in simulation do not match the real-world hardware precisely; **c)** controller inaccuracy [46–48], meaning that the results of deploying the same commands differ in simulation and real hardware; and **d)** poor physical realism [49], where physical interactions such as contact and collision are poorly simulated [87]. Traditional methods to address them include system identification [18, 30, 50, 51], domain randomization [13, 52–55], real-world adaptation [56], and simulator augmentation [58–60]. However, system identification is mostly engineered on a case-by-case basis. Domain randomization suffers from the inability to identify and randomize all physical parameters. Methods with real-world adaptation, usually through meta-learning [88], incur potential safety concerns during the adaptation phase. In contrast, our method leverages human intervention data to implicitly overcome the gap in a domain-agnostic way, leading to a safer deployment.



(a) Assemble a lamp (160 seconds in 1× speed).



(b) Assemble a square table (550 seconds in 1× speed).

Figure 8: **a)** The robot assembles a lamp. **b)** The robot assembles a square table from FurnitureBench [85]. Videos are available at `transic-robot.github.io`.

Human-in-The-Loop Robot Learning Human-in-the-loop machine learning is a prevalent framework to inject human knowledge into autonomous systems [62, 89, 90]. The recent trend focuses on continually improving robots' capability with human feedback [91] and autonomously generating corrective intervention data [92]. Our work further extends this trend by showing that sim-to-real gaps can be effectively eliminated by using human intervention and correction signals. In shared autonomy, robots and humans share the control authority to achieve a common goal [64, 65, 93–95]. This control paradigm has been largely studied in assistive robotics and human-robot collaboration [96–98]. In this work, we provide a novel perspective by employing it in the sim-to-real transfer of robot control policies and demonstrating its importance in attaining effective transfer.

5 Limitations and Conclusion

Limitations 1) Current tasks are in a single-arm tabletop scenario. Though TRANSIC can potentially be applied to more complicated robots with recent teleoperation interfaces [99–103]. 2) Human operators still manually decide when to intervene. This could be automated using failure detection techniques [104, 105]. 3) TRANSIC requires simulation policies with reasonable performance. Nevertheless, it is compatible with recent advances in synthesizing manipulation data [106, 107].

In this work, we present TRANSIC, a human-in-the-loop method for sim-to-real transfer in contact-rich manipulation tasks. We show that combining a strong base policy from simulation with limited real-world data can be effective. However, utilizing human correction data without causing catastrophic forgetting of the base policy is challenging. TRANSIC overcomes this by learning a gated residual policy from a small amount of human correction data. We show that TRANSIC effectively addresses various sim-to-real gaps, both collectively and individually, and scales with human effort.

Acknowledgments

We are grateful to Josiah Wong, Chengshu (Eric) Li, Weiyu Liu, Wenlong Huang, Stephen Tian, Sanjana Srivastava, and the SVL PAIR group for their helpful feedback and insightful discussions. This work is in part supported by the Stanford Institute for Human-Centered AI (HAI), ONR MURI N00014-22-1-2740, ONR MURI N00014-21-1-2801, and Schmidt Sciences. Ruohan Zhang is partially supported by Wu Tsai Human Performance Alliance Fellowship.

References

[1] H. Choi, C. Crump, C. Duriez, A. Elmquist, G. Hager, D. Han, F. Hearl, J. Hodgins, A. Jain, F. Leve, C. Li, F. Meier, D. Negrut, L. Righetti, A. Rodriguez, J. Tan, and J. Trinkle. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1):e1907856118, 2021. doi:10.1073/pnas.1907856118. URL https://www.pnas.org/doi/abs/10.1073/pnas.1907856118.

[2] C. K. Liu and D. Negrut. The role of physics-based simulators in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):35–58, 2021. doi:10.1146/annurev-control-072220-093055. URL https://doi.org/10.1146/annurev-control-072220-093055.

[3] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi:10.1145/1015330.1015430. URL https://doi.org/10.1145/1015330.1015430.

[4] H. D. III, J. Langford, and D. Marcu. Search-based structured prediction. *arXiv preprint arXiv: Arxiv-0907.0786*, 2009.

[5] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv: Arxiv-2303.04129*, 2023.

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

[7] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[8] 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. *arXiv preprint arXiv: Arxiv-2009.12293*, 2020.

[9] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv: Arxiv-2108.10470*, 2021.

[10] C. Li, C. Gokmen, G. Levine, R. Martín-Martín, S. Srivastava, C. Wang, J. Wong, R. Zhang, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei. BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=_8DoIe8G3t.

[11] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv: Arxiv-2212.06817*, 2022.

[12] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, A. Laurens, C. Fantacci, V. Dalibard, M. Zambelli, M. Martins, R. Pevceviciute, M. Blokzijl, M. Denil, N. Batchelor, T. Lampe, E. Parisotto, K. Żołna, S. Reed, S. G. Colmenarejo, J. Scholz, A. Abdolmaleki, O. Groth, J.-B. Regli, O. Sushkov, T. Rothörl, J. E. Chen, Y. Aytar, D. Barker, J. Ortiz, M. Riedmiller, J. T. Springenberg, R. Hadsell, F. Nori, and N. Heess. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv: Arxiv-2306.11706*, 2023.

[13] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arXiv: Arxiv-1910.07113*, 2019.

[14] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. doi:10.1126/scirobotics.adc9244. URL https://www.science.org/doi/abs/10.1126/scirobotics.adc9244.

[15] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv: Arxiv-2309.00987*, 2023.

[16] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. *arXiv preprint arXiv: Arxiv-2210.04887*, 2022.

[17] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. *arXiv preprint arXiv: Arxiv-2309.09979*, 2023.

[18] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv: Arxiv-1804.10332*, 2018.

[19] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: rapid motor adaptation for legged robots. In D. A. Shell, M. Toussaint, and M. A. Hsieh, editors, *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi:10.15607/RSS.2021.XVII.011. URL https://doi.org/10.15607/RSS.2021.XVII.011.

[20] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv: Arxiv-2309.05665*, 2023.

[21] R. Yang, G. Yang, and X. Wang. Neural volumetric memory for visual locomotion control. *arXiv preprint arXiv: Arxiv-2304.01201*, 2023.

[22] H. Benbrahim and J. A. Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22(3):283–302, 1997. ISSN 0921-8890. doi:https://doi.org/10.1016/S0921-8890(97)00043-2. URL https://www.sciencedirect.com/science/article/pii/S0921889097000432. Robot Learning: The New Wave.

[23] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid. Reinforcement learning-based cascade motion policy design for robust 3d bipedal locomotion. *IEEE Access*, 10:20135–20148, 2022. doi:10.1109/ACCESS.2022.3151771.

[24] L. Krishna, G. A. Castillo, U. A. Mishra, A. Hereid, and S. Kolathaya. Linear policies are sufficient to realize robust bipedal walking on challenging terrains. *arXiv preprint arXiv: Arxiv-2109.12665*, 2021.

[25] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv: Arxiv-2105.08328*, 2021.

[26] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *arXiv preprint arXiv: Arxiv-2303.03381*, 2023.

[27] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *arXiv preprint arXiv: Arxiv-2401.16889*, 2024.

[28] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 2023. doi:10.1038/s41586-023-06419-4. URL https://doi.org/10.1038/s41586-023-06419-4.

[29] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82): eadg1462, 2023. doi:10.1126/scirobotics.adg1462. URL https://www.science.org/doi/abs/10.1126/scirobotics.adg1462.

[30] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg. Planar robot casting with real2sim2real self-supervised learning. *arXiv preprint arXiv: Arxiv-2111.04814*, 2021.

[31] W. Zhou and D. Held. Learning to grasp the ungraspable with emergent extrinsic dexterity. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 150–160. PMLR, 2022. URL https://proceedings.mlr.press/v205/zhou23a.html.

[32] M. Kim, J. Han, J. Kim, and B. Kim. Pre- and post-contact policy decomposition for non-prehensile manipulation with zero-shot sim-to-real transfer. *arXiv preprint arXiv: Arxiv-2309.02754*, 2023.

[33] X. Zhang, S. Jain, B. Huang, M. Tomizuka, and D. Romeres. Learning generalizable pivoting skills. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5865–5871. IEEE, 2023. doi:10.1109/ICRA48891.2023.10161271. URL https://doi.org/10.1109/ICRA48891.2023.10161271.

[34] S. Kozlovsky, E. Newman, and M. Zacksenhouse. Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices. *IEEE Robotics and Automation Letters*, 7(4):10898–10905, 2022. doi:10.1109/LRA.2022.3191070.

[35] D. Son, H. Yang, and D. Lee. Sim-to-real transfer of bolting tasks with tight tolerance. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9056–9063, 2020. doi:10.1109/IROS45743.2020.9341644.

[36] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. S. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.039. URL https://doi.org/10.15607/RSS.2023.XIX.039.

[37] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka. Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=gFXVysXh48K.

[38] X. Zhang, M. Tomizuka, and H. Li. Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion. *arXiv preprint arXiv: Arxiv-2311.07499*, 2023.

[39] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *arXiv preprint arXiv: Arxiv-1810.05687*, 2018.

[40] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Advances in Artificial Life*, pages 704–720, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49286-3.

[41] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *ArXiv*, abs/1511.07111, 2015. URL https://api.semanticscholar.org/CorpusID:1541419.

[42] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv: Arxiv-1709.07857*, 2017.

[43] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer. *arXiv preprint arXiv: Arxiv-2011.03148*, 2020.

[44] S. Koos, J.-B. Mouret, and S. Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, page 119–126, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300728. doi:10.1145/1830483.1830505. URL https://doi.org/10.1145/1830483.1830505.

[45] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. *arXiv preprint arXiv: Arxiv-1803.05580*, 2018.

[46] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. *arXiv preprint arXiv: Arxiv-1906.08880*, 2019.

[47] J. Wong, V. Makoviychuk, A. Anandkumar, and Y. Zhu. Oscar: Data-driven operational space control for adaptive and robust robot manipulation. *arXiv preprint arXiv: Arxiv-2110.00704*, 2021.

[48] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt. On the role of the action space in robot manipulation learning and sim-to-real transfer. *arXiv preprint arXiv: Arxiv-2312.03673*, 2023.

[49] Y. S. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, Á. Moravánszky, G. State, M. Lu, A. Handa, and D. Fox. Factory: Fast contact for robotic assembly. In K. Hauser, D. A. Shell, and S. Huang, editors, *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*, 2022. doi:10.15607/RSS.2022.XVIII.035. URL https://doi.org/10.15607/RSS.2022.XVIII.035.

[50] L. Ljung. *System Identification*, pages 163–173. Birkhäuser Boston, Boston, MA, 1998. ISBN 978-1-4612-1768-8. doi:10.1007/978-1-4612-1768-8_11. URL https://doi.org/10.1007/978-1-4612-1768-8_11.

[51] P. Chang and T. Padir. Sim2real2sim: Bridging the gap between simulation and real-world in flexible object manipulation. *arXiv preprint arXiv: Arxiv-2002.02538*, 2020.

[52] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv: Arxiv-1710.06537*, 2017.

[53] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. V. Wyk, A. Zhurkevich, B. Sundaralingam, and Y. S. Narang. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5977–5984. IEEE, 2023. doi:10.1109/ICRA48891.2023.10160216. URL https://doi.org/10.1109/ICRA48891.2023.10160216.

[54] J. Wang, Y. Qin, K. Kuang, Y. Korkmaz, A. Gurumoorthy, H. Su, and X. Wang. Cyberdemo: Augmenting simulated human demonstration for real-world dexterous manipulation. *arXiv preprint arXiv: Arxiv-2402.14795*, 2024.

[55] T. Dai, J. Wong, Y. Jiang, C. Wang, C. Gokmen, R. Zhang, J. Wu, and L. Fei-Fei. ACDC: Automated creation of digital cousins for robust policy learning. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=7c5rAY8oU3.

[56] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow. Meta-reinforcement learning for robotic industrial insertion tasks. *arXiv preprint arXiv: Arxiv-2004.14404*, 2020.

[57] Y. Zhang, L. Ke, A. Deshpande, A. Gupta, and S. Srinivasa. Cherry-Picking with Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.021.

[58] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 8973–8979. IEEE, 2019. doi:10.1109/ICRA.2019.8793789. URL https://doi.org/10.1109/ICRA.2019.8793789.

[59] J. P. Hanna and P. Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4931–4932. AAAI Press, 2017.

[60] E. Heiden, D. Millard, E. Coumans, and G. S. Sukhatme. Augmenting differentiable simulators with neural networks to close the sim2real gap. *arXiv preprint arXiv: Arxiv-2007.06045*, 2020.

[61] C. A. Cruz and T. Igarashi. A survey on interactive reinforcement learning: Design principles and open challenges. *arXiv preprint arXiv: Arxiv-2105.12949*, 2021.

[62] Y. Cui, P. Koppol, H. Admoni, S. Niekum, R. Simmons, A. Steinfeld, and T. Fitzgerald. Understanding the relationship between interactions and outcomes in human-in-the-loop machine learning. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4382–4391. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi:10.24963/ijcai.2021/599. URL https://doi.org/10.24963/ijcai.2021/599. Survey Track.

[63] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone. Leveraging human guidance for deep reinforcement learning tasks. *arXiv preprint arXiv: Arxiv-1909.09906*, 2019.

[64] S. Javdani, S. S. Srinivasa, and J. A. Bagnell. Shared autonomy via hindsight optimization. *arXiv preprint arXiv: Arxiv-1503.07619*, 2015.

[65] S. Reddy, A. D. Dragan, and S. Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv: Arxiv-1802.01744*, 2018.

[66] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. *arXiv preprint arXiv: Arxiv-1810.02890*, 2018.

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

[68] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv: Arxiv-2108.03298*, 2021.

[69] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv: Arxiv-2110.06169*, 2021.

[70] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv: Arxiv-1802.09564*, 2018.

[71] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *arXiv preprint arXiv: Arxiv-2307.03659*, 2023.

[72] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. *arXiv preprint arXiv: Arxiv-2211.09423*, 2022.

[73] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. doi:10.1109/JRA.1987.1087068.

[74] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27 (6):737–757, 2008. doi:10.1177/0278364908091463. URL https://doi.org/10.1177/0278364908091463.

[75] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv: Arxiv-1511.06295*, 2015.

[76] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *arXiv preprint arXiv: Arxiv-2211.11744*, 2022.

[77] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. In A. Faust, D. Hsu, and G. Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 297–307. PMLR, 2021. URL https://proceedings.mlr.press/v164/chen22a.html.

[78] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. *arXiv preprint arXiv: Arxiv-1812.03201*, 2018.

[79] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv: Arxiv-1812.06298*, 2018.

[80] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *arXiv preprint arXiv: Arxiv-1903.11239*, 2019.

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv: Arxiv-1707.06347*, 2017.

[82] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv: Arxiv-1612.00593*, 2016.

[83] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv: Arxiv-2103.03206*, 2021.

[84] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. *arXiv preprint arXiv: Arxiv-1810.00825*, 2018.

[85] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.041. URL https://doi.org/10.15607/RSS.2023.XIX.041.

[86] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper_files/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[87] P. C. Horak and J. C. Trinkle. On the similarities and differences among contact models in robot simulation. *IEEE Robotics and Automation Letters*, 4(2):493–499, 2019. doi:10.1109/LRA.2019.2891085.

[88] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv: Arxiv-1703.03400*, 2017.

[89] J. A. Fails and D. R. Olsen. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, page 39–45, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581135866. doi:10.1145/604045.604056. URL https://doi.org/10.1145/604045.604056.

[90] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, Dec. 2014. doi:10.1609/aimag.v35i4.2513. URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2513.

[91] 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. *arXiv preprint arXiv: Arxiv-2211.08416*, 2022.

[92] R. Hoque, A. Mandlekar, C. R. Garrett, K. Goldberg, and D. Fox. Interventional data generation for robust and data-efficient robot imitation learning. In *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023. URL https://openreview.net/forum?id=ckFRoOaA3n.

[93] J. Crandall and M. Goodrich. Characterizing efficiency of human robot interaction: a case study of shared-control teleoperation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1290–1295 vol.2, 2002. doi:10.1109/IRDS.2002.1043932.

[94] A. D. Dragan and S. S. Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7):790–805, 2013. doi:10.1177/0278364913490324. URL https://doi.org/10.1177/0278364913490324.

[95] D. Gopinath, S. Jain, and B. D. Argall. Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters*, 2(1):247–254, 2017. doi:10.1109/LRA.2016.2593928.

[96] A. D. Dragan and S. S. Srinivasa. *Formalizing assistive teleoperation*, volume 376. MIT Press, July, 2012.

[97] H. J. Jeon, D. P. Losey, and D. Sadigh. Shared autonomy with learned latent actions. *arXiv preprint arXiv: Arxiv-2005.03210*, 2020.

[98] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh. "no, to the right" – online language corrections for robotic manipulation via shared autonomy. *arXiv preprint arXiv: Arxiv-2301.02555*, 2023.

[99] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv: Arxiv-2309.13037*, 2023.

[100] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv: Arxiv-2403.07788*, 2024.

[101] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv: Arxiv-2401.02117*, 2024.

[102] S. Dass, W. Ai, Y. Jiang, S. Singh, J. Hu, R. Zhang, P. Stone, B. Abbatematteo, and R. Martín-Martín. Telemoma: A modular and versatile teleoperation system for mobile manipulation. *arXiv preprint arXiv: Arxiv-2403.07869*, 2024.

[103] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv: Arxiv-2403.04436*, 2024.

[104] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu. Model-based runtime monitoring with interactive imitation learning. *arXiv preprint arXiv: Arxiv-2310.17552*, 2023.

[105] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv: Arxiv-2306.15724*, 2023.

[106] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv: Arxiv-2310.17596*, 2023.

[107] L. Ankile, A. Simeonov, I. Shenfeld, and P. Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. *arXiv preprint arXiv: Arxiv-2404.03729*, 2024.

[108] M. N. Mistry and L. Righetti. Operational space control of constrained and underactuated systems. In *Robotics: Science and Systems*, 2011. URL `https://api.semanticscholar.org/CorpusID:10392712`.

[109] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv: Arxiv-1511.07289*, 2015.

[110] D. Makoviichuk and V. Makoviychuk. rl-games: A high-performance framework for reinforcement learning. `https://github.com/Denys88/rl_games`, May 2021.

[111] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv: Arxiv-1506.02438*, 2015.

[112] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv: Arxiv-1412.6980*, 2014.

[113] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi:10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[114] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv: Arxiv-1606.08415*, 2016.

[115] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. *6th Annual Conference on Robot Learning*, 2022.

[116] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv: Arxiv-1706.02413*, 2017.

[117] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv: Arxiv-1608.03983*, 2016.

[118] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox. Motion policy networks. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 967–977. PMLR, 2022. URL https://proceedings.mlr.press/v205/fishman23a.html.

[119] 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. *arXiv preprint arXiv: Arxiv-2109.08273*, 2021.

[120] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.026. URL https://doi.org/10.15607/RSS.2023.XIX.026.

[121] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv: Arxiv-1806.06920*, 2018.

[122] R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

[123] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. Sapien: A simulated part-based interactive environment. *arXiv preprint arXiv: Arxiv-2003.08515*, 2020.

[124] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, P. P. Tehrani, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *arXiv preprint arXiv: Arxiv-2301.04195*, 2023.

[125] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, A. Wahid, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv: Arxiv-2010.14406*, 2020.

[126] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. *arXiv preprint arXiv: Arxiv-2109.12098*, 2021.

[127] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv: Arxiv-2210.03094*, 2022.

[128] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv: Arxiv-2209.05451*, 2022.

[129] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv: Arxiv-2011.01975*, 2020.

[130] J. Gu, D. S. Chaplot, H. Su, and J. Malik. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv: Arxiv-2209.02778*, 2022.

[131] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. W. Clegg, J. Turner, Z. Kira, M. Savva, A. Chang, D. S. Chaplot, D. Batra, R. Mottaghi, Y. Bisk, and C. Paxton. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv: Arxiv-2306.11565*, 2023.

[132] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, R. Krishna, D. Schwenk, E. VanderBilt, and A. Kembhavi. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv preprint arXiv: Arxiv-2312.02976*, 2023.

[133] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv: Arxiv-1910.13439*, 2019.

[134] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *arXiv preprint arXiv: Arxiv-2105.03655*, 2021.

[135] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. *arXiv preprint arXiv: Arxiv-2211.09006*, 2022.

[136] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. *arXiv preprint arXiv: Arxiv-2203.17275*, 2022.

[137] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. *arXiv preprint arXiv: Arxiv-2208.01160*, 2022.

[138] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[139] A. Boeing and T. Bräunl. Leveraging multiple simulators for crossing the reality gap. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1113–1119, 2012. doi:10.1109/ICARCV.2012.6485313.

[140] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. doi:10.1126/scirobotics.aau5872. URL `https://www.science.org/doi/abs/10.1126/scirobotics.aau5872`.

[141] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 3080–3087. IEEE, 2019. doi:10.1109/ICRA.2019.8793506. URL `https://doi.org/10.1109/ICRA.2019.8793506`.

[142] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012. ISSN 0022-0000. doi:https://doi.org/10.1016/j.jcss.2011.12.028. URL https://www.sciencedirect.com/science/article/pii/S0022000012000281. JCSS Special Issue: Cloud Computing 2011.

[143] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. *arXiv preprint arXiv: Arxiv-1306.6294*, 2013.

[144] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv: Arxiv-1706.03741*, 2017.

[145] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *arXiv preprint arXiv: Arxiv-2006.14091*, 2020.

[146] K. Lee, L. Smith, and P. Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv: Arxiv-2106.05091*, 2021.

[147] X. Wang, K. Lee, K. Hakhamaneshi, P. Abbeel, and M. Laskin. Skill preferences: Learning to extract and execute robotic skills from human feedback. *arXiv preprint arXiv: Arxiv-2108.05382*, 2021.

[148] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv: Arxiv-2203.02155*, 2022.

[149] V. Myers, E. Bıyık, and D. Sadigh. Active reward learning from online preferences. *arXiv preprint arXiv: Arxiv-2302.13507*, 2023.

[150] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv: Arxiv-2305.18290*, 2023.

[151] J. Hejna, R. Rafailov, H. Sikchi, C. Finn, S. Niekum, W. B. Knox, and D. Sadigh. Contrastive preference learning: Learning from human feedback without rl. *arXiv preprint arXiv: Arxiv-2310.13639*, 2023.

[152] W. B. Knox and P. Stone. Reinforcement learning from human reward: Discounting in episodic tasks. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 878–885, 2012. doi:10.1109/ROMAN.2012.6343862.

[153] B. D. Argall, E. L. Sauser, and A. G. Billard. Tactile guidance for policy refinement and reuse. In *2010 IEEE 9th International Conference on Development and Learning*, pages 7–12, 2010. doi:10.1109/DEVLRN.2010.5578872.

[154] T. Fitzgerald, E. Short, A. Goel, and A. Thomaz. Human-guided trajectory adaptation for tool transfer. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 1350–1358, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

[155] A. V. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan. Learning robot objectives from physical human interaction. In *Conference on Robot Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:28406224.

[156] A. Najar, O. Sigaud, and M. Chetouani. Interactively shaping robot behaviour with unlabeled human instructions. *arXiv preprint arXiv: Arxiv-1902.01670*, 2019.

[157] N. Wilde, E. Bıyık, D. Sadigh, and S. L. Smith. Learning reward functions from scale feedback. *arXiv preprint arXiv: Arxiv-2110.00284*, 2021.

[158] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv: Arxiv-1605.06450*, 2016.

[159] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv: Arxiv-1707.05173*, 2017.

[160] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone. Appli: Adaptive planner parameter learning from interventions. *arXiv preprint arXiv: Arxiv-2011.00400*, 2020.

[161] C. Celemin and J. R. del Solar. An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Systems*, 95:77–97, 2018. doi:10.1007/s10846-018-0839-z. URL http://link.springer.com/article/10.1007/s10846-018-0839-z/fulltext.html.

[162] Z. Peng, W. Mo, C. Duan, Q. Li, and B. Zhou. Learning from active human involvement through proxy value propagation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=q8SukwaEBy.

[163] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, and J. Kober. Interactive imitation learning in robotics: A survey. *arXiv preprint arXiv: Arxiv-2211.00600*, 2022.

A Simulation Training Details

In this section, we provide details about simulation training, including the used simulator backend, task designs, reinforcement learning (RL) training of teacher policy, and student policy distillation.

A.1 The Simulator

We use Isaac Gym Preview 4 [9] as the simulator backend. NVIDIA PhysX[1] is used as the physics engine to provide realistic and precise simulation. Simulation settings are listed in Table A.I. The robot model is from Franka ROS package[2]. We borrow furniture models from FurnitureBench [85] to create various tasks that require complex and contact-rich manipulation.

Table A.I: **Simulation settings.**

Hyperparameter	Value
Simulation Frequency	60 Hz
Control Frequency	60 Hz
Num Substeps	2
Num Position Iterations	8
Num Velocity Iterations	1

A.2 Task Implementations

We implement four tasks based on the furniture model `square_table`: *Stabilize*, *Reach and Grasp*, *Insert*, and *Screw*. An overview of simulated tasks is shown in Fig A.1. We elaborate on their initial conditions, success criteria, reward functions, and other necessary information.

A.2.1 Stabilize

In this task, the robot needs to push the square tabletop to the right corner of the wall such that it is supported and remains stable in following assembly steps. The robot is initialized such that its gripper locates at a neutral position. The tabletop is initialized at the coordinate $(0.54, 0.00)$ relative to the robot base. We then randomly translate it with displacements drawn from $\mathcal{U}(-0.015, 0.015)$ along x and y directions (the distance unit is meter hereafter). We also apply random Z rotation with values drawn from $\mathcal{U}(-15°, 15°)$. Four table legs are initialized in the scene. The task is successful only when the following three conditions are met:

1) The square tabletop contacts the front and right walls;
2) The square tabletop is within a pre-defined region;
3) No table leg is in the pre-defined region.

We use the following reward function:

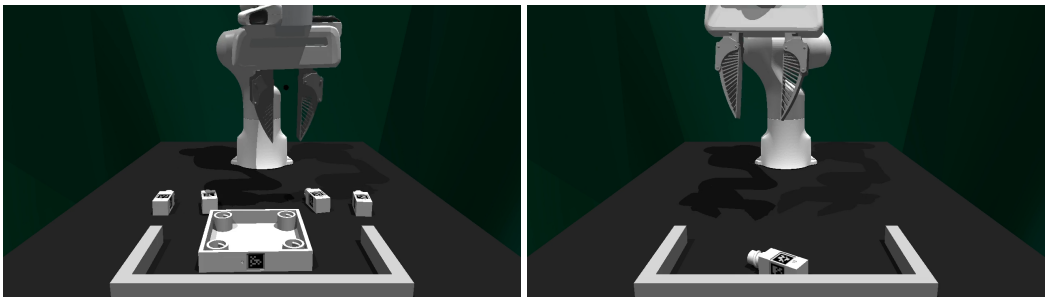
$$r_t = w_{success}\mathbb{1}_{success} - w_{\dot{\mathbf{q}}}\|\dot{\mathbf{q}}_t\| - w_{action}\|a_t\|, \tag{A.1}$$

where $w_{success}$ is the success reward, $\mathbb{1}_{success}$ indicates the success according to aforementioned conditions, $w_{\dot{\mathbf{q}}}$ penalizes large joint velocities, $\dot{\mathbf{q}}_t$ is the joint velocity, w_{action} penalizes large action commands, and a_t represents the action command at time step t. We set $w_{success} = 10$, $w_{\dot{\mathbf{q}}} = 10^{-5}$, and $w_{action} = 10^{-5}$. The episode length is 100. One episode terminates upon success or timeout.

A.2.2 Reach and Grasp

In this task, the robot needs to reach and grasp a table leg that is randomly spawned in the valid workspace region. The task is successful once the robot grasps the table leg and lifts it for a certain

[1] https://developer.nvidia.com/physx-sdk
[2] https://github.com/frankaemika/franka_ros



(a) Stabilize (b) Reach and Grasp

(c) Insert (d) Screw

Figure A.1: **Visualization of simulated tasks.**

height. The object's irregular shape limits certain grasping poses. For example, the end-effector needs to be near orthogonal to the table leg in the xy plane and far away from the screw thread. Therefore, we design a curriculum over the object geometry to warm up the RL learning. It gradually adjusts the object geometry from a cube, to a cuboid, and finally the table leg. In all curriculum stages, the reward function is

$$r_t = w_{distance} d + w_{lifted} \mathbb{1}_{lifted} + w_{success} \mathbb{1}_{success}. \tag{A.2}$$

Here, $w_{distance}$ is the weight for distance reward, w_{lifted} is the reward for the leg being lifted, and $w_{success}$ is the success weight. d is the distance to the table leg and is calculated as

$$d = 1 - \tanh\left(\frac{10}{4}(d_{eef} + d_{left\_finger} + d_{right\_finger} + d_{orthogonal})\right), \tag{A.3}$$

where d_{eef} is the distance between the end-effector and the table leg, d_{left_finger} is the distance between the left gripper tip to the table leg, d_{right_finger} is the distance between the right gripper tip to the table leg, and $d_{orthogonal}$ is the difference between the current and the orthogonal grasping orientations. We set $w_{distance} = 0.1$, $w_{lifted} = 1.0$, and $w_{success} = 200.0$. The episode length is 50. One episode terminates upon success or timeout.

A.2.3 Insert

In this task, the robot needs to insert a pre-grasped table leg into the far right assembly hole of the tabletop, while the tabletop is already stabilized. The tabletop is initialized at the coordinate $(0.53, 0.05)$ relative to the robot base. We then randomly translate it with displacements sampled from $\mathcal{U}(-0.02, 0.02)$ along x and y directions. We also apply random Z rotation with values drawn from $\mathcal{U}(-45°, 45°)$. We further randomize the robot's pose by adding noises sampled from $\mathcal{U}(-0.25, 0.25)$ to joint positions. The task is successful when the table leg remains vertical and is close to the correct assembly position within a small threshold. We design curricula over the randomization strength to facilitate the learning. The following reward function is used:

$$r_t = w_{distance} d + w_{success} \mathbb{1}_{success}, \tag{A.4}$$

where $w_{distance}$ is the weight for distance-based reward, d is the distance between the table leg and target assembly position, $w_{success}$ is the success weight, and $\mathbb{1}_{success}$ indicates task success. The distance d consists of an Euclidean distance $d_{position}$ and an orientation distance $d_{vertical}$ to encourage the robot to keep the table leg vertical.

$$d = 1 - \tanh\left(\frac{10}{2}\left(d_{position} + d_{vertical}\right)\right) \tag{A.5}$$

We set $w_{distance} = 1.0$ and $w_{success} = 100.0$. The episode length is 100. One episode terminates upon success or timeout.

A.2.4 Screw

In this task, the robot is initialized such that its end-effector is close to an inserted table leg. It needs to screw the table leg clockwise into the tabletop. We design curricula over the action space: at the early stage, the robot only controls the end-effector's orientation; at the latter stage, it gradually takes full control. We slightly randomize object and robot poses during initialization. The reward function is

$$r_t = \left(1 - \mathbb{1}_{failure}\right)\left(w_{screw} d_{screw} + w_{success}\mathbb{1}_{success}\right) - w_{deviation} d_{deviation}. \tag{A.6}$$

Here, $\mathbb{1}_{failure}$ indicates the task failure, w_{screw} is the screwing reward weight, d_{screw} measures the screwed angle, $w_{success}$ is the success weight, and $\mathbb{1}_{success}$ indicates the task success. The task is considered as successful when the leg has been screwed 180° into the tabletop. It is considered as failed when the table leg tilts more than 10° from the vertical pose. We set $w_{screw} = 0.1$, $w_{success} = 100.0$, and $w_{deviation} = 10^{-2}$. The episode length is 200. One episode terminates upon success, failure, or timeout.

A.3 Teacher Policy Training

A.3.1 Model Details

Observation Space Besides proprioceptive observations, teacher policies also receive privileged observations to facilitate the learning. They include objects' states (poses and velocities), end-effector's velocity, contact forces, gripper left and right fingers' positions, gripper center position, and joint velocities. Full observations are summarized in Table A.II.

Table A.II: **The observation space for teacher policies.**

Name	Dimension	Name	Dimension
Proprioceptive		Privileged	
Joint Position	7	Objects States	$N_{objects}\times$ 13
Cosine Joint Position	7	End-Effector Velocity	6
Sine Joint Position	7	Contact Forces	$N_{objects}\times$ 3
End-Effector Position	3	Left and Right Fingers' Positions	6
End-Effector Rotation	4	Gripper Center Position	3
Gripper Width	1	Joint Velocity	7

Controller and Action Space An operational space controller (OSC) [73] is used in teacher policy training to improve sample efficiency. We follow Mistry and Righetti [108] to add nullspace control torques to prevent large changes in joint configuration. The action space is thus the change of end-effector's pose. We further add a binary action to control gripper's opening and closing. Formally, it can be expressed as $\mathcal{A}_{teacher} = (\delta x, \delta y, \delta z, \delta r, \delta p, \delta y, \mathbb{1}_{gripper})$, where $(\delta x, \delta y, \delta z) \in \mathbb{R}^3$ is the translation change, $(\delta r, \delta p, \delta y) \in \mathbb{R}^3$ is the rotation change, and $\mathbb{1}_{gripper} \in \{0, 1\}$ is the gripper action.

Model Architecture We use feed-forward policies in RL training. It consists of MLP encoders to encode proprioceptive and privileged vector observations, and unimodal Gaussian distributions as the action head. Model hyperparameters are listed in Table A.III.

Table A.III: **Model hyperparameters for RL teacher policies.**

Hyperparameter	Value	Hyperparameter	Value
Obs. Encoder Hidden Depth	1	Obs. Encoder Activation	ReLU
Obs. Encoder Hidden Dim	256	Action Head Hidden Layers	[256, 128, 64]
Obs. Encoder Output Dim	256	Action Head Activation	ELU [109]

A.3.2 Domain Randomization

We apply domain randomization during RL training to learn more robust teacher policies. Parameters are summarized in Table A.IV.

Table A.IV: **Domain randomization used in RL training.**

Parameter	Type	Distribution	Parameter	Type	Distribution
Robot			**Objects**		
Mass	Scaling	$\mathcal{U}(0.5, 1.5)$	Mass	Scaling	$\mathcal{U}(0.5, 1.5)$
Friction	Scaling	$\mathcal{U}(0.7, 1.3)$	Friction	Scaling	$\mathcal{U}(0.5, 1.5)$
Joint Lower Limit	Scaling	$\log\mathcal{U}(1.00, 1.01)$	Rolling Friction	Scaling	$\mathcal{U}(0.5, 1.5)$
Joint Upper Limit	Scaling	$\log\mathcal{U}(1.00, 1.01)$	Torsion Friction	Scaling	$\mathcal{U}(0.5, 1.5)$
Joint Stiffness	Scaling	$\log\mathcal{U}(1.00, 1.01)$	Restitution	Additive	$\mathcal{U}(0.0, 1.0)$
Joint Damping	Scaling	$\log\mathcal{U}(1.00, 1.01)$	Compliance	Additive	$\mathcal{U}(0.0, 1.0)$

Parameter	Type	Distribution
Simulation		
Gravity	Additive	$\mathcal{U}(0.0, 0.4)$

A.3.3 RL Training Details

We use the model-free RL algorithm Proximal Policy Optimization (PPO) [81] to learn teacher policies. Hyperparameters are listed in Table A.V. We customize the framework from Makoviichuk and Makoviychuk [110] to use as our training framework.

Table A.V: **Hyperparameters used in PPO training.**

Hyperparameter	Value	Hyperparameter	Value
Learning Rate	5×10^{-4}	Critic Weight	4
Discount Factor	0.99	GAE [111] λ	0.95
Entropy Weight	0	PPO ϵ	0.2
Optimizer	Adam [112]	Horizon	32

A.4 Student Policy Distillation

A.4.1 Data Generation

We use trained teacher policies as oracles to generate data for student policies training. Concretely, we roll out each teacher policy to generate $10,000$ successful trajectories for each task. We exclude trajectories that are shorter than 20 steps.

A.4.2 Observation Space

Student policies receive observations that can be obtained in the real world. They are point-cloud and proprioceptive observations. We synthesize point clouds from objects' 6D poses to improve the training throughput. Concretely, given the groundtruth point cloud of the m-th object $\mathbf{P}^{(m)} \in \mathbb{R}^{K\times 3}$, we transform it into the global frame through $\mathbf{P}_g^{(m)} = \mathbf{P}^{(m)} \left(\mathbf{R}^{(m)}\right)^\intercal + \left(\mathbf{p}^{(m)}\right)^\intercal$. Here $\mathbf{R}^{(m)} \in \mathbb{R}^{3\times 3}$ and $\mathbf{p}^{(m)} \in \mathbb{R}^{3\times 1}$ denote the object's orientation and translation in the global frame. Further, the point-cloud representation of a scene $\mathbf{S}$ with M objects is aggregated as $\mathbf{P}^{\mathbf{S}} = \bigcup_{m=1}^{M} \mathbf{P}_g^{(m)}$. For the robot, we only include point clouds for its two fingers and ignore other parts. To facilitate policies to differentiate gripper fingers from the scene, we extend the coordinate dimension to include a semantic label $\in \{0, 1\}$ that indicates gripper fingers or not. This information can be obtained on real robots through forward kinematics. A full point cloud is then downsampled to 768 points. Table A.VI lists the observation space.

Table A.VI: **The observation space for student policies.**

Name	Dimension
Point Cloud	768×4
Proprioceptive	
Joint Position	7
Cosine Joint Position	7
Sine Joint Position	7
End-Effector Position	3
End-Effector Rotation	4
Gripper Width	1

A.4.3 Action Space Distillation

To reduce the controller sim-to-real gap before transfer, we train student policies to output in the configuration space. To achieve that, we relabel actions $\hat{a}$ in trajectories generated by teacher policies from end-effector's delta poses to absolute joint positions. This is equivalent to set $\hat{a}_t = \mathbf{q}_{t+1}$ for all time steps. Therefore, the action space for student policies is $\mathcal{A}_{student} = (\mathbf{q}, \mathbb{1}_{gripper})$, where $\mathbf{q} \in \mathbb{R}^7$ is the joint position within the valid range. In simulation, student policies' actions are deployed with a joint position controller.

A.4.4 Model Architecture

We use feed-forward policies for tasks *Reach and Grasp* and *Insert* and recurrent policies for tasks *Stabilize* and *Screw* as we find they achieve the best distillation results. PointNets [82] are used to encode point clouds. Recall that each point in the point cloud also contains a semantic label indicating the gripper or not. We concatenate point coordinates with these semantic labels' vector embeddings before passing into the PointNet encoder. We use Gaussian Mixture Models (GMM) [68] as the action head. Detailed model hyperparameters are listed in Table A.VII.

A.4.5 Data Augmentation

We apply strong data augmentation during distillation. For point-cloud observations, random translation and random jitter are independently applied with a probability $P_{pcd\_aug} = 0.4$. We also add Gaussian noises to proprioceptive observations. Augmentation parameters are listed in Table A.VIII.

A.4.6 Training Details

To regularize point-cloud features, we separately collect a dataset containing 59 pairs of matched point clouds in simulation and reality. One pair from them is visualized in Fig A.2. Student policies

Table A.VII: **Model hyperparameters for student policies.**

Hyperparameter	Value	Hyperparameter	Value
Point Cloud		RNN	
PointNet Hidden Dim	256	RNN Type	LSTM [113]
PointNet Hidden Depth	2	RNN Num Layers	2
PointNet Output Dim	256	RNN Hidden Dim	512
PointNet Activation	GELU [114]	RNN Horizon	5
Gripper Semantic Embd Dim	128	GMM Action Head	
Feature Fusion		Hidden Dim	128
MLP Hidden Dim	512	Hidden Depth	3
MLP Hidden Depth	1	Num Modes	5
MLP Activation	ReLU	Activation	ReLU

Table A.VIII: **Data augmentation used in distillation.**

Hyperparameter	Value	Hyperparameter	Value
Point Cloud		Proprioception	
Augmentation Probability	0.4	Prop. Noise Distribution	$\mathcal{N}(0, 0.1)$
Random Translation Distribution	$\mathcal{U}(-0.04, 0.04)$	Prop. Noise Low	-0.3
Random Jittering Ratio	0.1	Prop. Noise High	0.3
Random Jittering Distribution	$\mathcal{N}(0, 0.01)$		
Random Jittering Low	-0.015		
Random Jittering High	0.015		

are trained by minimizing the loss in Sec. 2.2, where we set $\beta = 10^{-3}$. We use the Adam optimizer [112] with a learning rate of 10^{-4} during training. We periodically roll out student policies in simulation for $1,000$ episodes. We then select the checkpoint that corresponds to the highest success rate to use as the base policy in the real-world learning stage.

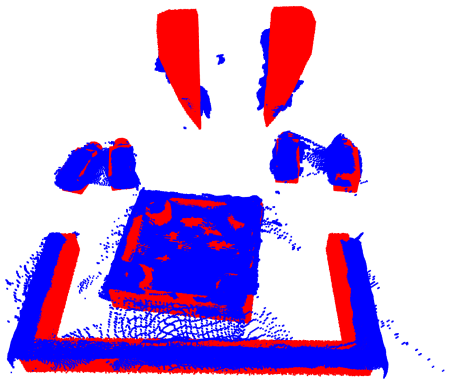


Figure A.2: **Visualization of paired point clouds in simulation (red) and reality (blue).**

B Real-World Learning Details

In this section, we provide details about real-world learning, including the hardware setup, human-in-the-loop data collection, and residual policy training.

B.1 Hardware Setup

As shown in Fig. A.3, our system consists of a Franka Emika 3 robot mounted on the tabletop. We use four fixed cameras and one wrist camera for point cloud reconstruction. They are three RealSense D435 and two RealSense D415. There is also a 3d-printed three-sided wall glued on top of the table to provide external support. We use a joint position controller from the Deoxys library [115] to control our robot at 1000 Hz.

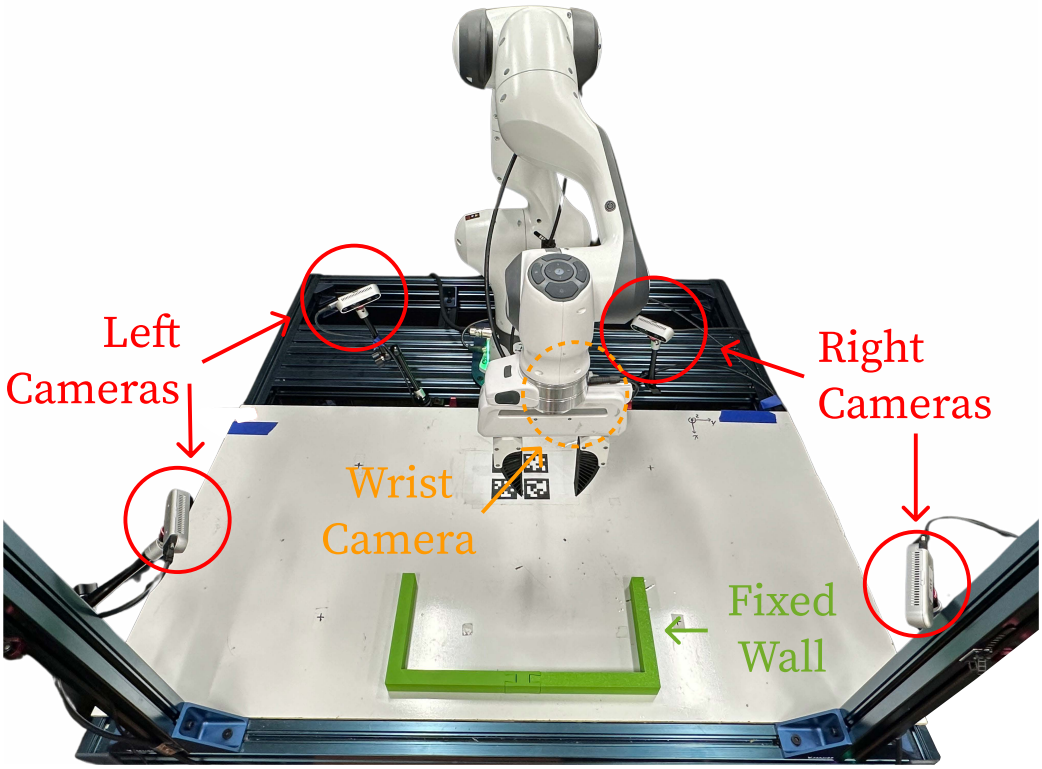


Figure A.3: **System setup.** Our system consists of a Franka Emika 3 robot mounted on the tabletop, four fixed cameras and one wrist camera (positioned at the rear side of the end-effector) for point cloud reconstruction, and a 3d-printed three-sided wall glued onto tabletop to provide external support.

B.2 Obtaining Point Clouds from Multi-View Cameras

We use multi-view cameras for point cloud reconstruction to avoid occlusions. Specifically, we first calibrate all cameras to obtain their poses in the robot base frame. We then transform captured point clouds in camera frames to the robot base frame and concatenate them together. We further perform cropping based on coordinates and remove statistical and radius outliers. To identify points belonging to the gripper so that we can add gripper semantic labels (Sec. A.4.2), we compute poses for two gripper fingers through forward kinematics. We then remove measured points corresponding to gripper fingers through K-nearest neighbor, given fingers' poses and synthetic point clouds. Subsequently, we add semantic labels to points belonging to the scene and synthetic gripper's point clouds. Finally, we uniformly down-sample without replacement. We opt to not use farthest point sampling [116] due to its slow speed. One example is shown in Fig. A.4.

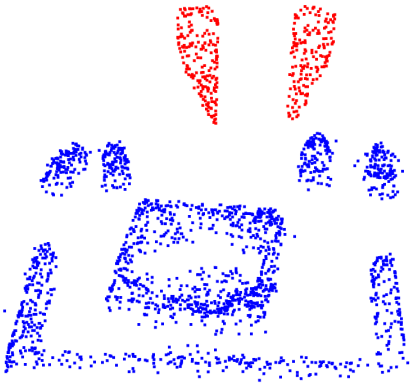


Figure A.4: **Visualization of real-world point-cloud observations.** We obtain them by 1) cropping point clouds fused from multi-view cameras based on coordinates, 2) removing statistical and radius outliers, 3) removing points corresponding to gripper fingers and replacing with synthetic point clouds through forward kinematics, 4) uniformly sampling without replacement, and 5) appending semantic labels to indicate gripper fingers (red) and the scene (blue).

B.3 Human-in-the-Loop Data Collection

This data collection procedure is illustrated in Algorithm 1. As shown in Fig. A.5, we use a 3Dconnexion SpaceMouse as the teleoperation device. We design a specific UI to facilitate the synchronized data collection. Here, the human operator will be asked to intervene or not. The operator answers through keyboard. If the operator does not intervene, the base policy's next action will be deployed. If the operator decides to intervene, the SpaceMouse is then activated to teleoperate the robot. After the correction, the operator can exit the intervention mode by pressing one button on the SpaceMouse. We use this system and interface to collect 20, 100, 90, and 17 trajectories with correction for tasks *Stabilize*, *Reach and Grasp*, *Insert*, and *Screw*, respectively. We use 90% of them as training data and the remaining as held-out validation sets. We visualize the cumulative distribution function of human correction in Fig. A.6.

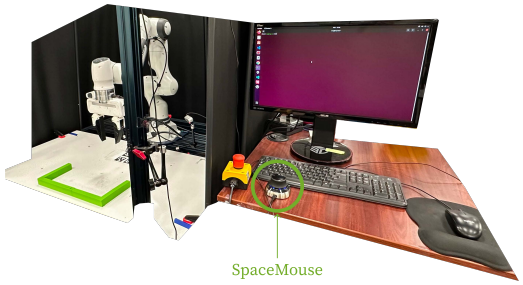


Figure A.5: **Real workspace setup for human-in-the-loop data collection.** The human operator provides online correction through a 3Dconnexion SpaceMouse while monitoring the robot's execution.

B.4 Residual Policy Training

B.4.1 Model Architecture

The residual policy takes the same observations as the base policy (Table A.VI). Furthermore, to effectively predict residual actions, it is also conditioned on base policy's outputs. Its action head outputs eight-dim vectors, while the first seven dimensions correspond to residual joint positions and the last dimension determines whether to negate base policy's gripper action or not. Besides, a separate intervention head predicts whether the residual action should be applied or not (learned gated residual policy, Sec. 2.4).

Algorithm 1: Human Intervention and Online Correction Data Collection

input : Base policy π^B, human policy π^H, real-world environment $\mathcal{E}$
output : Human correction dataset $\mathcal{D}^H$
initialize: $\mathcal{D}^H \leftarrow \emptyset$

$o \leftarrow \mathcal{E}.reset()$
while *not* $\mathcal{E}.terminated$ **do**
  ▷ deploy the base policy
  $a^B \leftarrow a^B \sim \pi^B(o)$
  $o^{next} \leftarrow \mathcal{E}.deploy(a^B)$
  ▷ human decides intervention or not
  $\mathbb{1}^H \leftarrow \pi^H.intervene(o, o^{next})$
  if $\mathbb{1}^H$ **then**
    $\mathbf{q}^{pre} \leftarrow \mathcal{E}.robot\_state$
    ▷ deploy human correction
    $a^H \leftarrow a^H \sim \pi^H(o, o^{next})$
    $o^{next} \leftarrow \mathcal{E}.deploy(a^H)$
    $\mathbf{q}^{post} \leftarrow \mathcal{E}.robot\_state$
    ▷ update dataset
    $\mathcal{D}^H \leftarrow \mathcal{D}^H \cup \left(\mathbf{q}^{pre}, \mathbf{q}^{post}, \mathbb{1}^H, o\right)$
  end
  ▷ update the next observation
  $o \leftarrow o^{next}$
end

For tasks *Stabilize* and *Insert*, we use a PointNet [82] as the point-cloud encoder. For tasks *Reach and Grasp* and *Screw*, we use a Perceiver [83, 84] as the point-cloud encoder. Residual policies are instantiated as feed-forward policies in all tasks. We use GMM as the action head and a simple two-way classifier as the intervention head. Model hyperparameters are summarized in Table A.IX.

Table A.IX: **Model hyperparameters for residual policies.**

Hyperparameter	Value	Hyperparameter	Value
PointNet		Feature Fusion	
PointNet Hidden Dim	256	MLP Hidden Dim	512
PointNet Hidden Depth	2	MLP Hidden Depth	1
PointNet Output Dim	256	MLP Activation	ReLU
PointNet Activation	GELU	GMM Action Head	
Gripper Semantic Embd Dim	128	Hidden Dim	128
Perceiver		Hidden Depth	3
Perceiver Hidden Dim	256	Num Modes	5
Perceiver Number of Heads	8	Activation	ReLU
Perceiver Number of Queries	8	Intervention Head	
Gripper Semantic Embd Dim	128	Hidden Dim	128
Base Policy Action Conditioning		Hidden Depth	3
Base Policy Gripper Action Embd Dim	64	Activation	ReLU

B.4.2 Training Details

To train the learned gated residual policy, we first only learn the feature encoder and the action head. We then freeze the entire model and only learn the intervention head. We opt for this two-stage

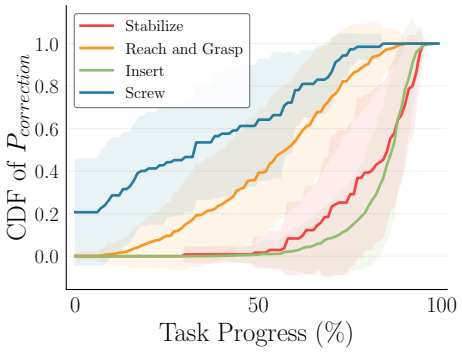


Figure A.6: **Cumulative distribution function (CDF) of human correction.** Shaded regions represent standard deviation. Human correction happens at different times across tasks. This fact necessitates TRANSIC's learned gating mechanism.

training since we find that training both action and intervention heads at the same time will result in sub-optimal residual action prediction. We follow the best practice for policy training, including using learning rate warm-up and cosine annealing [117]. Training hyperparameters are listed in Table A.X.

Table A.X: **Hyperparameters used in residual policy training.**

Hyperparameter	Value
Learning Rate	10^{-4}
Weight Decay	0
Learning Rate Warm Up Steps	$1,000$
Learning Rate Cosine Decay Steps	$100,000$
Minimal Learning Rate	10^{-6}
Optimizer	Adam

C Experiment Settings and Evaluation Details

In this section, we provide details about our experiment settings and evaluation protocols.

C.1 Task Definition

As shown in Fig. 3, we quantitatively benchmark four tasks. They are fundamental skills required to assemble a square table from FurnitureBench [85]. We randomize objects' initial poses during evaluation.

- *Stabilize*: The robot pushes the square tabletop to the right corner of the wall such that it remains stable in following assembly steps.
- *Reach and Grasp*: The robot reaches and grasps the table leg. It needs to properly adjust the end effector's orientation to avoid infeasible grasping poses.
- *Insert*: The robot inserts the pre-grasped table leg to the far right assembly hole of the tabletop.
- *Screw*: The robot's end-effector is initialized close to an inserted table leg and it screws the table leg clockwise into the tabletop.

C.2 Main Experiments

We evaluate all methods on four tasks for 20 trials. Each trail starts with different objects and robot poses. We make our best efforts to ensure the same initial settings when evaluating different methods. Specifically, we take pictures for these 20 different initial configurations and refer to them when resetting a new trial. See Figs. A.14, A.15, A.16, A.17 for initial configurations of tasks *Stabilize*, *Reach and Grasp*, *Insert*, and *Screw*, respectively. We follow Liu et al. [91] to label reward for IQL. Full numerical results are provided in Table A.XI.

Table A.XI: **Success rates per tasks.** TRANSIC outperforms all baseline methods in all four tasks.

Tasks	TRANSIC	Direct Transfer	DR. & Data Aug. [52]	BC Fine-Tune	IQL Fine-Tune	HG-Dagger [66]	IWR [67]	BC [86]	BC-RNN [68]	IQL [69]
Stabilize	100%	10%	35%	55%	0%	65%	65%	40%	40%	5%
Reach and Grasp	95%	35%	60%	35%	0%	30%	40%	25%	0%	5%
Insert	45%	0%	15%	15%	25%	35%	40%	10%	5%	0%
Screw	85%	0%	35%	50%	65%	40%	40%	15%	25%	0%

C.3 Experiments with Different Sim-to-Real Gaps

C.3.1 Experiment Setup

We explain how different sim-to-real gaps are created.

Perception Error This is done by applying random jitter to 25% points from point clouds with probability $P = 0.6$, as visualized in Fig. A.7. We test this sim-to-real gap on the task *Reach and Grasp*. The jittering noise is sampled independently from the distribution $\mathcal{N}(0, 0.03)$. We clip the noise to be within the $\pm$ 0.03 range.

(a) (b)

Figure A.7: **Visualization of introduced perception error. a)** The original point-cloud observation. **b)** The erroneous point-cloud observation with random jitter.

Underactuated Controller This is done by making the joint position controller less accurate. We test this gap on the task *Insert*. We emulate an underactuated controller through early stopping. Concretely, at every time a new joint position goal $\mathbf{q}_{goal}$ is set, we record the distance to the goal in configuration space $d_{\mathbf{q}} = \|\mathbf{q} - \mathbf{q}_{goal}\|$ and sample a factor $\Gamma \sim \mathcal{U}(0.80, 0.95)$. The controller will stop reaching the desired goal once it achieves Γ progress, i.e., stop early when $\|\mathbf{q} - \mathbf{q}_{goal}\| \leq (1 - \Gamma)d_{\mathbf{q}}$. Fig. A.8 visualizes the effect.

Embodiment Mismatch This is done by changing the robot gripper to be shorter length as demonstrated in Fig. A.9. We test this gap on the task *Screw*. We notice that the 9 cm length difference incurs a significant gap.

Dynamics Difference This is done by changing object surfaces and increasing friction. We test this gap on the task *Stabilize*. Concretely, we attach friction tapes to the square tabletop's surface to increase friction, hence change the dynamics (Fig. A.10).

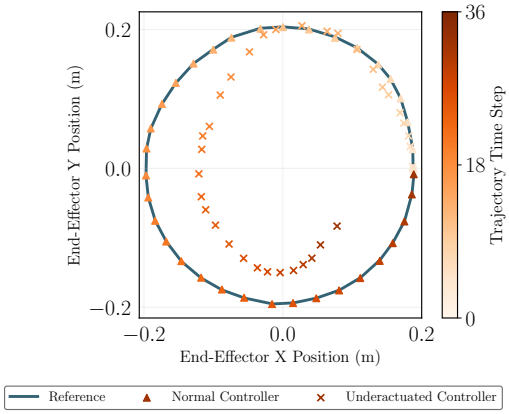


Figure A.8: **Visualization of the trajectory realized by an underactuated controller.** The plot displays the end-effector's position in the XY plane. It shows a reference circular movement, a trajectory tracked by the normal controller, and a trajectory tracked by the underactuated controller.

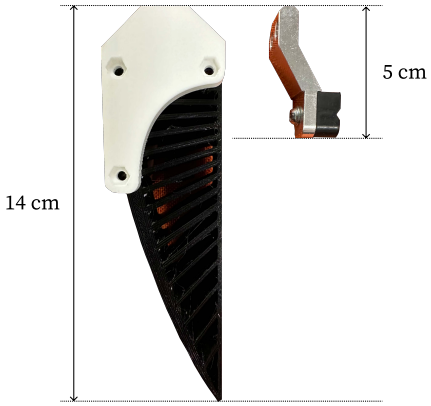


Figure A.9: **Two different gripper fingers used to create embodiment mismatch.** Policies are trained with the longer finger and tested on the shorter finger.

Object Assert Mismatch As shown in Fig. A.11, this is done by replacing the table leg with a light bulb. We test this gap on the task *Reach and Grasp*.

C.3.2 Evaluation

We conduct 20 trails with different initial configurations. Initial conditions for first four experiments are the same as main experiments (Figs. A.14, A.15, A.16, A.17). Fig. A.18 shows initial configurations for the experiment *Object Asset Mismatch*.

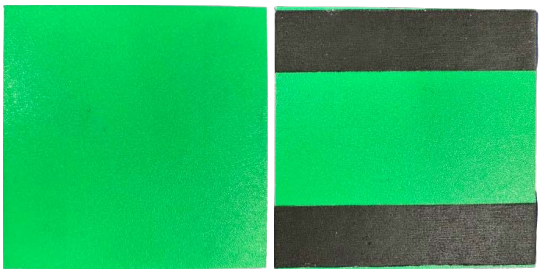
C.4 Data Scalability Experiments

In Table A.XII, we show quantitative results for scalability with human correction dataset size on four tasks.

C.5 Ablation Studies

C.5.1 Effects of Different Gating Mechanisms

We introduce the learned gated residual policy in Sec. 2.4 where the gating mechanism controls when to apply residual actions. To assess the quality of learned gating, we compare its performance with an actual human operator performing gating. Results are shown in Table 1 (row "w/ Human



(a) (b)

Figure A.10: **Two square tabletops used to create dynamics difference. a)** The original surface is smooth. **b)** We attach friction tapes to change the dynamics.

(a) (b)

Figure A.11: **Two objects used to create asset mismatch. a)** Policies are trained with the table leg. **b)** We test policies with an unseen light bulb.

Gating"). It is evident that the learned gating mechanism only incurs negligible performance drops compared to human gating. This suggests that TRANSIC can reliably operate in a fully autonomous setting once the gating mechanism is learned.

C.5.2 Policy Robustness

We investigate the policy robustness against 1) point cloud observations with inferior quality by removing two cameras, and 2) suboptimal correction data with noise injection. We remove two cameras and only keep three. Note that this is the same number of cameras as in FurnitureBench [85]. For tasks other than *Insert*, we keep the wrist camera, the right front camera, and the left rear camera. For the task *Insert*, we keep two front cameras and the left rear camera. We simulate suboptimal correction data by injecting noise into residual actions a^R. This noise is of large magnitude, which follows the normal distribution with zero mean and standard deviation corresponding to 5% of the largest residual action in the dataset. Results are shown in Table 1 (rows "Reduced Cameras" and "Noisy Correction"). We highlight that TRANSIC is robust to partial point cloud inputs caused by the reduced number of cameras. We attribute this to the heavy point cloud downsampling employed during training. Fishman et al. [118] echos our finding that policies trained with downsampled synthetic point cloud inputs can generalize to partial point cloud observations obtained in the real world without the need for shape completion. Meanwhile, when the correction data used to learn residual policies are suboptimal, TRANSIC only shows a relative decrease of 6% in the average success rate. We attribute this to the advantage of our integrated deployment—when the residual policy behaves suboptimally, the base policy could still compensate for the error in subsequent steps.

Table A.XII: **Quantitative results for scalability with human correction dataset size on four tasks.**

Method	Correction Dataset Size (%)				
	0	25	50	75	100
Stabilize					
TRANSIC	35%	80%	80%	100%	100%
IWR [67]		70%	75%	80%	65%
Reach and Grasp					
TRANSIC	60%	65%	80%	90%	95%
IWR [67]		60%	65%	40%	40%
Insert					
TRANSIC	5%	20%	35%	40%	45%
IWR [67]		5%	15%	30%	40%
Screw					
TRANSIC	35%	50%	65%	75%	85%
IWR [67]		20%	40%	40%	40%

C.5.3 Consistency in Learned Visual Features

To learn consistent visual features between the simulation and reality, we propose to regularize the point cloud encoder during the distillation stage. As shown in Table 1 (row "w/o Regularization"), the performance significantly decreases without such regularization, especially for tasks that require fine-grained visual features. Without it, simulation policies would overfit to synthetic point-cloud observations and hence are not ideal for sim-to-real transfer.

C.6 Qualitative Analysis and Emergent Behaviors

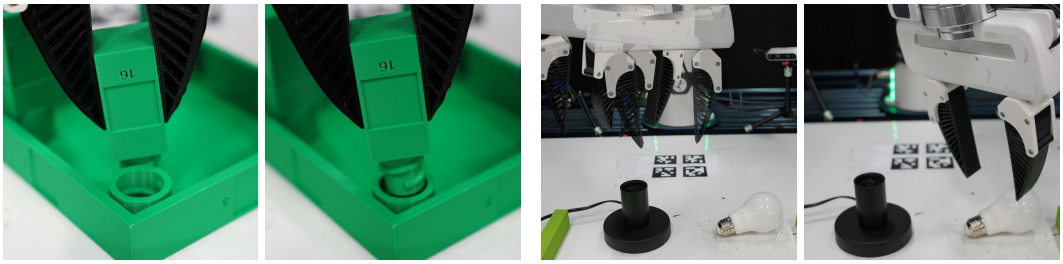
We examine the distribution of the collected human correction dataset. During the human-in-the-loop data collection, the probability of intervening and correcting is reasonably low ($P_{\text{correction}} \approx 0.20$). This is consistent with our intuition that, with a good base policy, interventions are not necessary for most of the time. However, they become critical when the robot tends to behave abnormally due to unaddressed sim-to-real gaps. Moreover, as highlighted in Fig. A.6, interventions happen at different times across tasks. This fact renders heuristics-based methods [119] for deciding when to intervene difficult, and further necessitates our learned residual policy. Several representative behaviors learned by TRANSIC are demonstrated in Fig. A.12.

D Additional Experiment Results and Discussions

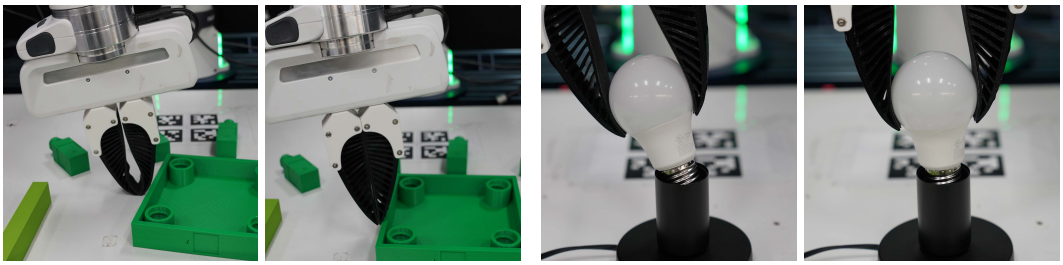
D.1 Empirical Justifications for *Action Space Distillation*

Reasons for the proposed *action space distillation* are twofold. The first is mainly because an OSC is hard to sim-to-real transfer, while a joint position controller can be seamlessly transferred. As suggested in Nakanishi et al. [74], an OSC requires accurate modeling of robot parameters, such as the task-space inertia matrix and gravity compensation. System identification helps but is insufficient. Furthermore, it is often the case that given the same joint torque, the end-effector moves differently in simulation and the real world. Because an OSC uses a task-space error to compute joint torques, this will lead to large joint position deviation.

The second is for better training efficiency. As shown in Fig. A.13, it is almost impossible to directly train RL with point-cloud inputs and joint position action space. Even after 7-day training, RL still



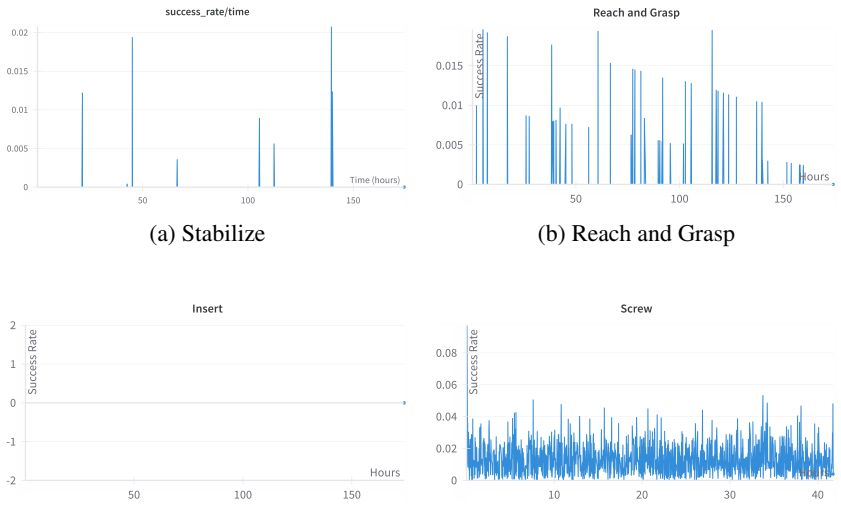
(a) Error Recovery (b) Unsticking



(c) Safety-aware Actions (d) Failure Prevention

Figure A.12: **Emergent behaviors learned by TRANSIC. a) Error recovery.** Left: The robot tries to insert the table leg but the direction is wrong; Right: TRANSIC raises the end effector and moves to the correct insertion position. **b) Unsticking.** Left: The robot hovers for a while and never reaches the light bulb; Right: TRANSIC helps the robot get unstuck and move to the bulb. **c) Safety-aware actions.** Left: When pushing the tabletop, the gripper is too low and bends. This might damage the robot; Right: TRANSIC compensates for the command that causes the end effector to move too low. **d) Failure prevention.** Left: The light bulb will fall and break after gripper opening; Right: TRANSIC adjusts the bulb to a stable pose to prevent failure.

shows no sign of improvement. In contrast, TRANSIC takes around 3 days to train on NVIDIA GeForce RTX 3090 GPUs. Therefore, the distillation is important to make the training feasible.



(a) Stabilize (b) Reach and Grasp

(c) Insert (d) Screw

Figure A.13: **Learning curves for RL with point-cloud observations and joint position actions.**

D.2 Distilling Simulation Base Policy with Diffusion Policy

We experiment with learning simulation base policies (Sec. 2.2) with the Diffusion Policy [120]. Concretely, when performing *action space distillation* to learn student policies, we replace the Gaussian Mixture Model (GMM) action head with the Diffusion Policy. Proper data augmentation (Table A.VIII) is also applied to robustify learned policies. Hyperparameters are provided in Table A.XIII.

Table A.XIII: **Diffusion Policy hyperparameters.**

Hyperparameter	Value	Hyperparameter	Value
Architecture	UNet	T_o	2
UNet Hidden Dims	[64, 128]	T_a	8
UNet Kernel Size	5	T_p	16
UNet GroupNorm Num Groups	8	Num Denoising Steps (Train)	100
Diffusion Step Embd Dim	128	Num Denoising Steps (Eval)	16

The comparison between GMMs on the real robot is shown in Table. A.XIV. We highlight two findings. First, the significant domain difference between simulation and reality generally exists regardless of different policy parameterizations. Second, since the Diffusion Policy plans and executes a future trajectory, it is more vulnerable to simulation-to-reality gaps due to planning inaccuracy and the consequent compounding error. Only executing the first action from the planned trajectory and re-planning at every step may help, but the inference latency renders the real-time execution infeasible.

Table A.XIV: **The real-robot performance difference between GMM and Diffusion Policy.** The policy error caused by simulation-to-reality gaps will be amplified by the Diffusion Policy because it plans and executes a future trajectory.

	Average	Stablize	Reach and Grasp	Insert	Screw
GMM	33.7%	35%	60%	5%	35%
Diffusion Policy	22.5%	35%	50%	5%	0%

D.3 Gating Function Justification and Conceptual Comparison

Recall several design choices in the proposed gating mechanism: 1) takes inputs of unstructured sensory observations (point cloud); 2) conditioned on base policy's outputs for effective prediction; 3) the intervention classifier shares the same feature encoder with the residual policy; and 4) the entire pipeline is learned end-to-end. We contrast against several mechanisms from the literature.

Table A.XV: **Gating mechanism conceptual comparison.**

	How to decide apply gating or not	**Input**	**Condition on base policy's outputs**	**Shared feature encoder**
Ours	End-to-end learned	Point cloud and proprioception	Yes	Yes
Residual Policy Learning [79]	No gating	Low-dimensional state	No	No
Residual RL [78]	No gating	Low-dimensional state	No	No
ThriftyDAgger [119]	Thresholded based on neural network ensemble	Low-dimensional state	No	No
Runtime Monitoring [104]	End-to-end learned	RGB and proprioception	No	Yes

Although this gating function and residual policy can be merged into one, we opt not to do so for better empirical performance. In the task *Screw*, using the gating function with the residual policy achieves a success rate of 85%, while only using the residual policy achieves a success rate of 55%. We hypothesize that this is mainly due to the data imbalance—because intervention and correction happen with a low frequency ($P_{correction} \approx 0.20$), most residual actions are zero. If we naively learn a residual policy from them, it will be biased toward predicting near-zero actions. On the

other hand, by training the residual policy only on human corrections, we offload learning whether to apply the correction to a gating function and thus reduce the negative effects of unbalanced data.

D.4 Long-Horizon Tasks Statistics

We show statistics about task length from FurnitureBench [85] in Table A.XVI.

Table A.XVI: **Statistics about long-horizon tasks from FurnitureBench [85].**

	Number of Steps	Average Human Demo Length
Lamp	594	2 Minutes
Square Table	1689	6 Minutes

E Extended Preliminaries

E.1 Intervention-Based Policy Learning

We adopt an intervention-based learning framework [66, 67, 91] where a human operator can intervene and take control during the execution of the robot base policy π_B. Denote the human policy as π_H, the following combined policy is deployed during data collection:

$$\pi^{deployed} = \mathbb{1}^H \pi^H + \left(1 - \mathbb{1}^H\right) \pi^B, \tag{A.7}$$

where $\mathbb{1}^H$ is a binary function indicating human interventions. Introducing a trajectory distribution $q(\tau)$ that consists of two observation-action distributions generated by the robot ρ^B and human operator ρ^H, the original RL objective leads to the maximization of a variational lower bound on logarithmic return [67, 121]:

$$\mathcal{J}(\theta, q) = \mathbb{E}_{q(\tau)} \left[\log R(\tau) + \log p_{\pi_\theta} - \log q(\tau)\right], \tag{A.8}$$

where p_{π_θ} is the induced trajectory distribution. While the human operator optimizes Eq. A.8 through intervention and correction, the robot learner maximizes it through

$$\theta = \arg\max_{\theta \in \Theta} \mathbb{E}_{(s,a)\sim q(\tau)} \left[\log \pi_\theta(a|s)\right]. \tag{A.9}$$

Various intervention-based policy learning methods have been derived by weighting observation-action pairs in Eq. A.9 differently. For example, HG-Dagger [66] completely ignores robot data $\mathcal{D}^B$ and only trains on human data $\mathcal{D}^H$ that contain intervention samples. This is equivalent to $q(\tau) \propto \rho^H$. Intervention Weighted Regression (IWR) [67] balances the data distribution by emphasizing human intervention: $q(\tau) \propto \alpha\rho^H + \rho^B$ with $\alpha = |\mathcal{D}^B|/|\mathcal{D}^H|$. Non-intervention-based methods such as traditional behavior cloning (BC) [86] only learn on $\mathcal{D}^H$ with full human demonstrations instead of intervention. This effectively sets $q(\tau) \propto \rho^H$.

F Extended Related Work

Robot Learning via Sim-to-Real Transfer Physics-based simulations [6–10, 49, 122–124] have become a driving force [1, 2] for developing robotic skills in tabletop manipulation [125–128], mobile manipulation [129–132], fluid and deformable object manipulation [133–136], dexterous in-hand manipulation [13–17], locomotion with various robot morphology [18–26, 137], object tossing [80], acrobatic flight [28, 29], etc. However, the domain gap between the simulators and the reality is not negligible [10]. Successful sim-to-real transfer includes locomotion [18–27], in-hand re-orientation for dexterous hands where objects are initially placed near the robot [13–17], and non-prehensile manipulation limited to simple tasks [30–39]. In this work, we tackle more challenging sim-to-real transfer for complex manipulation tasks and successfully demonstrate that our approach can solve sophisticated contact-rich manipulation tasks. More importantly, it requires significantly fewer real-robot data compared to the prevalent imitation learning and offline RL approaches [68, 69, 86]. This makes solutions that are based on simulators and sim-to-real transfer more appealing to roboticists.

Sim-to-Real Gaps in Manipulation Tasks Despite the complex manipulation skills recently learned with RL in simulation [138], directly deploying learned control policies to physical robots often fails. The sim-to-real gaps [10, 40, 44, 139] that contribute to this performance discrepancy can be coarsely categorized as follows: **a)** perception gap [18, 41–43], where synthetic sensory observations differ from those measured in the real world; **b)** embodiment mismatch [18, 44, 45], where the robot models used in simulation do not match the real-world hardware precisely; **c)** controller inaccuracy [46–48], meaning that the results of deploying the same high-level commands (such as in configuration space [140] and task space [141]) differ in simulation and real hardware; and **d)** poor physical realism [49], where physical interactions such as contact and collision are poorly simulated [87].

Although these gaps may not be fully bridged, traditional methods to address them include system identification [18, 30, 50, 51], domain randomization [13, 52–55], real-world adaptation [56], and simulator augmentation [58–60]. However, system identification is mostly engineered on a case-by-case basis. Domain randomization suffers from the inability to identify and randomize all physical parameters. Methods with real-world adaptation, usually through meta-learning [88], incur potential safety concerns during the adaptation phase. Most of these approaches also rely on explicit and domain-specific knowledge about tasks and the simulator *a priori*. For instance, to perform system identification for closing the embodiment gap for a quadruped, Tan et al. [18] disassembles the physical robot and carefully calibrates parameters including size, mass, and inertia. Kim et al. [32] reports that collaborative robots, such as the commonly used Franka Emika robot, have intricate joint friction that is hard to identify and randomized in typical physics simulators. To make a simulator more akin to the real world, Chebotar et al. [39] deploys trained virtual robots multiple times to refine the distributions of simulation parameters. This procedure not only introduces a significant real-world sampling effort, but also incurs potential safety concerns due to deploying suboptimal policies. In contrast, our method leverages human intervention data to implicitly overcome the transferring problem in a domain-agnostic way and also leads to a safer deployment.

Human-in-The-Loop Robot Learning Human-in-the-loop machine learning is a prevalent framework to inject human knowledge into autonomous systems [62, 89, 90]. Various forms of human feedback exist [63], ranging from passive judgement, such as preference [142–151] and evaluation [152–157], to active involvement, including intervention [158–160] and correction [161, 162]. They are widely adopted in solutions for sequential decision-making tasks. For instance, interactive imitation learning [66, 67, 91, 163] leverages human intervention and correction to help naïve imitators address data mismatch and compounding error. In the context of RL, reward functions can be derived to better align agent behaviors with human preferences [145, 148, 149, 152]. Noticeably, recent trend focuses on continually improving robots' capability by iteratively updating and deploying policies with human feedback [91], combining active human involvement with RL [162], and autonomously generating corrective intervention data [92]. Our work further extends this trend by showing that sim-to-real gaps can be effectively eliminated by using human intervention and correction signals.

In shared autonomy, robots and humans share the control authority to achieve a common goal [64, 65, 93–95]. This control paradigm has been largely studied in assistive robotics and human-robot collaboration [96–98]. In this work, we provide a novel perspective by employing it in sim-to-real transfer of robot control policies and demonstrating its importance in attaining effective transfer.

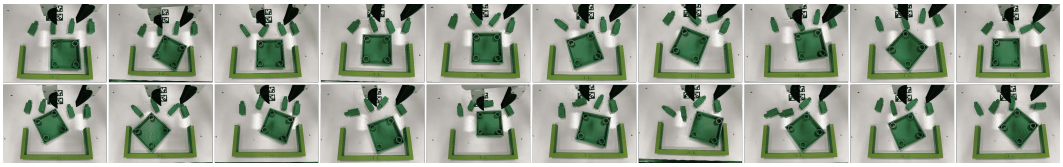


Figure A.14: **Initial settings for evaluating the task *Stabilize*.**

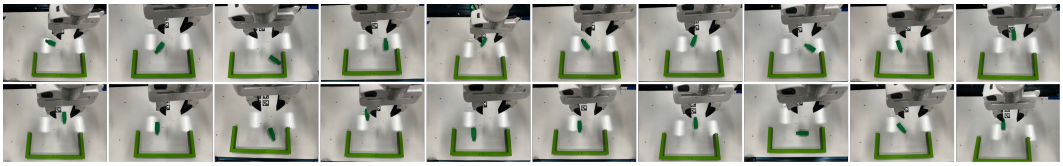


Figure A.15: **Initial settings for evaluating the task *Reach and Grasp*.**

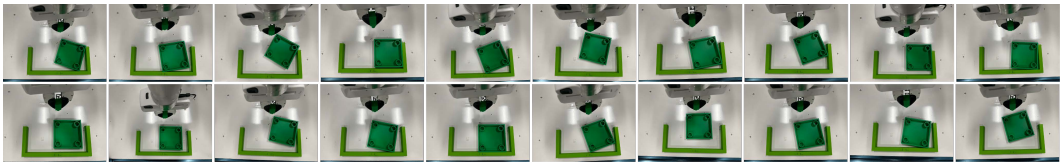


Figure A.16: **Initial settings for evaluating the task *Insert*.**

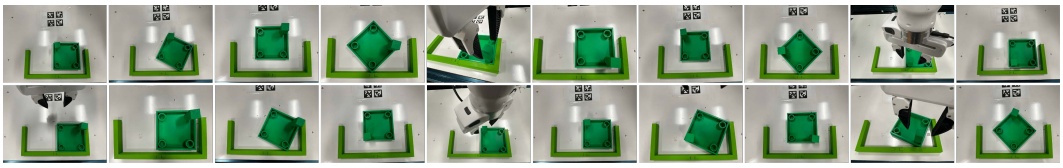


Figure A.17: **Initial settings for evaluating the task *Screw*.**

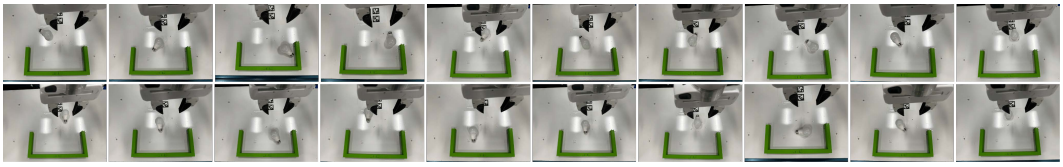


Figure A.18: **Initial settings for the experiment *Object Asset Mismatch*.**