RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation

Tianxing Chen[2,15*†], **Zanxin Chen**[3,5*], **Baijun Chen**[7*], **Zijian Cai**[3,5*], **Yibin Liu**[13*], **Qiwei Liang**[5], **Zixuan Li**[5], **Xianliang Lin**[5], **Yiheng Ge**[1], **Zhenyu Gu**[8], **Weiliang Deng**[3,11], **Yubin Guo**[9], **Tian Nian**[3,5], **Xuanbing Xie**[12], **Qiangyu Chen**[5], **Kailun Su**[5], **Tianling Xu**[10], **Guodong Liu**[6], **Mengkang Hu**[2], **Huan-ang Gao**[6,15], **Kaixuan Wang**[2,15], **Zhixuan Liang**[2,3†], **Yusen Qin**[4,6], **Xiaokang Yang**[1], **Ping Luo**[2,14✉], **Yao Mu**[1,3✉†]

[1] SJTU ScaleLab‡, [2] HKU MMLab‡, [3] Shanghai AI Lab, [4]D-Robotics, [5]SZU, [6]THU, [7]NJU, [8]FDU, [9]USTC, [10]SUSTech, [11]SYSU, [12]CSU, [13]NEU, [14]HKU-Shanghai ICRC, [15]Lumina EAI

[*] Equal contribution ✉ Corresponding authors [†] Co-project leads

[‡] Equally leading organizations

https://robotwin-platform.github.io

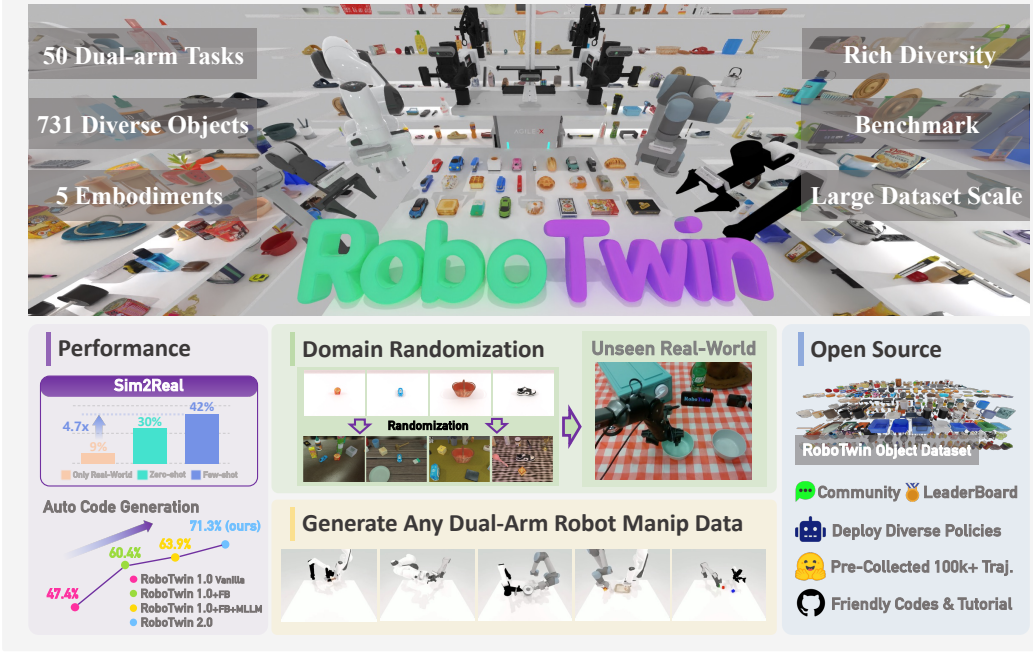

Figure 1: **Overview of RoboTwin 2.0.** RoboTwin 2.0 is a scalable framework for data generation and benchmarking in bimanual robotic manipulation. It integrates an expert data generation pipeline and a 50-task benchmark built on the RoboTwin Object Dataset (731 objects, 147 categories). A multimodal language model agent enables automatic task program synthesis, while flexible dual-arm configurations facilitate scalable and diverse data collection. Policies trained on RoboTwin 2.0 data demonstrate improved robustness and generalization to unseen environments.

Abstract

Simulation-based data synthesis has emerged as a powerful paradigm for enhancing real-world robotic manipulation. However, existing synthetic datasets remain insufficient for robust bimanual manipulation due to two challenges: (1) the lack

of an efficient, scalable data generation method for novel tasks, and (2) oversimplified simulation environments that fail to capture real-world complexity. We present **RoboTwin 2.0**, a scalable simulation framework that enables automated, large-scale generation of diverse and realistic data, along with unified evaluation protocols for dual-arm manipulation. We first construct RoboTwin-OD, a large-scale object library comprising 731 instances across 147 categories, each annotated with semantic and manipulation-relevant labels. Building on this foundation, we develop an expert data synthesis pipeline that combines multimodal large language models (MLLMs) with simulation-in-the-loop refinement to generate task-level execution code automatically. To improve sim-to-real transfer, RoboTwin 2.0 incorporates structured domain randomization along five axes: clutter, lighting, background, tabletop height and language instructions, thereby enhancing data diversity and policy robustness. We instantiate this framework across 50 dual-arm tasks spanning five robot embodiments, and pre-collect over 100,000 domain-randomized expert trajectories. Empirical evaluation shows a 10.9% gain in code generation success rate and improved generalization to novel real-world conditions. A vision-language-action (VLA) model fine-tuned on our dataset achieves a 367% relative improvement (42.0% vs. 9.0%) on unseen scene real-world tasks, while zero-shot models trained exclusively on our synthetic data attain a 228% relative gain, demonstrating strong generalization without real-world supervision. We release the data generator, benchmark, pre-collected dataset, and code to support scalable research in robust bimanual manipulation.

1 Introduction

Bimanual robotic manipulation is critical for enabling robots to perform complex real-world tasks such as collaborative assembly, tool use, and object handovers. Developing generalizable bimanual policies—particularly vision–language–action (VLA) foundation models—requires datasets that are simultaneously high-quality, diverse, and large-scale. In the absence of sufficient variability in object geometry, scene clutter, lighting conditions, instruction language, and robot embodiments, learned policies often overfit to narrow distributions and fail to generalize to novel environments or hardware platforms. Yet collecting real-world demonstrations at scale remains prohibitively expensive, time-consuming, and logistically challenging, especially when aiming to cover a broad range of tasks, objects, and embodiments.

Simulation-based data generation provides a scalable alternative for collecting large-scale multimodal datasets and has shown promise in enabling sim-to-real transfer [30, 10]. However, existing pipelines fall short in three critical aspects. First, they lack automated quality control: without an expert-level validation loop, many generated trajectories include execution failures or suboptimal grasps, which degrade policy learning. Second, their domain randomization is often superficial, yielding overly clean and homogeneous scenes that omit essential real-world factors such as clutter, lighting variation, and ambiguous language instructions—elements crucial for robust sim-to-real transfer. Third, they overlook cross-embodiment variation: different bimanual platforms can differ substantially in their kinematic capabilities and grasp strategies. For example, a low-degree-of-freedom (DoF) platform like the Piper often relies on lateral grasps due to its limited dexterity, whereas a high-DoF arm such as the Franka is capable of top-down precision grasps. Yet, current synthetic datasets rarely encode such embodiment-specific affordances or task constraints, limiting their generality.

To address these challenges, we introduce **RoboTwin 2.0**, a scalable simulation-based data generation framework designed to produce high-quality, diverse, realistic, and interaction-rich datasets for bimanual manipulation. RoboTwin 2.0 integrates three key components: (1) an automated expert data generation pipeline that leverages multimodal large language models (MLLMs) and simulation-in-the-loop feedback to iteratively validate and refine task execution code; (2) comprehensive domain randomization over language instructions, object clutter, background textures, lighting conditions, and tabletop configurations, aimed at closing the sim-to-real gap and enhancing policy generalization; and (3) embodiment-aware adaptation, in which object affordances are annotated and robot-specific action candidates are generated to account for heterogeneous dual-arm kinematics.

Building on these components, we introduce three new resources to support scalable research in bimanual manipulation: (1) the RoboTwin-OD asset library, comprising 731 annotated object

instances across 147 categories; (2) an automated data generation pipeline with comprehensive domain randomization and a collection of over 100,000 expert trajectories spanning 50 tasks across five dual-arm robot platforms; and (3) a benchmark for evaluating policy generalization to cluttered environments and open-ended language goals. Together, these resources enable the community to train and evaluate robust bimanual manipulation policies under conditions that closely reflect real-world complexity and diversity.

In summary, our main contributions are as follows: (1) We develop an automated expert data generation framework that integrates multimodal large language models with simulation-in-the-loop feedback to ensure high-quality, expert-level trajectories; (2) We propose a systematic domain randomization strategy that enhances policy robustness by increasing data diversity and sim-to-real generalization; (3) We introduce an embodiment-aware adaptation mechanism that generates robot-specific manipulation candidates based on object affordances; (4) We release the RoboTwin-OD asset library, a large-scale pre-collected multi-embodiment domain-randomized trajectory dataset, a scalable bimanual data generator, and a standardized evaluation benchmark to support scalable training and evaluation of generalizable policies across different robot embodiments, scene configurations, and language instructions.

2 Method

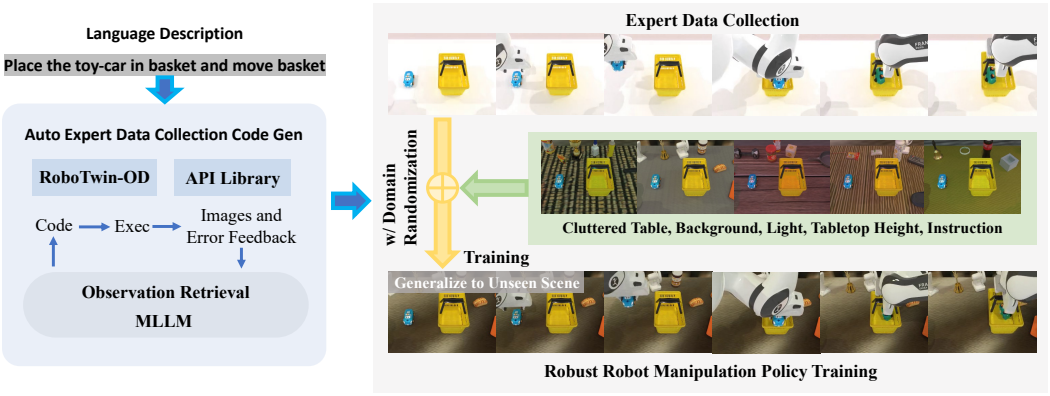

Figure 2: **RoboTwin 2.0 Pipeline.** Leveraging RoboTwin-OD and a predefined skill API, our framework employs an MLLM-based expert code generation module with simulation-in-the-loop feedback to automatically synthesize expert-level task programs. These programs are used to generate diverse, domain-randomized trajectories that support downstream policy training and evaluation.

We illustrate the overall RoboTwin 2.0 pipeline in Fig. 2. The framework begins with a task code generation module that leverages multimodal large language models (MLLMs) and simulation-in-the-loop feedback to automatically synthesize executable task plans from natural language instructions. This module is grounded on a large-scale object asset library (RoboTwin-OD) and a predefined skill library, enabling scalable task instantiation across a broad range of object categories and manipulation scenarios. To ensure high-quality expert demonstrations, we integrate this automated generation pipeline with RoboTwin 2.0's comprehensive domain randomization scheme, which diversifies observations along language, visual, and spatial axes. This pipeline supports the synthesis of diverse and realistic training data, facilitating the development of manipulation policies that are robust to real-world environmental variability.

2.1 Expert Code Generation via MLLMs and Simulation-in-the-Loop Feedback

Recent advances in language models have demonstrated their capability to generate intermediate task representations—such as textual plans [16, 15], API calls, or executable code [29, 6]—for complex robotic tasks. Multimodal large language models (MLLMs) extend this capability by incorporating visual and proprioceptive signals, enabling more grounded reasoning over real-world sensory inputs. However, most prior systems either rely on strong manual priors or lack robust closed-loop feedback during program synthesis, limiting their reliability in diverse or dynamic environments.

Building on this foundation, we propose an automated expert data generation pipeline that integrates programmatic code synthesis with multimodal execution feedback to produce high-quality manipulation programs, as illustrated in Figure 3. The system operates through a closed-loop architecture comprising two AI agents: a code-generation agent and a vision-language model (VLM) observer. By executing and monitoring code within a simulated environment, the observer systematically detects execution failures and suggests corrections, enabling the code-generation agent to iteratively refine the task program. This feedback loop facilitates the generation of robust, self-improving expert data with minimal human supervision.

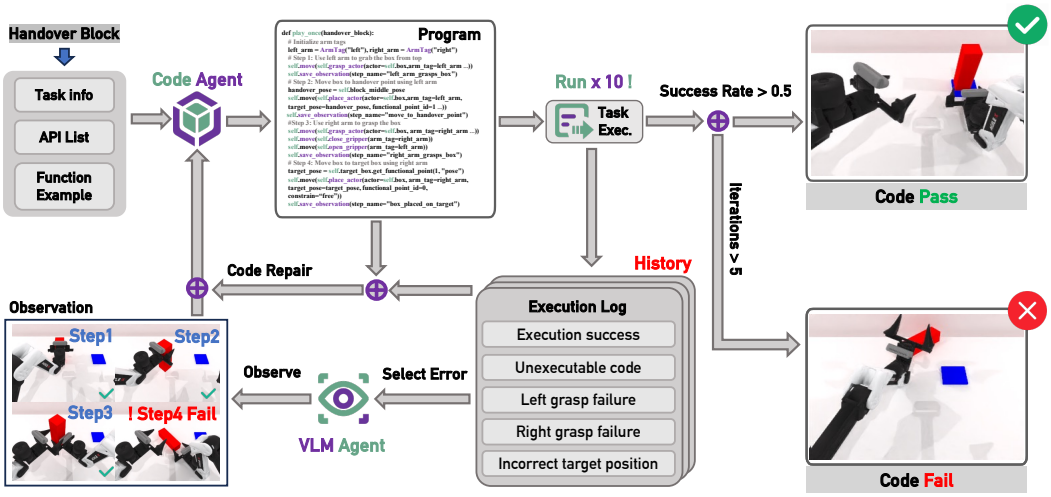

Figure 3: **Expert Code Generation Pipeline.**

Input Specification. Each task is defined by a task name (e.g., *Handover Block*) and a natural language description of the objective. The code-generation agent is conditioned on three key inputs: a general API list, a set of example function calls, and a hierarchical constraint specification. These components collectively guide the synthesis of Python code to execute the task. Additionally, each task may include task-specific function call examples to further ground code generation in context.

Initial Code Generation. The code-generation agent synthesizes an initial Python program conditioned on the provided task inputs. It models the program synthesis process as a structured prediction problem over the space of available API calls, leveraging natural language understanding and few-shot prompting from task-specific examples. The generated code specifies a stepwise sequence of robot actions designed to accomplish the target manipulation objective.

Simulated Execution and Logging. The generated program is executed ten times per iteration within a simulated robotic environment. Multiple trials are used to account for stochastic variations in simulation dynamics, robot controllers, and sensor noise. After each execution batch, the system generates a structured execution log that records the success or failure of each trial and annotates failure cases with their corresponding causes—such as unexecutable code, left/right grasp failure, or incorrect object placement.

Multimodal Observation and Error Localization. In parallel with execution, a vision-language model (VLM) agent observes the robot's behavior across all ten trials. The VLM performs frame-by-frame inspection to evaluate the success of each program step and localize the point of failure when errors occur. Beyond temporal localization, the VLM also diagnoses failure modes by inferring whether the underlying cause stems from flawed logic, incorrect API usage, or other systemic issues. This diagnostic capability enables the system to address root causes rather than merely responding to superficial execution errors.

Code Repair and Iterative Refinement. The code-generation agent receives two complementary feedback signals: (i) a quantitative execution log and (ii) a qualitative, localized diagnostic from the VLM. It integrates these inputs to revise the program by modifying or replacing instructions identified as failure-prone. The updated program is then re-evaluated in the next iteration. This refinement loop continues until predefined termination criteria are satisfied, yielding expert-level task code with minimal human supervision.

Termination Criteria. The refinement process terminates under one of two conditions: (i) the generated program achieves a success rate exceeding 50% across ten simulated executions in a single iteration, or (ii) the system fails to reach this threshold after five consecutive iterations. These criteria prevent indefinite refinement and ensure that only programs meeting a minimum standard of task competence are retained.

The outcome of this pipeline is a collection of robust, automatically synthesized programs that generate high-quality expert trajectories for downstream training and evaluation. By integrating multimodal reasoning with execution-level feedback, the system produces code that is not only syntactically correct but also semantically aligned with task objectives. This closed-loop generation framework substantially reduces human supervision while enabling scalable and self-improving expert data creation for complex robotic manipulation tasks.

2.2 Domain Randomization for Robust Robotic Manipulation

To improve policy robustness to real-world environmental variability, we apply domain randomization across five key dimensions: (1) cluttered placement of task-irrelevant objects, (2) background textures, (3) lighting conditions, (4) tabletop heights, and (5) diverse language instructions. This systematic diversification enriches the training data distribution and significantly improves generalization to unseen scenarios. Fig. 4 (a) visualizes the effects of our domain randomization strategy.

Figure 4: **Visualization of domain randomization and our texture library.**

Scene Clutter. To improve policy robustness to environmental variation, we introduce cluttered tabletop scenes by randomly populating the workspace with task-irrelevant distractor objects. Leveraging our internally constructed object asset library, RoboTwin-OD, illustrated in Section 3.1, which contains 731 fully annotated objects across 147 categories, we synthesize diverse and semantically rich cluttered scenes during data generation. Each object is annotated with placement points, enabling a generic placement API that inserts arbitrary objects into the scene with semantically valid poses. To ensure physical plausibility, we precompute collision volumes and apply collision-aware placement, avoiding unrealistic overlaps introduced by randomization. Additionally, we annotate intra-class similarity groups within RoboTwin-OD to encode visual and functional similarity. During clutter sampling, we restrict the placement of distractors that are visually or semantically similar to task-relevant objects, thereby reducing ambiguity and mitigating potential policy confusion.

Diverse Background Textures. We randomize tabletop surfaces and surrounding backgrounds using a large library of textures generated via procedural methods and generative models. This exposes policies to a broad visual distribution and mitigates overfitting to clean, synthetic environments. To construct this texture library, we first leverage large language models (LLMs) combined with web crawling to collect 1,000 diverse text prompts describing real-world surface appearances, varying in style, color, and granularity. Using these prompts, we employ Stable Diffusion v2 to synthesize 20 texture samples per description, resulting in an initial pool of 20,000 candidate textures. A human-in-the-loop verification step filters out low-quality or irrelevant samples, yielding a curated set of 12,000 high-quality textures. This library is used to randomize background environments and tabletop surfaces during simulation, significantly enriching the visual diversity of the generated data. Representative samples are shown in Fig. 4 (b).

Lighting Variation. Real-world environments exhibit significant lighting variability, including differences in color temperature, light source type (e.g., point lights, area lights), number of sources, and spatial configuration. Such variations alter object appearance, shading, and reflections in 2D visual inputs, posing challenges for vision-based manipulation policies. To improve robustness under diverse lighting conditions, we apply lighting randomization in our simulation pipeline. Specifically, we randomize the color, type, intensity, and position of light sources within physically plausible bounds to simulate realistic illumination diversity. As shown in Fig. 4 (second row), variations in color temperature can dramatically shift the perceived appearance of objects—for instance, changing the color of a shoe under warm versus cool lighting. By training under diverse lighting configurations, policies become more robust to real-world illumination shifts.

Tabletop Heights. In real-world settings, the height of manipulation platforms—typically tables—can vary due to differences in workspace layout and hardware configurations. These variations affect robot perception, kinematics, and interaction strategies, making policy generalization more challenging. To improve robustness to such physical discrepancies, we randomize tabletop heights during simulation. Specifically, during data generation, the table height is uniformly sampled from a physically plausible range, introducing variability in viewpoint and spatial relationships between the robot and manipulated objects.

Trajectory-Level Diverse Language Instructions. To improve policy robustness to variation in natural language, we automatically generate diverse task instructions and object descriptions using a multimodal large language model. These include (i) task instruction templates and (ii) object descriptions that reflect geometry, appearance, and part-level attributes. The prompts used for generation are listed in Appendix E, with examples shown in Appendix F. For each trajectory, we construct language instructions by sampling from task instruction templates and object descriptions. For example, in the *Move Can Pot* task, a template such as "Use {a} to place {A} to the left of {B}" is instantiated with sampled object descriptions (e.g., {A}: "sauce can" or "white plastic lid sauce can"; {B}: "gray kitchenpot" or "kitchenpot for boiling and cooking") and robot parameters (e.g., left arm). This results in instructions such as "Use left arm to place white plastic lid sauce can to the left of kitchenpot for boiling and cooking" or "Use left arm to place sauce can to the left of gray kitchenpot." This compositional augmentation introduces linguistic variety at the trajectory level and improves generalization to unseen instructions and scene configurations.

2.3 Embodiment-Aware Grasp Adaptation

Due to differences in DoF and kinematic structures, robotic arms exhibit varying reachable workspaces and preferred manipulation strategies for the same task. For example, when grasping a can, the Franka arm typically favors a top-down approach, while the lower-DoF Piper arm is better suited to side grasps. As a result, a task successfully completed by Franka using a top-down grasp may require a side approach when executed with Piper, as shown in Fig. 6.

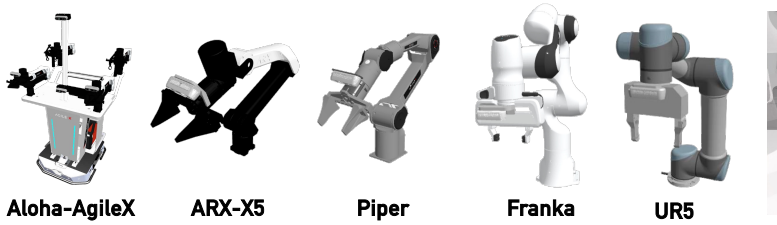

Figure 5: **Five RoboTwin 2.0 Embodiment (Aloha-AgileX, ARX-X5, Piper, Franka and UR5).**

Figure 6: **Different Grasping Behavior.**

To address these embodiment-specific variations, we annotate each object with a diverse set of manipulation candidates that span multiple grasp axes and approach directions. This ensures that the dataset captures both manipulation diversity and robot-specific preferences. Specifically, for each object, we generate candidate grasps by incorporating preferred operation directions, randomized pose perturbations, and parallelized motion planning attempts. Additionally, we introduce angular perturbations toward directions with higher arm reachability, further expanding the space of feasible manipulation poses.

3 RoboTwin 2.0 Data Generator, Benchmark and Large Scale Dataset

3.1 RoboTwin-OD: RoboTwin Object Dataset

To enhance both manipulation capability and visual understanding, we construct a large-scale object dataset with rich semantic annotations, called RoboTwin-OD, covering 147 categories and 731 diverse objects. Specifically, this includes 534 instances across 111 categories with custom-generated and optimized meshes, 153 objects from 27 categories in Objaverse [9], and 44 articulated object instances from 9 categories in SAPIEN PartNet-Mobility [40]. Objects from all sources, including Objaverse, are used for cluttered scene construction, with Objaverse specifically serving to further increase the visual and semantic diversity of distractor objects. Additionally, we develop a comprehensive surface and background texture library using generative AI and human-in-the-loop verification to ensure both diversity and realism.

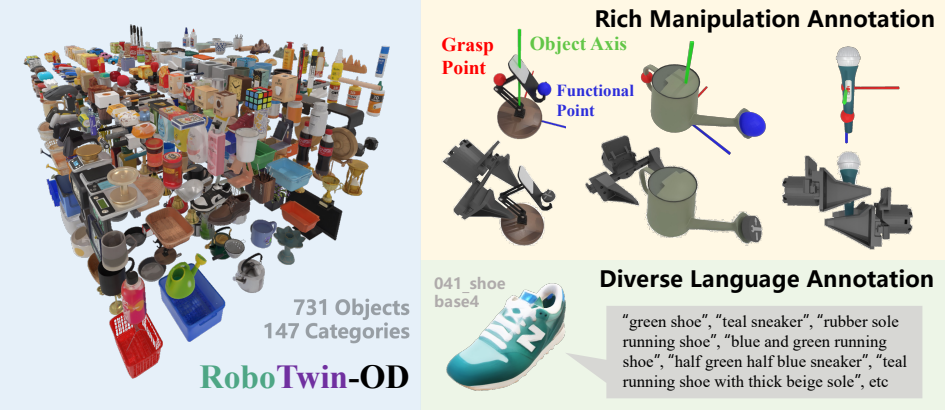

Figure 7: **RoboTwin-OD.** A large-scale object dataset for robotic manipulation with 147 categories and 731 objects, annotated with rich interaction labels and diverse language descriptions.

For robust robotic manipulation, it is essential for policies to generalize across diverse objects, which requires a highly varied training dataset. Specifically, the dataset should include a broad range of object categories to promote general manipulation skills, as well as diverse instances within each category to capture intra-class manipulation patterns. In addition, high-quality language annotations are crucial for grounding object understanding. To this end, we developed an automated object description generator and applied it to our entire dataset, followed by human verification. For each object, we generate 15 language annotations, spanning both seen and unseen descriptions. This variation captures differences in object shape, texture, category, functionality, part structure, and description granularity.

To support learning object-centric interaction strategies, we further annotate each object with key point-axis information. These include placement points, functional points, grasping points, and grasp axis directions, explicitly encoding object affordances. Combined with our robotic manipulation API library, these annotations enable generalizable grasp execution in simulation. All objects information can be found in http://robotwin-platform.github.io/doc/objects/.

3.2 Support for Flexible Embodiment Combinations

Our object-centric, embodiment-agnostic data generation framework enables seamless deployment across a wide range of dual-arm robotic systems. The pipeline supports flexible embodiment configurations, allowing arbitrary combinations of heterogeneous manipulators and relative arm placements. This design ensures compatibility with diverse hardware setups and facilitates extensibility to future robotic platforms.

To execute high-success-rate manipulation trajectories across different embodiments (see Section 2.3), we integrate Curobo, a high-performance, GPU-accelerated motion planner that enables efficient and reliable planning under varied kinematic constraints.

Figure 8: **Heterogeneous Dual-Arm Control via Object-Centric Manipulation.**

Currently, our framework supports five robotic arms—Franka, Piper, UR5, ARX-X5, and Aloha-AgileX—along with multiple gripper types, including the Panda gripper and WSG gripper. As shown in Fig. 8, we demonstrate successful task executions across a variety of dual-arm pairings, highlighting RoboTwin 2.0's ability to scale to heterogeneous robot configurations and its readiness for future real-world deployment.

3.3 50 Tasks for Data Generation and Benchmarking

Building on our automated task generation framework, embodiment-adaptive behavior synthesis, and the large-scale object asset library RoboTwin-OD, we construct a suite of over 50 dual-arm collaborative manipulation tasks. In addition, we support data collection and evaluation across 5 distinct robot platforms, enabling comprehensive benchmarking of manipulation policies. Keyframes from representative tasks are shown in Fig. 9, and the complete task set is available at http://robotwin-platform.github.io/doc/tasks/.

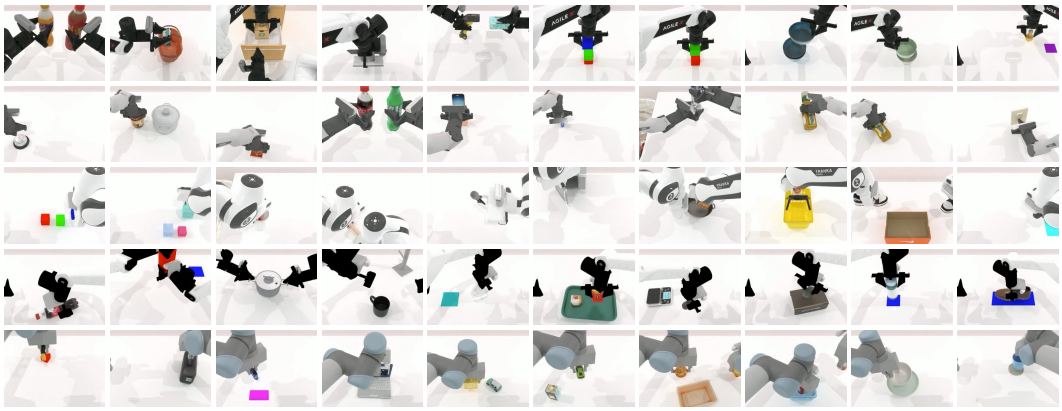

Figure 9: **50 RoboTwin 2.0 Bimanual Manipulation Tasks**.

3.4 Pre-collected RoboTwin 2.0 Dataset

We collected over 100,000 dual-arm manipulation trajectories across 50 tasks in RoboTwin2.0, covering five distinct dual-arm embodiments. For each task-embodiment pair, we provide 100 clean (non-randomized) trajectories and 400 domain-randomized trajectories. The full dataset is available at: https://huggingface.co/datasets/TianxingChen/RoboTwin2.0/tree/main/RoboTwin2_dataset.

RoboTwin 2.0 supports a wide range of embodiment configurations and scene settings, making it a highly versatile framework for large-scale robotic data collection. It enables efficient, automated trajectory generation across diverse tasks and hardware setups with minimal human supervision.

4 Experiment

We design experiments to evaluate the effectiveness of RoboTwin 2.0 in three key aspects: (1) automating the generation of high-quality expert code for manipulation tasks; (2) improving policy robustness to environmental variation via diversified training data; and (3) demonstrating the utility and diversity of RoboTwin 2.0 as a standardized benchmark for evaluating policy generalization across tasks, scenes, and embodiments.

4.1 Evaluation of Automated Expert Code Generation

In this section, we evaluate our closed-loop expert data generation system on a suite of 10 robot manipulation tasks, using programmatic generation followed by iterative refinement in simulation. Each task is specified with a natural language instruction, and for each system variant, the code-generation agent produces 10 candidate programs, each executed 10 times to account for stochasticity in dynamics, control, and perception. Task-level success is defined as the average success rate across all executions of all candidates, as described in Section 2.1.

Table 1: **Overall performance comparison across RoboTwin variants.** Evaluated on the subset of tasks supported by both RoboTwin 1.0 and RoboTwin 2.0. Per-task success rate comparison is provided in Appendix 8.

Method	ASR	Top5-ASR	CRSR	CR-Iter	Token
R1.0 Vanilla	47.4%	47.4%	47.4%	1.00	1236.6
R1.0 + FB	60.4%	71.4%	60.4%	2.46	1190.4
R1.0 + MM FB	63.9%	74.2%	63.9%	2.42	1465.0
R2.0 Vanilla	62.1%	62.1%	62.1%	1.00	**569.4**
R2.0 + FB	66.7%	73.6%	66.7%	1.89	581.6
R2.0 + MM FB	**71.3%**	**78.6%**	**71.3%**	**1.76**	839.7

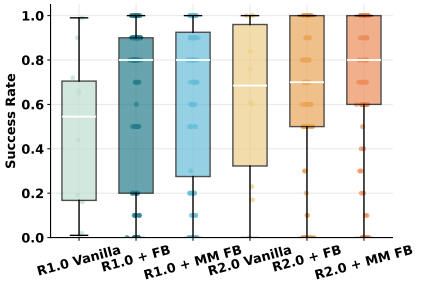

Figure 10: **RoboTwin Success Rate Distribution.**

Table 2: **Per-task success rates of our proposed R2.0 + MM FB algorithm on all RoboTwin 2.0-supported tasks.**

Task	Rate	Task	Rate	Task	Rate	Task	Rate
Adjust Bottle	100%	Beat Block Hammer	53%	Blocks Ranking Rgb	80%	Blocks Ranking Size	80%
Click Alarmclock	0%	Click Bell	10%	Dump Bin Bigbin	0%	Grab Roller	74%
Handover Block	27%	Handover Mic	0%	Hanging Mug	0%	Lift Pot	40%
Move Can Pot	30%	Move Pillbottle Pad	50%	Move Playingcard Away	90%	Move Stapler Pad	100%
Open Laptop	0%	Open Microwave	0%	Pick Diverse Bottles	62%	Pick Dual Bottles	100%
Place A2B Left	50%	Place A2B Right	60%	Place Bread Basket	0%	Place Bread Skillet	0%
Place Can Basket	0%	Place Cans Plasticbox	100%	Place Container Plate	82%	Place Dual Shoes	22%
Place Empty Cup	85%	Place Fan	70%	Place Burger Fries	100%	Place Mouse Pad	100%
Place Object Basket	0%	Place Object Scale	80%	Place Object Stand	90%	Place Phone Stand	0%
Place Shoe	100%	Press Stapler	0%	Put Bottles Dustbin	0%	Put Object Cabinet	0%
Rotate Qrcode	80%	Scan Object	0%	Shake Bottle	0%	Shake Bottle Horizontally	0%
Stack Blocks Three	82%	Stack Blocks Two	100%	Stack Bowls Three	20%	Stack Bowls Two	30%
Stamp Seal	20%	Turn Switch	0%	*Avg Success Rate*	*43.34%*		

To quantitatively characterize performance, we report five metrics: **ASR** (Average Success Rate), which measures mean task-level success over all candidates; **Top5-ASR**, the average success of the top 5 performing programs per task; **CRSR** (Code Revision Success Rate), the success rate after feedback-based refinement; **CR-Iter**, the average number of iterations taken before termination; and **Token**, which reflects the average LLM token cost incurred during program synthesis and repair.

Table 1 presents the evaluation results across six configurations of our system, computed over the subset of tasks supported by both RoboTwin 1.0 and 2.0 (see Appendix 8 for per-task success rates). The *Vanilla* setting corresponds to one-shot code generation without any iterative correction. In the *FB* setting, the system incorporates structured feedback from execution logs to revise failure-prone code. The *MM FB* configuration further augments this process with multimodal observation, allowing a vision-language model to localize and classify failures more precisely.

Across all configurations (evaluated on tasks jointly supported by RoboTwin 1.0 and 2.0), we observe that incorporating multimodal feedback consistently improves performance. For instance, within the RoboTwin 1.0 architecture, introducing feedback increases ASR from 47.4% (Vanilla) to 60.4%, and rises further to 63.9% when multimodal feedback is used. A similar trend is observed in RoboTwin 2.0, where the ASR improves from 62.1% (Vanilla) to 66.7% with execution feedback, and reaches 71.3% with the addition of vision-language-guided repair. Improvements are also evident in Top5-ASR, indicating that high-quality program candidates benefit even more from refinement strategies that incorporate perceptual understanding. These results suggest that multimodal observation not only detects failure but also provides critical context for generating more effective code repairs.

Another notable trend is the reduction in the average number of repair iterations. In the feedback-only setting, RoboTwin 2.0 requires fewer steps (1.89 vs. 2.46 in RoboTwin 1.0) to reach satisfactory performance. This pattern persists in the multimodal feedback setting, where RoboTwin 2.0 converges in 1.76 iterations on average, compared to 2.42 for RoboTwin 1.0. This suggests that RoboTwin 2.0 benefits from stronger priors in initial code generation and greater sample efficiency in refinement. Additionally, token costs are substantially reduced in RoboTwin 2.0 under the Vanilla setting (569.4 vs. 1236.6 in RoboTwin 1.0), indicating more concise and accurate initial programs.

Figure 10 visualizes the full distribution of success rates across all tasks. RoboTwin 1.0 in the Vanilla setting shows high variance and a low median, with many samples falling below the success threshold. Feedback-based configurations reduce this variance and improve central performance. In particular, RoboTwin 2.0 with multimodal feedback yields highly compact distributions centered above 80%, demonstrating both robustness and reliability. These distributional improvements reflect not only average-case performance gains but also stronger guarantees in worst-case scenarios—crucial for downstream deployment in real-world robotic systems.

Table 2 reports per-task success rates for the R2.0 + MM FB configuration across all RoboTwin 2.0-supported tasks. While many tasks yield high success (e.g., 100% on "Adjust Bottle" and "Pick Dual Bottles"), others remain unsolved. Tasks with 0% success—such as "Open Laptop" and "Shake Bottle"—highlight limitations of the code generation model in handling articulated objects, precise pose constraints, and dynamic interactions. These behaviors are difficult to specify through current action abstractions and remain challenging for program synthesis without richer multimodal memory or fine-grained control.

Taken together, these results highlight three key findings. First, vision-language feedback meaningfully improves the quality and precision of code revisions, enabling the system to address not just what failed but why it failed. Second, architectural improvements in RoboTwin 2.0 reduce both iteration count and language model token cost, enhancing efficiency. Third, the combination of symbolic execution logs with perceptual diagnostics yields expert data that is more reliable, more scalable, and more aligned with task-level semantics. These observations validate the closed-loop, self-improving architecture proposed in this work and underscore the value of multimodal reasoning in grounded code generation systems.

4.2 Evaluating Efficiency with and without Adaptive Grasping

Table 3: **Overall Performance Comparison between RoboTwin 1.0 and RoboTwin 2.0.**

Method	Aloha-AgileX	Piper	Franka	UR5	ARX-X5	*Average*
RoboTwin 1.0	65.1%	2.4%	**67.3%**	**57.6%**	68.6%	52.2%
RoboTwin 2.0	**78.8%**	**25.1%**	67.2%	57.1%	**74.2%**	**60.5%**
Difference	**+13.7%**	**+22.7%**	**-0.1%**	**-0.5%**	**+5.6%**	**+8.3%**

To evaluate the effectiveness of our embodiment-aware grasp augmentation strategy, we measure the task success rate of automated data collection across 50 RoboTwin 2.0 tasks on five different robot embodiments. As shown in Table 3, we compare our RoboTwin 2.0 pipeline against the RoboTwin 1.0 baseline, which lacks diverse grasping and candidate augmentation. Results show that our method improves success rates, particularly for robots with constrained planning spaces, achieving an average improvement of 8.3% across all embodiments. Specifically, for high-DoF arms with large reachable workspaces, such as Franka and UR5 (7-DoF), success rates remain largely unchanged, indicating limited benefit when the robot already has sufficient kinematic flexibility. However, for lower-DoF platforms such as Aloha-AgileX, Piper, and ARX-X5 (6-DoF), our method leads to substantial gains of 13.5%, 22.7%, and 5.6%, respectively. These results demonstrate that our approach provides additional feasible grasp options that effectively mitigate the planning limitations of low-DoF manipulators. Success rates for all tasks can be found in Appendix H.

4.3 Assessing the Impact of RoboTwin 2.0 on Policy Robustness

To assess the impact of RoboTwin 2.0 data on improving model robustness to environmental perturbations, we design an ablation study based on the VLA framework. We independently collect a

total of 9,600 expert trajectories across 32 tasks (300 per task) under two distinct settings: RoboTwin 2.0 with domain randomization and RoboTwin 1.0, which is RoboTwin 2.0 without randomization. Each dataset is used separately to fine-tune pretrained models: RDT for 30,000 steps and Pi0 for 10,000 steps, using offline imitation learning. To evaluate generalization, we select five unseen tasks and collect 50 new expert demonstrations under clean (non-randomized) conditions for single-task training and fine-tuning. As a control group, we include non-pretrained versions of ACT, DP, RDT, and Pi0. This setup enables a controlled evaluation of the effects of domain-randomized data on policy robustness in previously unseen environments. All experimental configurations are provided in Appendix C.

Table 4: **Evaluating the Impact of RoboTwin 2.0 on Policy Robustness.**

Simulation Tasks	ACT	DP	RDT	Pi0	RDT +R1.0	Pi0 +R1.0	RDT +R2.0	Pi0 +R2.0
Stack Bowls Two	0.0%	0.0%	28.0%	41.0%	21.0%	29.0%	36.0%	55.0%
Pick Dual Bottles	0.0%	0.0%	5.0%	10.0%	13.0%	16.0%	20.0%	20.0%
Move Can Pot	0.0%	0.0%	20.0%	32.0%	23.0%	48.0%	25.0%	32.0%
Place Object Basket	0.0%	0.0%	11.0%	10.0%	14.0%	3.0%	17.0%	18.0%
Place Shoe	0.0%	0.0%	10.0%	12.0%	12.0%	7.0%	29.0%	24.0%
Average	0.0%	0.0%	14.8%	21.0%	16.6%	20.6%	**25.4%**	**29.8%**

As shown in Table 4, we observe that models fine-tuned with RoboTwin 1.0 data show negligible improvements in average success rate compared to their pretrained counterparts, indicating that data without domain randomization does not help the model handle environmental variations. In contrast, models pretrained with RoboTwin 2.0 data exhibit significantly improved generalization. Specifically, RDT and Pi0 achieve absolute gains of 10.6% and 8.8%, corresponding to relative improvements of 71.6% and 41.9%, respectively. Notably, this performance gain persists even though the downstream tasks were trained using only clean, non-randomized data. This suggests that domain-randomized pretraining with RoboTwin 2.0 effectively equips models with robustness to visual and spatial variations. As a result, models pretrained with RoboTwin 2.0 can adapt to new tasks without requiring additional data augmentation or complex scene variations.

4.4 Evaluation on Sim-to-Real Performance

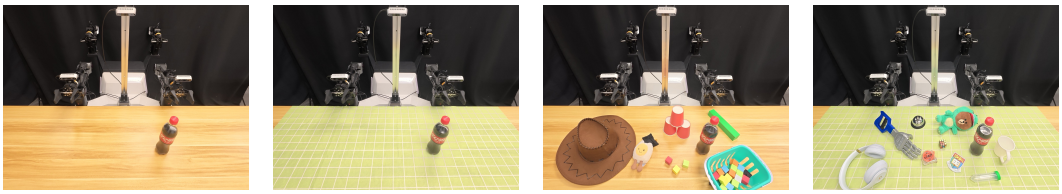

Figure 11: **Real-World Evaluation across Four Configurations.**

To evaluate the effectiveness of RoboTwin 2.0 in enhancing real-world policy robustness, we conduct controlled experiments on four bimanual manipulation tasks: *Stack Bowls*, *Handover Block*, *Pick Bottle*, and *Click Bell*. The experiments use RDT as the policy backbone and are executed on the COBOT-Magic dual-arm robotic platform. For each task, we compare three training settings: (1) a baseline trained on 10 real-world demonstrations collected in clean tabletop environments; (2) the same real-world data augmented with 1,000 domain-randomized synthetic trajectories generated by RoboTwin 2.0 under cluttered scenes with diverse lighting and backgrounds; and (3) a synthetic-only setting trained solely on the 1,000 domain-randomized trajectories. To further enhance robustness to real-world jitter and calibration errors, we apply random 3D spatial perturbations to simulated object and robot poses, with the total displacement magnitude bounded by 1cm.

For evaluation, we define four test configurations: (i) clean tabletop with seen backgrounds (same as used in the 10 real-world demos); (ii) clean tabletop with unseen backgrounds; (iii) cluttered tabletop

with seen backgrounds; and (iv) cluttered tabletop with unseen backgrounds, as illustrated in Fig.11. Notably, the synthetic-only (zero-shot) setting does not include seen backgrounds during training, so corresponding entries in the success rate table are left blank. This setup allows us to assess whether RoboTwin 2.0 can enable robust policy generalization without requiring additional real-world data from visually complex environments. Results are shown in Table 5.

Table 5: **Real-World Experiment Results.** We conduct controlled experiments on 4 dual-arm tasks: *Bowls Stack*, *Block Handover*, *Bottle Pick*, and *Bell Click*, each evaluated under 4 different settings.

Real World Task	Background Type	Cluttered or Not	10 Clean Real	10 Clean Real + 1k RoboTwin 2.0	1k RoboTwin 2.0 (Zero-shot)
Stack Bowls	Seen	False	22.0%	**64.0%**	/
		True	12.0%	**58.0%**	/
	Unseen	False	10.0%	50.0%	**60.0%**
		True	12.0%	**56.0%**	52.0%
Handover Block	Seen	False	40.0%	**48.0%**	/
		True	**16.0%**	12.0%	/
	Unseen	False	36.0%	**56.0%**	**56.0%**
		True	0.0%	**36.0%**	20.0%
Pick Bottle	Seen	False	20.0%	**36.0%**	/
		True	8.0%	**40.0%**	/
	Unseen	False	4.0%	**26.0%**	10.0%
		True	8.0%	28.0%	**32.0%**
Click Bell	Seen	False	**36.0%**	24.0%	/
		True	20.0%	**56.0%**	/
	Unseen	False	12.0%	**24.0%**	20.0%
		True	16.0%	**48.0%**	14.0%
Average	Seen	False	29.5%	**43.0%** $_{+13.5\%}$	/
		True	14.0%	**41.5%** $_{+27.5\%}$	/
	Unseen	False	15.5%	**39.0%** $_{+23.5\%}$	36.5% $_{+21.0\%}$
		True	9.0%	**42.0%** $_{+33.0\%}$	29.5% $_{+20.5\%}$

The experimental results show that real-world bimanual policies augmented with RoboTwin 2.0 data achieve substantial improvements across all evaluation settings. When adding 1,000 domain-randomized synthetic trajectories from RoboTwin 2.0, the success rates increased by 13.5%, 27.5%, 23.5%, and 33.0% across the four test configurations, respectively. For the zero-shot setting trained solely on synthetic data, we observe notable improvements of 21.0% and 20.5% on the two unseen-background scenarios. Notably, performance gains tend to increase with scene complexity, indicating that RoboTwin 2.0 enhances policy robustness under challenging visual conditions.

These gains can be attributed to two key factors. First, the visual and physical fidelity of RoboTwin 2.0 closely matches real-world conditions, enabling strong sim-to-real transfer. Second, even when real-world training data lacks such complexity, domain-randomized synthetic data effectively prepares the policy to handle challenging environments.

Interestingly, we also observe that models trained with RoboTwin 2.0 data exhibit consistent average success rates across difficulty levels in all four tasks. This suggests that the learned policies generalize well to different environments and are capable of selectively attending to task-relevant objects, further validating the effectiveness of training with RoboTwin 2.0.

4.5 RoboTwin 2.0 Benchmark

To evaluate the benchmarking utility and generalization challenges introduced by RoboTwin 2.0, we assess five policy models: ACT, DP, RDT, Pi0, and DP3. All evaluations are conducted on the full set of 50 benchmark tasks using the Aloha AgileX dual-arm embodiment. For each task, we collect 50 expert demonstrations in a clean simulation environment without domain randomization, which are used for training. During evaluation, each policy is tested with 100 rollouts under two conditions: a clean setting (Easy) and a domain-randomized setting (Hard). The Hard setting introduces cluttered scenes, random lighting, background textures, and varying tabletop heights. Detailed configuration

settings are provided in Appendix. D. We report success rates as indicators of few-shot adaptation capability and robustness to challenging, randomized variations. Full benchmark results can be found at http://robotwin-platform.github.io/leaderboard/.

Table 6: **RoboTwin 2.0 Simulation Benchmark (13 tasks sampled).**

Simulation Task	ACT		DP		DP3		RDT		Pi0	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
Grab Roller	66.0%	6.0%	**98.0%**	1.0%	77.0%	1.0%	74.0%	43.0%	96%	**80.0%**
Handover Mic	9.0%	0.0%	53.0%	0.0%	93.0%	7.0%	98.0%	**41.0%**	**100%**	13.0%
Lift Pot	7.0%	2.0%	37.0%	0.0%	**85.0%**	0.0%	82.0%	19.0%	84%	**36.0%**
Move Can Pot	0.0%	0.0%	42.0%	0.0%	28.0%	0.0%	47.0%	23.0%	**74.0%**	**32.0%**
Open Laptop	32.0%	0.0%	46.0%	0.0%	54.0%	3.0%	61.0%	36.0%	**85.0%**	**46.0%**
Pick Dual Bottles	4.0%	0.0%	26.0%	0.0%	25.0%	0.0%	42.0%	**13.0%**	**57.0%**	12.0%
Place Object Basket	0.0%	0.0%	17.0%	0.0%	2.0%	0.0%	42.0%	**14.0%**	**62.0%**	10.0%
Place Dual Shoes	0.0%	0.0%	7.0%	0.0%	1.0%	0.0%	4.0%	**4.0%**	**15.0%**	0.0%
Place Phone Stand	0.0%	0.0%	17.0%	0.0%	8.0%	1.0%	15.0%	6.0%	**35.0%**	**7.0%**
Put Bottles Dustbin	0.0%	0.0%	23.0%	0.0%	0.0%	0.0%	21.0%	4.0%	**54.0%**	**13.0%**
Put Object Cabinet	4.0%	18.0%	50.0%	17.0%	40.0%	16.0%	30.0%	**30.0%**	**69.0%**	29.0%
Stack Blocks Two	0.0%	0.0%	6.0%	0.0%	0.0%	0.0%	32.0%	**1.0%**	**40.0%**	**1.0%**
Stack Bowls Two	0.0%	0.0%	34.0%	0.0%	0.0%	0.0%	**73.0%**	21.0%	**73.0%**	**41.0%**
Average	9.4%	2.0%	35.1%	1.4%	31.8%	2.2%	47.8%	19.6%	**64.9%**	**24.6%**

Fig. 6 presents evaluation results on 13 representative tasks. As shown, non-pretrained models such as ACT, DP, and DP3 perform poorly under Hard conditions, with notably low success rates. In contrast, pretrained models like RDT and Pi0 show stronger resilience, suggesting that their vision-language-action pretraining provides useful priors for generalization to unseen scenarios. However, success rates still drop by 28.2% for RDT and 40.3% for Pi0 when transitioning from clean to randomized settings, indicating that robustness under heavy domain shifts remains a key challenge. This degradation likely stems from insufficient diversity in pretraining data, despite its large scale. Combined with the results from Figures 4.4 and 4.3, these findings highlight RoboTwin 2.0's potential to complement existing pretraining datasets by providing diverse, domain-randomized trajectories that improve generalization and robustness under real-world variations.

5 Related Work

5.1 Datasets and Benchmarks for Robotic Manipulation

Physics-based simulators form the backbone of modern manipulation research. SAPIEN [40] supports fully dynamic interaction with over 2,300 articulated objects from PartNet-Mobility [40, 28], ManiSkill2 supplies more than four million demonstration frames across twenty task families [14], Meta-World [42] offers fifty standardized manipulation tasks for multi-task and meta-reinforcement learning, CALVIN pairs long-horizon, language-conditioned instructions with rich sensor suites [27], LIBERO [25] defines 130 lifelong-learning tasks with high-quality human teleoperation data, and RoboVerse [13] unifies multiple simulators and embodiments under a common benchmark with domain randomization.

In parallel, large-scale real-world datasets are essential to bridge simulation and reality. AgiBot World [4] provides over one million human-verified trajectories spanning 217 tasks in diverse deployment scenarios, RoboMIND [39] contributes 107 K teleoperated episodes (including annotated failures) across 479 tasks and four robot platforms, Open X-Embodiment [31] consolidates more than one million demonstrations from twenty-two robot embodiments into a unified format, and

Bridge [11] delivers 60 K+ trajectories of 13 skills on low-cost hardware to evaluate transfer under varying visual and physical conditions.

RoboTwin-1.0 [30] introduced a bidirectional twin framework that mirrors real teleoperated demonstrations with AI-generated simulated replicas for unified evaluation of dual-arm manipulation. In this work, we extend RoboTwin by integrating interactive LLM-driven feedback into the simulation loop and applying systematic domain randomization over visual, physical, and task parameters. These enhancements yield richer, more diverse training corpora and significantly improve policy robustness and generalization. A detailed comparison between RoboTwin 2.0 and prior datasets and benchmarks is provided in Appendix B.

5.2 Robot Learning in Manipulation

Many task-specific policy architectures [36, 17, 43, 8, 12, 7, 24, 36, 22, 23, 38, 37] achieve strong single-task performance but struggle to transfer across embodiments. In contrast, foundation models trained on million-scale, multi-robot corpora have enabled robust zero-shot generalization: RT-1 [3] unifies vision, language and actions in a single transformer for real-time kitchen tasks; RT-2 [2] co-fine-tunes large vision–language models on web and robot data to unlock semantic planning and object reasoning; diffusion-based RDT-1B [26] and the π_0[1] capture diverse bimanual dynamics from over a million episodes. Vision–language–action (VLA) frameworks like OpenVLA [19] and CogACT [21], together with adaptations like Octo [34], LAPA [41], and OpenVLA-OFT [18] demonstrate efficient fine-tuning to novel robots and sensor modalities.

To further advance this direction, our work introduces digital-twin data collection paired with extensive domain randomization, yielding datasets that closely mirror real robot dynamics and support the training of robust and generalizable bi-manual manipulation policies.

5.3 Domain Randomization in Imitation Learning

Prior works have shown that randomizing visual and physical parameters, including but not limited to textures, lighting, camera pose, mass, friction and control latency combined with noise injection in expert demonstrations, enables sim-to-real transfer and robust visuomotor policies [35, 32, 5, 7, 23], and optimizing over worst-case ensembles further improves resilience to extreme domain shifts [33, 20, 22]. However, these approaches apply randomization in isolation and lack bidirectional digital-twin feedback; our method integrates interactive simulation feedback with systematic domain randomization to generate higher-fidelity imitation data.

6 Conclusion

This paper presented RoboTwin 2.0, a scalable simulation framework for generating diverse, high-fidelity expert data to support robust bimanual manipulation. Our system integrates MLLM-based task generation, embodiment-adaptive behavior synthesis, and comprehensive domain randomization to address key limitations in prior synthetic datasets.

By leveraging an annotated object library and automating trajectory generation, RoboTwin 2.0 produces data with rich visual, linguistic, and physical diversity while minimizing manual engineering effort. Experiments demonstrate its effectiveness in improving policy robustness to cluttered environments, generalization to unseen tasks, and cross-embodiment manipulation.

These findings highlight the importance of scalable, automated generation of semantically rich, domain-randomized data for learning robust manipulation policies. RoboTwin 2.0 provides a foundation for unified benchmarks and scalable sim-to-real pipelines, with future work focusing on real-world deployment and multi-object task complexity.

7 Acknowledgments

This paper is partially supported by AgileX Robotics, D-Robotics, and the Jockey Club STEM Lab of Autonomous Intelligent Systems funded by The Hong Kong Jockey Club Charities Trust.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[4] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[5] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[6] Junting Chen, Yao Mu, Qiaojun Yu, Tianming Wei, Silang Wu, Zhecheng Yuan, Zhixuan Liang, Chao Yang, Kaipeng Zhang, Wenqi Shao, Yu Qiao, Huazhe Xu, Mingyu Ding, and Ping Luo. Roboscript: Code generation for free-form manipulation tasks across real and simulation, 2024.

[7] Tianxing Chen, Yao Mu, Zhixuan Liang, Zanxin Chen, Shijia Peng, Qiangyu Chen, Mingkun Xu, Ruizhen Hu, Hongyuan Zhang, Xuelong Li, et al. G3flow: Generative 3d semantic flow for pose-aware and generalizable object manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1735–1744, 2025.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.

[10] Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Heming Cui, et al. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. *arXiv preprint arXiv:2505.03233*, 2025.

[11] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[12] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[13] Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Charlie Tianyue Cheng, Haozhe Lou, Peihao Li, Yen-Jen Wang, et al. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning. *arXiv preprint arXiv:2504.18904*, 2025.

[14] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. *arXiv preprint arXiv:2408.09559*, 2024.

[16] Mengkang Hu, Tianxing Chen, Yude Zou, Yuheng Lei, Qiguang Chen, Ming Li, Yao Mu, Hongyuan Zhang, Wenqi Shao, and Ping Luo. Text2world: Benchmarking large language models for symbolic world model generation. *arXiv preprint arXiv:2502.13092*, 2025.

[17] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[18] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[19] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*.

[20] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

[21] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[22] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In *International Conference on Machine Learning*, pages 20725–20745. PMLR, 2023.

[23] Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16467–16476, 2024.

[24] Zhixuan Liang, Yao Mu, Yixiao Wang, Tianxing Chen, Wenqi Shao, Wei Zhan, Masayoshi Tomizuka, Ping Luo, and Mingyu Ding. Dexhanddiff: Interaction-aware diffusion planning for adaptive dexterous manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1745–1755, 2025.

[25] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[26] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[27] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[28] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.

[29] Yao Mu, Junting Chen, Qing-Long Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, et al. Robocodex: Multimodal code generation for robotic behavior synthesis. In *International Conference on Machine Learning*, pages 36434–36454. PMLR, 2024.

[30] Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2025.

[31] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[32] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[33] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations*, 2017.

[34] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[35] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[36] Chenxi Wang, Hongjie Fang, Hao-Shu Fang, and Cewu Lu. Rise: 3d perception makes real-world robot imitation simple and effective. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2870–2877. IEEE, 2024.

[37] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[38] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025.

[39] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

[40] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[41] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *CoRL 2024 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*.

[42] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[43] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy. *arXiv e-prints*, pages arXiv–2403, 2024.

[44] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A Contributions

Project Leaders
Tianxing Chen, Yao Mu, Zhixuan Liang

Roadmap & Methodology
Yao Mu, Tianxing Chen, Ping Luo, Yusen Qin, Xiaokang Yang, Kaixuan Wang

Data Generator & Benchmark
Tianxing Chen, Zanxin Chen, Baijun Chen, Qiwei Liang, Zixuan Li, Xianliang Lin

CodeGen Agent
Yibin Liu, Zanxin Chen, Yiheng Ge, Tianxing Chen, Mengkang Hu

RoboTwin-OD
Baijun Chen, Qiangyu Chen, Kailun Su, Xuanbing Xie, Zanxin Chen

Policies Training & Evaluation
Tianxing Chen, Zijian Cai, Tian Nian, Huan-ang Gao, Tianling Xu

Real-World Deployment
Tianxing Chen, Tian Nian, Weiliang Deng

Domain Randomization
Baijun Chen, Yubin Guo, Qiwei Liang, Zhenyu Gu, Guodong Liu, Zanxin Chen, Tianxing Chen

B Benchmarking RoboTwin 2.0 Against Existing Datasets

We compare RoboTwin 2.0 against existing benchmarks and datasets across several key dimensions, including the number of supported tasks, the presence of domain randomization, support for automatic data generation, and compatibility with vision-language-action (VLA) model training and evaluation. The comparison is summarized in Table 7.

Table 7: **Comparison of RoboTwin 2.0 with previous manipulation benchmarks and datasets.**

Benchmark & Dataset	#Tasks	Domain Randomization	Auto Data Generation	VLA Model Train & Eval
Meta-world [42]	50	×	✓	×
Robosuite [44]	9	×	×	×
RoboCasa [42]	25	✓	×	×
Maniskill2 [14]	20	×	✓	×
AutoBio [44]	16	×	✓	✓
RoboTwin 1.0 [30]	14	×	✓	✓
RoboTwin 2.0 (ours)	50	✓	✓	✓

C Domain Randomization Setting

Domain randomization in all experiments includes cluttered scenes, random lighting, table height variation (up to 3 cm), and randomized background textures.

D Policies Training Details

RDT was pretrained for 100,000 steps with a batch size of 16 per GPU on 8 GPUs, and subsequently fine-tuned for 20,000 steps with a batch size of 16 per GPU on 4 GPUs.

Pi0 was pretrained for 100,000 steps with a batch size of 32, and fine-tuned for 30,000 steps using the same batch size.

For all ACT-based policies, we use a unified training setup with a chunk size of 50, batch size of 8, and single-GPU training for 6000 steps. During deployment, we apply `temporal_agg` for temporal aggregation to enhance execution stability.

DP was trained for 600 steps with a batch size of 128 and a planning horizon of 8.

DP3 was trained for 3,000 steps with a batch size of 256, using a planning horizon of 8 and a point cloud resolution of 1,024.

D.1 Task-Specific Performance Comparison on Code Generation

We compare the code generation success rates of RoboTwin 2.0 and RoboTwin 1.0 across all tasks. As shown, RoboTwin 2.0 consistently matches or outperforms the baseline on the majority of tasks, demonstrating the effectiveness of our multimodal feedback and refinement pipeline.

Task	R1.0 Vanilla	R1.0 + FB	R1.0 + MM FB	R2.0 Vanilla	R2.0 + FB	R2.0 + MMFB
beat_block_hammer	16%	48%	**56%**	23%	34%	53%
handover_block	2%	41%	45%	17%	**50%**	27%
pick_diverse_bottles	**65%**	**65%**	64%	60%	60%	62%
pick_dual_bottles_easy	99%	99%	**100%**	**100%**	**100%**	**100%**
place_container_plate	66%	79%	**91%**	84%	84%	82%
place_dual_shoes	19%	22%	**25%**	0%	2%	22%
place_empty_cup	90%	90%	**100%**	61%	61%	85%
place_shoe	72%	90%	90%	**100%**	**100%**	**100%**
stack_blocks_three	1%	2%	4%	76%	76%	**82%**
stack_blocks_two	44%	68%	64%	**100%**	**100%**	**100%**

Table 8: Task-Specific Performance Comparison between RoboTwin 2.0 and RoboTwin 1.0. R1.0/R2.0: RoboTwin 1.0 / 2.0. Bold numbers indicate the best result for each task.

E Prompts for Generating Task Instructions and Object Descriptions

```
# Task Instruction Template
- Goal: Generate task instruction template
- Requirements:
  - Generate 60 items. Vary in sentence length and structure
  - Use natural action verbs (grab, slide, place)
- split
  - 50 items for training
  - 10 items for evaluation

## Schema Requirements
- Goal: Use placeholders for objects in instructions
- Requirements:
  - Format: {X} for objects defined in schema
  - Include all object placeholders ({A-Z}) in every instruction
  - Omit arm references and placeholders ({a-z}) in 50% of instructions
  - Ensure natural flow when placeholders are replaced with actual values

# Object Description
- Goal: Generate natural object descriptions for robotic manipulation
- Requirements:
  - Generate 15 items. Vary in sentence length and structure
  - Use natural oral language
  - Include essential physical properties (color, shape, size, texture)
  - Use noun-focused phrases
  - For multi-part objects, use structures like 'X with Y'
- split
  - 12 items for training
  - 3 items for evaluation

# Episode
An episode is a specified task, in which each task may have different objects to be
    manipulated,
resulting in the same task template being reused by replacing the placeholders with
    specific objects.
For example:
  {A} -> 'medium-sized yellow bottle'
  {A} -> 'green drink bottle with bold labels'

General Task -> Specific Episode:
  {A} -> bottle/0.glb
  {A} -> bottle/1.glb

The number of task instructions for an episode can be calculated by:
  Episode_num = TaskInstruction_num * ObjectDescription_num
```

Listing 1: Prompts for Generating Task Instructions and Object Descriptions

F Task Instruction and Object Description Example

Instruction Templates (task: 'Pick Dual Bottles')

```
"Use {a} to place {A} left of {B}.", "Set {A} to the left of {B}.", "Move {A} beside {B} using
{a}.", "Place {A} on {B}'s left side.", "Using {a}, position {A} next to {B}.", "Stick {A} on the
left of {B}.", "Use {a} and place {A} on {B}'s left.", etc
```

Object Description

```
# object id - '001_bottle/0':
"red bottle", "red soda bottle", "plastic red bottle", "red bottle with yellow label", "red
plastic bottle with smooth surface", "yellow text printed on red bottle surface", "red bottle with
white label design and markings", "red bottle with white sealing and brown top screw cap", etc
# object id - '039_mug/0':
"black mug", "dark coffee mug", "sleek black mug", "black ceramic mug", "single-handle mug",
"smooth black surface mug", "medium-sized drinking mug", "round mug with curved side", "dark mug
with sturdy handle", "solid black mug with smooth finish", etc
```

G Experimental Details and Metric Definitions for Code Generation

We use the *DeepSeek-V3* model for program synthesis and the *moonshot-v1-32k-vision-preview* model for multimodal error localization and verification. These models were selected for their strong performance in language reasoning and visual understanding while maintaining efficiency suitable for large-scale iterative refinement. The success rate of the i-th program is computed as $R_i = \frac{1}{M}\sum_{j=1}^{M} s_{i,j}$, and the final success rate for a given task under a specific system variant is then defined as $R_{\text{task}} = \frac{1}{N}\sum_{i=1}^{N} R_i$.

G.1 Metric Definitions

We report the following metrics across all tasks:

ASR (Average Success Rate) is the average of R_task across all 10 tasks. It reflects overall task performance across all generated programs.

Top5-ASR is the mean success rate computed using only the top 5 highest-performing programs per task. This metric estimates system potential under a best-of-selection strategy.

CRSR (Code Revision Success Rate) measures the final success rate of each task after up to five rounds of code repair, incorporating either execution feedback (FB) or multimodal feedback (MM FB).

CR-Iter indicates the average number of feedback iterations required per task before reaching a success rate above 50% or exhausting the iteration budget.

Token denotes the average number of tokens generated by the language model per task, including both initial synthesis and all revision steps. It serves as a proxy for computational cost and LLM inference budget.

These metrics jointly evaluate both the reliability and efficiency of the expert data generation pipeline under varying conditions of feedback, model capability, and refinement strategy.

H Success Rates of Different Embodiments on RoboTwin 2.0 Tasks

We report the success rates of five robot embodiments across the 50 RoboTwin 2.0 tasks, using the same set of expert code for data generation, as shown in Table 9.

Table 9: **Success Rates of Different Embodiments on RoboTwin 2.0 Tasks.**

	RoboTwin1.0					RoboTwin2.0				
Task Name	**Aloha**	**ARX**	**Franka**	**Piper**	**UR5**	**Aloha**	**ARX**	**Franka**	**Piper**	**UR5**
Adjust Bottle	92%	88%	39%	0%	7%	93%	94%	34%	0%	12%
Beat Block Hammer	68%	86%	95%	0%	86%	64%	93%	98%	15%	90%
Blocks Ranking Rgb	92%	98%	96%	0%	82%	96%	97%	99%	13%	53%
Blocks Ranking Size	90%	95%	92%	0%	60%	96%	97%	89%	7%	38%
Click Alarmclock	89%	99%	100%	0%	95%	92%	99%	100%	0%	95%
Click Bell	100%	100%	100%	9%	100%	100%	100%	100%	91%	100%
Dump Bin Bigbin	85%	98%	90%	0%	82%	84%	100%	84%	9%	80%
Grab Roller	95%	69%	99%	0%	80%	95%	69%	99%	7%	81%
Handover Block	1%	3%	0%	0%	4%	83%	81%	0%	44%	0%
Handover Mic	62%	80%	92%	28%	0%	87%	98%	84%	65%	14%
Hanging Mug	68%	76%	5%	0%	12%	63%	73%	11%	0%	11%
Lift Pot	27%	50%	24%	5%	40%	27%	50%	36%	31%	40%
Move Can Pot	18%	0%	37%	2%	4%	93%	65%	92%	96%	99%
Move Pillbottle Pad	30%	52%	15%	0%	35%	67%	90%	69%	47%	86%
Move Playingcard Away	93%	100%	100%	0%	87%	99%	100%	100%	63%	66%
Move Stapler Pad	94%	92%	88%	0%	95%	92%	96%	89%	13%	75%
Open Laptop	76%	91%	78%	14%	55%	82%	92%	77%	23%	51%
Open Microwave	65%	85%	75%	5%	33%	96%	80%	59%	2%	23%
Pick Diverse Bottles	11%	1%	0%	0%	0%	51%	2%	0%	27%	4%
Pick Dual Bottles Easy	8%	3%	0%	0%	0%	92%	6%	0%	81%	7%
Place A2B Left	65%	75%	70%	0%	72%	80%	88%	64%	29%	76%
Place A2B Right	70%	68%	68%	0%	69%	81%	82%	64%	31%	66%
Place Bread Basket	91%	91%	69%	0%	78%	89%	88%	62%	1%	67%
Place Bread Skillet	31%	28%	42%	0%	42%	34%	26%	42%	0%	37%
Place Can Basket	47%	1%	38%	0%	11%	70%	28%	61%	0%	3%
Place Cans Plasticbox	96%	93%	98%	0%	11%	100%	96%	85%	0%	82%
Place Container Plate	86%	85%	83%	0%	82%	89%	86%	86%	37%	81%
Place Dual Shoes	73%	28%	36%	0%	40%	77%	31%	41%	1%	32%
Place Empty Cup	92%	100%	100%	0%	100%	92%	100%	100%	4%	100%
Place Fan	93%	96%	75%	0%	85%	95%	93%	83%	0%	65%
Place Burger Fries	96%	95%	85%	0%	78%	97%	98%	80%	36%	74%
Place Mouse Pad	100%	80%	99%	2%	96%	99%	89%	100%	23%	73%
Place Object Basket	68%	13%	68%	0%	30%	74%	14%	61%	0%	7%
Place Object Scale	77%	93%	94%	0%	87%	78%	92%	82%	2%	76%
Place Object Stand	90%	92%	81%	0%	90%	97%	99%	81%	9%	92%
Place Phone Stand	66%	78%	52%	22%	44%	66%	78%	45%	53%	49%
Place Shoe	87%	85%	70%	0%	97%	84%	85%	74%	7%	91%
Press Stapler	87%	96%	99%	0%	77%	98%	96%	100%	59%	72%
Put Bottles Dustbin	0%	0%	0%	0%	0%	71%	1%	0%	56%	0%
Put Object Cabinet	13%	56%	43%	0%	0%	14%	24%	55%	0%	0%
Rotate Qrcode	78%	83%	98%	0%	81%	75%	74%	94%	0%	67%
Scan Object	8%	13%	21%	0%	8%	4%	45%	26%	0%	19%
Shake Bottle	62%	95%	82%	1%	98%	89%	94%	85%	74%	97%
Shake Bottle Horizontally	64%	93%	81%	1%	97%	90%	94%	85%	74%	98%
Stack Blocks Three	98%	97%	95%	0%	83%	94%	96%	80%	0%	51%
Stack Blocks Two	99%	99%	100%	0%	94%	98%	99%	96%	2%	68%
Stack Bowls Three	27%	64%	76%	0%	76%	43%	58%	82%	0%	81%
Stack Bowls Two	63%	84%	88%	0%	94%	78%	82%	88%	4%	94%
Stamp Seal	46%	91%	95%	0%	100%	56%	91%	4%	37%	100%
Turn Switch	27%	3%	51%	28%	10%	74%	3%	36%	81%	10%
Average	**65.3%**	**68.8%**	**67.6%**	**2.3%**	**57.7%**	**78.8%**	**74.2%**	**67.2%**	**25.1%**	**57.1%**
Difference	/	/	/	/	/	**+13.5%**	**+5.4%**	**-0.4%**	**+22.8%**	**-0.6%**