REAL2GYM: BUILDING GYMS FROM VIDEOS, BRINGING SKILLS TO ROBOTS

Anonymous authors
Paper under double-blind review



Figure 1: Real2Gym converts human or robot videos into visually aligned and physically executable Blender and MuJoCo environments, where reconstructed interactions are validated under native physics and expanded into feasible task variations. Rather than treating simulation as the endpoint, the agent uses these gyms to execute, fail, reflect, and accumulate reusable skills. The resulting skills are re-grounded from human demonstrations and transferred to real robots through a shared control interface, enabling Real2Sim2Real self-improvement.

ABSTRACT

Real-world videos provide rich demonstrations of manipulation, but turning them into reusable robot skills requires visually aligned environments, executable physical interactions, and mechanisms for learning from experience. We introduce Real2Gym, an agentic Real2Sim2Real framework that turns human and robot demonstrations into interactive simulation gyms and brings skills acquired in simulation to physical robots. The Real2Sim module reconstructs editable scenes, aligns objects and cameras with the input, validates demonstrated or retargeted actions through native physics execution, and generates task-conditioned variations with action-feasibility checks. Within these environments, the agent generates executable code for manipulation stages, observes their outcomes, and distills successful attempts and failures into reusable task procedures, object-relative motions, and recovery strategies. Through a shared perception-and-control interface, these skills guide subsequent execution in simulation and on real robots, with motions adapted to current observations and no updates to the underlying model weights. Extensive evaluations demonstrate that Real2Gym enables high-fidelity simulation environment reconstruction, outperforming GPT-6 Astra Direct Mode by 16.7% in success rate with approximately 74.9% fewer policy-execution tokens across these environments, while exceeding it by 33.3% in physical robot execution success rate across four tasks on a real Franka robot.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Robot self-improvement offers a promising pathway toward embodied intelligence: through iterative interaction, robots can diagnose execution failures, test policy adjustments, and accumulate experience to refine subsequent behavior (Kober et al., 2013). Recent robotic agents highlight how execution feedback and reusable skills drive this process (Lu et al., 2026; Jia et al., 2026). Despite this progress, deploying this trial-and-error paradigm directly on physical hardware remains prohibitively costly. Specifically, physical motion and scene resets are time-consuming, failed attempts risk damaging the manipulator or surrounding environment, and constant trials demand continuous supervision while accelerating hardware wear (Dulac-Arnold et al., 2019). Furthermore, limited physical setups restrict parallel exploration, and reliably restoring initial physical states after a failure is often difficult (Eysenbach et al., 2017). Consequently, scaling self-improvement entirely in the real world is bottlenecked by high operational costs, safety risks, and low sample efficiency.

To overcome these real-world bottlenecks, simulation offers a scalable setting for iterative trial-and-error before physical deployment (Andrychowicz et al., 2020; Tao et al., 2024). However, successful Real2Sim2Real transfer hinges on faithfully reproducing the target scene's task-relevant geometry, object articulations, and physical interactions (Zhao et al., 2020). Prior work addresses this transfer through visual randomization (Tobin et al., 2017) and physics calibration (Tan et al., 2018). Existing pipelines, such as RialTo (Torne et al., 2024), construct these environments through a cumbersome workflow spanning scene scanning, mesh repair, articulation modeling, and physics parameterization. Heavily constrained by manual intervention and disjointed tools, building interactive simulation environments for new tasks remains prohibitively labor-intensive.

Addressing this heavy manual overhead, recent progress in visual geometry estimation (Wang et al., 2024; 2025) and generative simulation (Wang et al., 2023b) has accelerated automated scene construction. For instance, Agentic Real2Sim converts recordings of robot–object interactions into executable episodic twins (Chen et al., 2026), while state-of-the-art multimodal models like GPT-6 Astra can synthesize detailed Blender scenes from visual inputs. However, our comparisons show that GPT-6 Astra reconstructions can still contain inaccurate object dimensions, relative spatial layouts, and camera extrinsics (Fig. 3). Errors in scale and placement distort grasp clearances and contact geometry, while camera misalignment obscures correspondence with the demonstrated motion. During fine-grained manipulation, these spatial inaccuracies cause interpenetration, missed contacts, and failed grasps, preventing the reconstructed scene from executing reliably under native physics. Consequently, such physical invalidity compromises the fidelity required for downstream agent exploration and skill accumulation.

We resolve these physical and geometric fidelity gaps with Real2Gym, a unified Real2Sim2Real framework that couples physics-verified environment synthesis with agentic skill accumulation. Given a human or robot demonstration video, our Real2Sim pipeline constructs visually aligned Blender and MuJoCo environments, rigorously verifying physical interaction feasibility under native physics. Validated scenes are then augmented with action-adapted procedural variations to build diverse, execution-ready gyms for agent exploration. Within these environments, the agent operates across high-level manipulation stages, distilling execution feedback into reusable, object-relative skills that adapt to current observations without model weight updates. Across benchmark scenes reconstructed from public datasets, Real2Gym markedly improves reconstruction quality. Its skill-guided agent achieves an 87.5% success rate compared to 70.8% for GPT-6 Astra Direct Mode, while consuming approximately 74.9% fewer policy-execution tokens (Tables 1 and 2). Finally, real-robot deployments confirm that simulation-refined skills enable successful physical execution on tasks that initially failed, validating our end-to-end pipeline.

Our main contributions are summarized as follows:

- We introduce a Real2Sim pipeline that constructs *executable* digital twins from human and robot demonstrations through event-centered correction, native-physics validation, and task-conditioned augmentation with action-feasibility checks.
- We introduce an experience-driven manipulation agent that integrates perception tools, executable operation stages, and feedback-driven skill extraction to support efficient interaction and skill reuse in simulation and on physical robots.

- Across 24 reconstructed environments, Real2Gym delivers consistent gains in reconstruction fidelity, task success, and token efficiency across both DROID and EgoDex, while real-robot experiments validate closed-loop Real2Sim2Real self-improvement.

2 Related Work

2.1 Real-to-Simulation Reconstruction

Real-to-simulation reconstruction builds interactive digital twins from real observations and connects them to physics engines for robot policy training, data generation, and evaluation. RialTo (Torne et al., 2024) constructs digital twins of real environments and uses reinforcement learning in simulation to improve manipulation policies before transferring them back to physical robots.

Building on 3D Gaussian Splatting (Kerbl et al., 2023), recent methods combine photorealistic observations with simulated interactions. SplatSim (Qureshi et al., 2025) uses Gaussian rendering to generate visual training data for RGB manipulation policies and demonstrates zero-shot deployment on real robots. RoboGSim (Li et al., 2024) integrates Gaussian reconstruction with a physics engine for demonstration synthesis and closed-loop policy evaluation. Splatting Physical Scenes (Moran et al., 2025) combines Gaussian appearance representations with explicit object meshes, jointly refining geometry, robot poses, and physical parameters through differentiable rendering and MuJoCo simulation. Recent agentic approaches automate the coordination of perception, modeling, and simulation tools. Agentic Real2Sim (Chen et al., 2026) uses vision-language agents to convert robot–object interaction recordings into executable episodic twins, incorporating simulator feedback into reconstruction and refinement.

2.2 Agents for Robot Control

Language-model-based robot control connects task instructions to executable behavior through perception and control interfaces. SayCan (Ahn et al., 2022) grounds language plans in the affordances of learned robot skills. Code as Policies (CaP) (Liang et al., 2023) generates programs that compose these interfaces to perform spatial reasoning and organize robot actions. Complementary approaches ground language reasoning in observations and feedback: VoxPoser (Huang et al., 2023) constructs composable 3D value maps for motion planning, while Inner Monologue (Huang et al., 2022) uses scene descriptions and execution outcomes to update task plans.

Recent coding-agent frameworks increasingly emphasize iterative execution, self-correction, and experience reuse. CaP-X (Fu et al., 2026) introduces an interactive environment and benchmark for robot programming agents, systematically investigating how multi-turn interaction, structured execution feedback, and program refinement enhance manipulation performance. ASPIRE (Lu et al., 2026) broadens this interactive loop toward persistent skill accumulation, distilling validated code repairs into a reusable skill library while employing evolutionary search to explore diverse task sequences and control programs.

In parallel, recent studies explore general-purpose models and agentic architectures as robot policies. Agent as Policy (Jia et al., 2026) unifies planning and execution within a general-purpose agent that interprets visual observations, generates programs, and adapts actions in response to physical outcomes, entirely without task-specific training. Furthermore, it demonstrates how reusing saved procedures and programs substantially reduces execution time across repeated real-robot trials.

3 Method

Fig. 2 presents the overall pipeline of Real2Gym, a unified framework that converts real-world demonstrations into interactive simulation environments and reusable manipulation skills. Given a human or robot demonstration video $\mathcal{I} = \{\mathbf{I}_t^v\}_{t,v}$ (indexed by time t and camera view v) and a robot URDF $\mathcal{U}$, our framework operates in two distinct phases: environment construction and skill accumulation. First, we transform the input video and URDF into a collection of executable simulation environments $\mathcal{G}$, where each environment combines a visually aligned Blender scene with a MuJoCo model that supports physical interaction. Second, within these reconstructed environments, an agent policy π_θ interacts with the scenes to build a reusable skill library $\mathcal{K}$. Specifically, Sec. 3.1

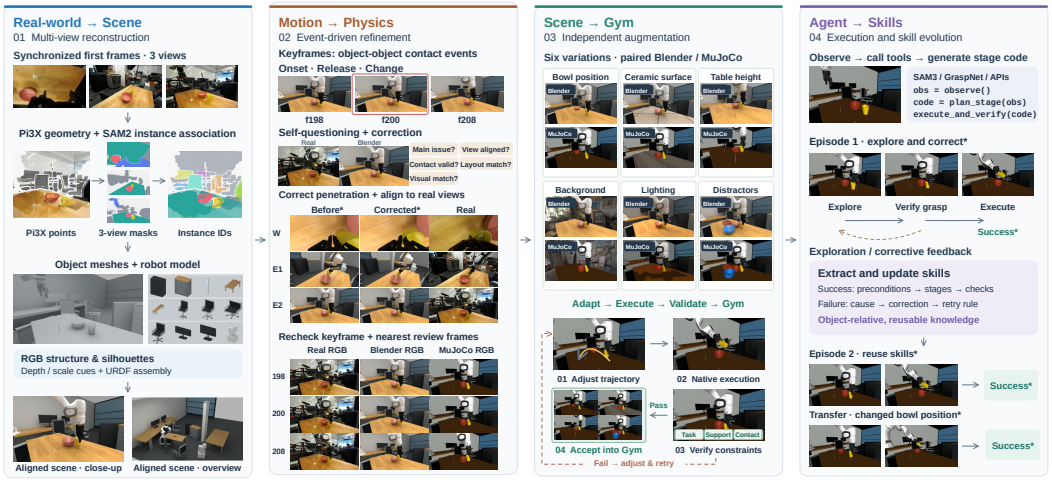


Figure 2: **Overview of Real2Gym.** Real2Gym reconstructs aligned Blender and MuJoCo scenes from human or robot demonstrations, refines them through event-driven correction and physics validation, and augments them into diverse interactive gyms. The agent then executes operation-stage code and distills feedback into reusable skills for simulation and real-robot deployment.

describes scene construction, action reconstruction, and augmentation, while Sec. 3.2 introduces the agent policy and experience-driven skill extraction.

3.1 Interactive Gym Construction

Scene Reconstruction and Alignment. We reconstruct an editable 3D scene that faithfully preserves the objects, spatial relationships, and viewpoints from the input demonstration. Scene geometry is initialized from the first frame using MoGe-3 (Kong et al., 2026) for single-view inputs or the Pi3X implementation of π^3 (Wang et al., 2026) across available views. Calibrated camera parameters and metric reference cues strictly constrain global scale and the shared coordinate frame. To parse scene contents, the agent integrates semantic reasoning with segmentation from SAM2 (Ravi et al., 2025) to identify manipulated objects, supporting surfaces, and salient background entities, maintaining consistent identity association across views. Each instance is represented as a complete mesh within a unified scene frame. While initial point clouds provide depth, orientation, and scale cues, occluded surfaces are completed using RGB silhouettes, visible structures, and up to ten sparse multi-view frames. Finally, the target robot is imported via its URDF or MJCF description, aligning its base pose, initial joint configuration, and camera mount with visual observations. Iterative reprojection checks then jointly refine object geometries, poses, and camera parameters while enforcing physical support relations and robot kinematics.

Action Reconstruction and Physical Validation. We recover demonstrated interactions via keyframes linked to topological changes in contact, grasp, support, and containment relationships. Where available, recorded joint and gripper states are directly utilized; for human demonstrations, motions are retargeted by mapping observed hand–object interactions onto the target robot. At each event keyframe across all available views, the agent performs iterative self-inspection and correction across five diagnostic dimensions: primary discrepancy, camera alignment, relative object placement, contact/penetration, and appearance fidelity. Any detected anomaly triggers temporal inspection of neighboring frames, prompting localized corrections and subsequent re-verification. Once visually aligned, the scene is instantiated in MuJoCo (Todorov et al., 2012) with articulated robot models, collision geometries, container cavities, physical material properties, and actuators. Physical execution is calibrated progressively—first adjusting approach trajectories and contact orientations, then verifying gripper closure, grasp retention, support stability, and object release. The final model is executed end-to-end from its initial state to validate physical consistency and task completion. Finally, the native physics trajectory is reimported into Blender, enabling frame-matched visual comparisons across Real RGB, Blender RGB, and MuJoCo RGB renderings.

Task-Conditioned Scene Augmentation. Starting from a validated seed environment, we systematically synthesize variations across object geometry, pose, support height, material properties, distractors, background, and lighting. Each factor is initially perturbed in isolation to evaluate its impact on task feasibility. Geometric modifications are consistently propagated across visual models, collision meshes, supporting surfaces, and robot mounting constraints. Corresponding manipulation stages are then adapted to updated grasp regions, target poses, and spatial clearances. Each candidate environment undergoes native physics execution to verify task completion, contact dynamics, support stability, and Blender–MuJoCo cross-renderer consistency. Failed candidates are routed back for scene or action refinement, while validated ones are appended to the environment pool $\mathcal{G}$. This closed-loop validation guarantees physical feasibility as environment diversity scales.

3.2 Agent Execution and Skill Accumulation

Subtask-Level Closed-Loop Execution. The agent alternates observation, code generation, execution, and feedback at the level of manipulation subtasks. At decision step k, the policy receives the task instruction g, current observations o_k, within-episode history $\mathcal{H}_k$, and explicitly selected skills $\mathcal{K}_e$:

$$c_k = \pi_\theta(g, o_k, \mathcal{H}_k, \mathcal{K}_e), \tag{1}$$

where c_k is an executable Python program for a manipulation subtask, with entry conditions, intermediate checks, and an observable completion condition. Observations comprise available camera views, robot proprioception, and permitted execution feedback. The program combines perception API calls to SAM3 (Carion et al., 2026) for object segmentation and GraspNet (Sundermeyer et al., 2021) for grasp proposals, simple numerical computations such as coordinate transformations and target-pose offsets, and robot-control commands such as *goto_pose()*, *open_gripper()*, and *close_gripper()*. Conditional checks on updated observations verify progress within the program. A single response can thus coordinate several related actions, such as approaching an object, closing the gripper, and testing grasp retention, before returning control to the policy. The executor runs the code within bounded execution segments and returns updated observations and feedback for the next decision. Decisions within an episode share one continuous context; each new episode starts with a fresh context and the selected skill inputs. Task success is assessed independently from the recorded execution under predefined criteria.

Extraction and Reuse. After an episode, a separate extraction process analyzes its observations, generated code, execution feedback, and final outcome. Each code round is assigned a positive, negative, or unknown local effect, allowing successful substeps and unsuccessful approaches to be distinguished within the same episode. Failure analysis compares unsuccessful attempts with subsequent corrections when both are supported by the execution record. The resulting skills contain applicability conditions, task procedures, effect checks, object-relative motion rules, and recovery guidance. Motion rules specify the acting entity, an object anchor, a relative position or orientation, and a stopping condition. During execution, these rules are instantiated using object poses estimated from current observations, allowing the same skill to adapt to changes in object placement. New lessons are merged with the existing library, retaining supporting evidence and revising contradicted guidance. Their utility is tested through fresh execution. The same representation supports real-robot deployment by grounding the selected skills in live observations through the corresponding perception-and-control interface. Experience accumulation updates the skill library while keeping the underlying model parameters θ fixed.

4 Experiments

4.1 Experimental Setup

Datasets. We select 12 scenes from DROID (Khazatsky et al., 2024) and 12 scenes from EgoDex (Hoque et al., 2026) for Real2Sim pipeline evaluation. DROID provides robot demonstrations collected in everyday real-world environments, including RGB videos from both external and wrist-mounted cameras together with synchronized robot trajectories, while EgoDex provides egocentric human manipulation videos with 3D hand and finger tracking. Each dataset contributes four easy, four medium, and four hard scenes. The 24 MuJoCo simulation environments reconstructed by our pipeline subsequently serve as the evaluation environments for agents.

Table 1: **Quantitative comparison of Real2Sim reconstruction.** Results are averaged over 12 DROID and 12 EgoDex scenes. **Bold** and underlined denote best and second-best results.

	Method	Content alignment↑	Viewpoint alignment↑	Action fidelity↑	Simulation success score↑
DROID	GPT-5.6 Sol xhigh	50.00	24.17	63.83	48.96
	GPT-6 Astra Medium	55.00	30.00	76.08	71.08
	Ours	**66.25**	**70.83**	**83.17**	**80.88**
EgoDex	GPT-5.6 Sol xhigh	55.42	34.17	54.83	46.77
	GPT-6 Astra Medium	57.58	38.67	65.75	56.59
	Ours	**75.08**	**73.75**	**86.25**	**85.76**

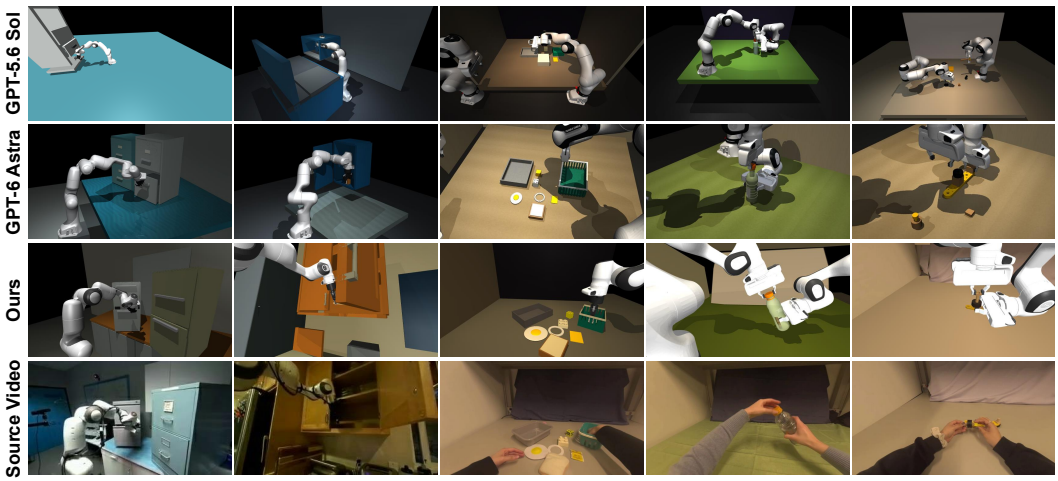


Figure 3: **Qualitative comparison of Real2Sim reconstruction.** Five manipulation scenes from DROID and EgoDex are shown. Rows present GPT-5.6 Sol, GPT-6 Astra, Real2Gym, and the source video, from top to bottom.

Metrics. For Real2Sim evaluation, GPT-6 Astra with high reasoning effort scores content alignment, viewpoint alignment, action fidelity, and simulation success score using the prompts in Appendix E. Collectively, these metrics quantify visual correspondence with the source demonstration alongside the physical fidelity and feasibility of simulated interactions. For agent policy evaluation, we measure task success rate, response count, token usage, and execution time, where success is evaluated based on predefined completion criteria and human inspection, while the remaining metrics quantify interaction efficiency. Token usage is the sum of input and output tokens; cached input tokens are included in the input count and are not counted twice. Agent-policy execution time measures the wall-clock duration of task execution, including agent perception, reasoning and planning, and controller execution. These policy-execution costs exclude scene construction and the separate post-episode skill-extraction process.

Baselines. For both Real2Sim and agent-policy evaluation, we compare against GPT-6 Astra with medium reasoning effort and GPT-5.6 Sol with xhigh reasoning effort. For Real2Sim, the baselines directly reconstruct Blender and MuJoCo scenes from the provided demonstrations and robot models, and reproduce the demonstrated manipulation through physical simulation; detailed construction prompts are provided in Appendix F. The policy baselines use Direct Mode: given task instructions, live multi-view observations, robot URDFs, and perception and arm-control APIs, each model directly selects and executes actions in a closed loop. Baseline prompts and interfaces are detailed in Appendices F and G. We additionally evaluate *Ours (w/ skills)*, which reports performance on the second execution of each task using skills extracted from its first execution.

Implementation Details. Scene construction, policy execution, and skill extraction all use GPT-6 Astra with medium reasoning effort. Each new task starts in an independent session, while successive decisions within the task retain the conversation context. For all simulation experiments, both

Table 2: **Quantitative comparison of agent policies.** Mean results over 12 tasks per dataset, including failures. **Bold** and underlined denote best and second-best results.

	DROID				EgoDex			
Method	**SR (%)↑**	**Responses↓**	**Tokens (M)↓**	**Time (min)↓**	**SR (%)↑**	**Responses↓**	**Tokens (M)↓**	**Time (min)↓**
GPT-5.6 Sol xhigh	58.33	71.42	8.57	25.72	41.67	105.50	13.63	28.73
GPT-6 Astra Medium	75.00	28.33	1.42	7.81	66.67	33.75	2.56	10.31
Ours	75.00	16.17	0.64	8.60	**83.33**	16.67	0.66	9.09
Ours (w/ skills)	**91.67**	**13.33**	**0.51**	**6.32**	**83.33**	**14.25**	**0.49**	**7.05**

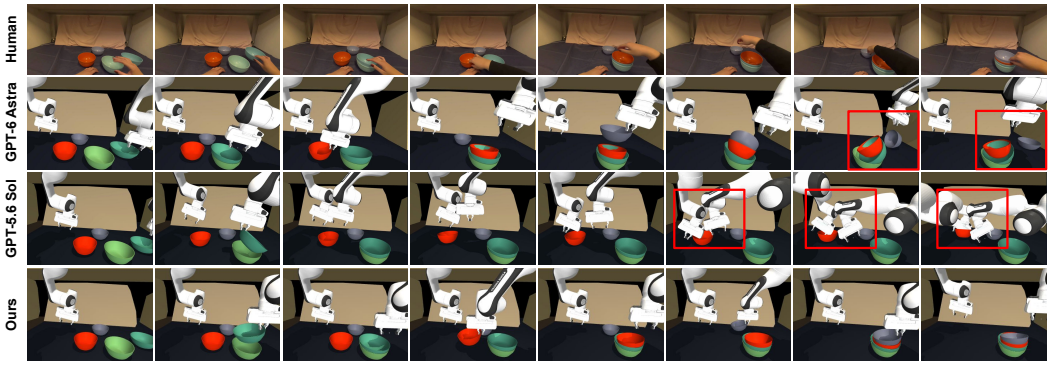


Figure 4: **Qualitative comparison of zero-shot agent execution.** Rows depict the human demonstration, GPT-6 Astra, GPT-5.6 Sol, and our agent on the bowl-stacking task. Our agent successfully completes the stack, whereas both baselines fail, with red boxes highlighting key failure cases.

the baselines and our framework are strictly restricted to the designated observations and are prohibited from accessing privileged simulator information, such as ground-truth object poses, trajectories, or other internal simulator states. Our standard policy budget is 50 model decisions per task and 600 control steps per code execution. A task is unsuccessful if its predefined completion criteria remain unmet when the budget is exhausted. We use Blender 4.5.3 LTS for scene construction and rendering, MuJoCo 3.3.7 (Todorov et al., 2012) for physics simulation, SAM3 0.1.0 (Carion et al., 2026) for agent perception, and the PyTorch implementation of Contact-GraspNet (Sundermeyer et al., 2021) for grasp proposals.

4.2 Comparison

Real2Sim Reconstruction. Table 1 shows that Real2Gym outperforms both baselines across all four metrics on DROID and EgoDex. On DROID, viewpoint alignment improves by 40.83 points over GPT-6 Astra. On EgoDex, the simulation success score reaches 85.76, a relative improvement of 51.5% over the same baseline. Figure 3 illustrates the visual improvements: our reconstructions more closely preserve cabinet structures, relative object sizes and positions, and camera framing, whereas the baselines often simplify or rearrange the scene. In the human-demonstration examples, Real2Gym retains the tabletop layout and task-relevant object relationships while replacing human hands with robot arms. The higher action-fidelity and simulation-success scores further indicate that closer visual correspondence is accompanied by better reproduction of the demonstrated interactions.

Agent Policy Performance. Table 2 shows that, averaged across both datasets, our agent achieves higher task success with fewer responses, lower token usage, and shorter execution time than both baselines. Before skill accumulation, success averages 79.2%, with 67.3% fewer tokens than GPT-6 Astra. A key difference is the decision granularity: the GPT-6 Astra baseline makes individual-action decisions, whereas our agent generates code for a complete operation stage. Each response can therefore coordinate multiple actions and verification checks, reducing the number of model interactions. SAM3 and GraspNet additionally support object localization and grasp generation, helping the agent handle more demanding manipulation tasks.

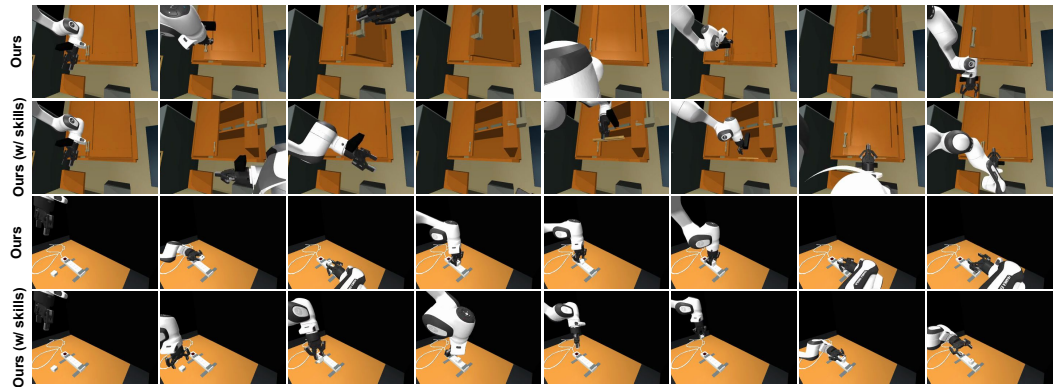


Figure 5: **Qualitative examples of skill reuse** in cabinet manipulation and adapter placement. Earlier exploratory executions are compared with subsequent skill-conditioned runs.

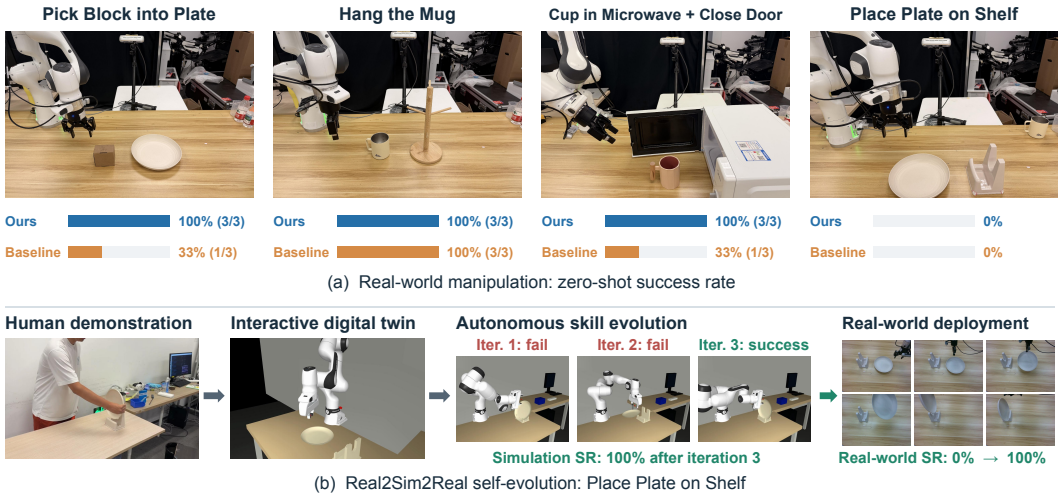


Figure 6: **Real-world deployment and Real2Sim2Real self-evolution.** (a) Four real-world manipulation tasks and zero-shot success rates of our method versus Direct Mode. (b) Failure-driven self-evolution: reconstructing a digital twin from a human demonstration, acquiring skills through simulation, and deploying the evolved skills to improve real-world success.

Figure 4 shows our agent successfully completing the illustrated bowl-stacking task, while GPT-6 Astra and GPT-5.6 Sol fail to complete the stack. Figure 5 illustrates how experience from exploratory executions guides subsequent skill-conditioned runs toward successful cabinet manipulation and more direct adapter placement. Reusing these lessons reduces repeated exploration, consistent with the lower average token usage of *Ours (w/ skills)* in Table 2.

4.3 Real-World Deployment and Self-Evolution

Experimental Setup. We rigorously evaluate our system on a 7-DoF Franka Emika Research 3 robot with a Robotiq 2F-85 gripper and two Intel RealSense RGB-D cameras (wrist-mounted eye-in-hand and static exterior views). We design four physical manipulation tasks (Fig. 6(a)): (1) **Pick Block into Plate** (short-horizon tabletop pick-and-place), (2) **Hang the Mug** (high-precision handle hanging), (3) **Place Cup in Microwave and Close Door** (long-horizon articulated manipulation), and (4) **Place Plate on Shelf** (narrow-clearance contact-rich insertion). We benchmark against the Direct Mode baseline (powered by GPT-6 Astra with medium reasoning effort) across multiple trials per task.

Zero-Shot Performance. Our zero-shot agent matches or exceeds the Direct Mode baseline in task success rate (SR) on all four tasks: (1) Our method achieves 100% SR $(3/3)$ with reduced completion time versus 33% SR $(1/3)$ for the baseline(Fig. 6(a)). Baseline failures occur because the block width slightly exceeds the horizontal gripper span; lacking spatial orientation awareness, the baseline attempts horizontal grasps that jam the gripper, whereas our agent executes a feasible top-down grasp. (2) Both methods achieve 100% SR $(3/3)$, but our approach completes the task substantially faster ($1,302.7$ s vs. $1,522.7$ s total execution time) due to phase-level code generation that plans complete operational stages rather than relying on rigid, fixed-length action chunks. (3) Our agent maintains 100% SR $(3/3)$ compared to 33% $(1/3)$ for the baseline. Baseline trials fail due to imprecise cup placement on the doorway threshold or joint safety stops triggered by end-effector collisions with the door frame during retraction. (4) Both methods achieve 0% SR under zero-shot settings (Fig. 6(a)), reflecting the substantially higher precision requirements in grasping, orientation control, and narrow-clearance placement.

Real2Sim2Real. The zero-shot failures on **Place Plate on Shelf** arise from strict physical and geometric constraints: (1) grasping requires high-precision depth control to prevent table collision, (2) the approach orientation must adapt the curved plate edge to the parallel-jaw geometry, and (3) the placement demands high dexterity to snugly insert the plate within the narrow shelf frame. To recover, we record an uncalibrated third-person video of a human performing the task and reconstruct it into an interactive digital twin via our Real2Sim pipeline(Fig. 6(b)). The agent autonomously explores within the simulation twin. Although the first two attempts fail, it acquires a stable skill by the third evolution iteration, achieving a 100% simulation success rate (SR). Deploying the evolved skill library back onto the physical Franka manipulator yields a 100% success rate (Fig. 6(b)), demonstrating our framework's capability for failure-driven skill acquisition.

5 LIMITATIONS

Although Real2Gym enables complex simulation-based exploration and successful Real2Sim2Real skill transfer, several limitations remain. First, regarding 3D reconstruction fidelity, multi-view misalignment between wrist-mounted and static cameras can yield ghosting or layered surfaces in noisy Pi3X point clouds, leaving residual geometric and textural discrepancies from the original source video. Enhancing cross-view registration and appearance refinement remains a key future priority. Second, regarding control granularity, while coarse stage-level code generation optimizes interaction efficiency, it lacks the local fine-grained reactivity required for highly constrained manipulation. We plan to explore adaptive switching between stage-level and fine action-level control to balance execution speed with local kinematic precision. Third, regarding embodiment coverage, our real-world experiments are currently confined to physical Franka arms with parallel-jaw grippers. Extending the framework to dexterous hands and diverse humanoid platforms will be critical to rigorously assessing its broader practical generality.

6 CONCLUSION

We introduce Real2Gym, a unified framework that converts human and robot demonstrations into interactive simulation environments and uses execution feedback to accumulate reusable robot skills. Our Real2Sim pipeline produces reconstructions with higher visual fidelity and more faithful physical interactions than the evaluated baselines. In simulation, our agent with accumulated skills achieves 87.5% task success with approximately 75% fewer policy-execution tokens than GPT-6 Astra. Experiments on a physical Franka robot also show higher overall success than direct GPT-6 Astra control, while skills refined in simulation enable a previously failed plate-placement task, completing the Real2Sim2Real loop. The framework supports building collections of simulation environments from human demonstrations, where agents make decisions, receive feedback, and update skills that subsequently guide physical robot execution. This provides a paradigm for robot self-improvement from real-world data, using simulation to turn human manipulation experience into reusable robot capabilities.

AI USE STATEMENT

In this work, we used generative AI tools to assist with generating synthetic datasets, implementing methodologies and writing or modifying research code, translating research-related text, improving the readability and presentation of the manuscript, and searching for and summarizing relevant literature. We did not use generative AI tools to design the research methods or experiments, interpret experimental results, propose or refine hypotheses, clean or reformat datasets, support qualitative or thematic data analysis, formulate mathematical claims, provide key elements for proving mathematical claims, or assist in writing mathematical proofs; these tasks were either performed by the authors or were not applicable to this work.

All AI-assisted work was carefully reviewed by the authors. Synthetic datasets were inspected for relevance and correctness before use. AI-assisted code was inspected, tested, and verified by the authors. Translated and polished text was cross-checked sentence-by-sentence to ensure that the original meaning was preserved. Literature identified or summarized with the assistance of generative AI was independently checked against the corresponding original sources. The authors take full responsibility for the final content of this work, including all text, methods, code, datasets, claims, experimental results, and other artifacts produced with the assistance of generative AI.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Guanxiong Chen, Qianjun Xia, Jiawei Peng, Heng Zhang, Bole Ma, Justin Qian, Ziyi Jiao, Bingyang Zhou, Luoxin Ye, Kaifeng Zhang, et al. Agentic Real2Sim: Physics-based world modeling with vision-language agents. *arXiv preprint arXiv:2607.19190*, 2026.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Ryan Hoque, Peide Huang, David Yoon, Jian Zhang, et al. Egodex: Learning dexterous manipulation from large-scale egocentric video. In *International Conference on Learning Representations*, volume 2026, pp. 4218–4237, 2026.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Mengzhao Jia, Yang Lin, Xixin Zhang, Zhihan Zhang, Xiaobai Liu, and Meng Jiang. Agent as policy for robotic manipulation. *arXiv preprint arXiv:2609.12541*, 2026.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Lingyu Kong, Ruicheng Li, Ruicheng Wang, Sicheng Xu, Chengtang Yao, Jianfeng Xiang, and Jiaolong Yang. MoGe-3: Fine-detail monocular geometry estimation with self-guided sparse volumetric refinement. *arXiv e-prints*, 2026. URL `https://arxiv.org/abs/2607.17967`.

Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator. *arXiv preprint arXiv:2411.11839*, 2024.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

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.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic/skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

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

Ben Moran, Mauro Comi, Arunkumar Byravan, Steven Bohez, Tom Erez, Zhibin Li, and Leonard Hasenclever. Splatting physical scenes: End-to-end real-to-sim from imperfect robot data. *arXiv preprint arXiv:2506.04120*, 2025.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

M Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6502–6509. IEEE, 2025.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 13438–13444. IEEE, 2021.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

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)*, pp. 23–30. IEEE, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE, 2024.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-Equivariant Visual Geometry Learning. In *International Conference on Learning Representations*, volume 2026, pp. 10481–10497, 2026.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pp. 737–744. IEEE, 2020.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Weipeng Zhong, Peizhou Cao, Yichen Jin, Li Luo, Wenzhe Cai, Jingli Lin, Hanqing Wang, Zhaoyang Lyu, Tai Wang, Bo Dai, Xudong Xu, and Jiangmiao Pang. InternScenes: A large-scale simulatable indoor scene dataset with realistic layouts. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/2ff259175587ee70657276aff28533b3-Abstract-Datasets_and_Benchmarks_Track.html.

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

This appendix provides detailed per-task Real2Sim results (Appendix A), per-task simulation control results (Appendix B), real-world execution sequences (Appendix C), an ablation study of VLM model choice and reasoning effort (Appendix D), Real2Sim evaluation criteria and prompts (Appendix E), Real2Sim baseline implementation details (Appendix F), and Agent Policy baseline instructions (Appendix G).

A Per-Task Real2Sim Results

Table 3 reports task-level Real2Sim results on DROID (Khazatsky et al., 2024) and EgoDex (Hoque et al., 2026) for the two model baselines and Real2Gym. The four metrics measure content alignment, viewpoint alignment, action fidelity, and simulation success score. Dataset means summarize performance across the listed tasks; the evaluation criteria are provided in Appendix E. Real2Gym generally improves both visual alignment and interaction fidelity across the two datasets, although gains vary by task.

Table 3: Per-task Real2Sim evaluation on DROID and EgoDex. Methods are evaluated across content alignment, viewpoint alignment, action fidelity, and simulation success score (0–100 scale, ↑). Bold indicates the top performance per task and metric, including ties.

	GPT-5.6 Sol xhigh				**GPT-6 Astra Medium**				**Real2Gym (Ours)**			
Task	Content alignment	Viewpoint alignment	Action fidelity	Simulation success score	Content alignment	Viewpoint alignment	Action fidelity	Simulation success score	Content alignment	Viewpoint alignment	Action fidelity	Simulation success score
DROID												
D01	55.00	30.00	85.00	76.50	60.00	30.00	80.00	80.00	**80.00**	**60.00**	**93.00**	**93.00**
D02	55.00	25.00	85.00	85.00	60.00	30.00	85.00	85.00	**70.00**	**80.00**	**88.00**	**88.00**
D03	55.00	20.00	**88.00**	**88.00**	60.00	30.00	80.00	80.00	**70.00**	**80.00**	**88.00**	**88.00**
D04	60.00	35.00	**90.00**	**90.00**	60.00	30.00	78.00	78.00	**75.00**	**80.00**	86.00	86.00
D05	**60.00**	30.00	40.00	34.00	**60.00**	30.00	**86.00**	**86.00**	**60.00**	**70.00**	65.00	45.50
D06	45.00	15.00	73.00	73.00	60.00	30.00	83.00	83.00	**70.00**	**80.00**	**88.00**	**88.00**
D07	50.00	20.00	40.00	32.00	**60.00**	30.00	83.00	**83.00**	**60.00**	**80.00**	**86.00**	81.70
D08	40.00	25.00	40.00	0.00	40.00	30.00	60.00	0.00	**60.00**	**70.00**	**80.00**	**80.00**
D09	55.00	30.00	83.00	83.00	**60.00**	30.00	80.00	80.00	**60.00**	**60.00**	**90.00**	**90.00**
D10	50.00	30.00	40.00	26.00	60.00	30.00	70.00	70.00	**70.00**	**60.00**	**83.00**	**83.00**
D11	30.00	15.00	40.00	0.00	30.00	30.00	40.00	40.00	**60.00**	**70.00**	**78.00**	**78.00**
D12	45.00	15.00	62.00	0.00	50.00	30.00	**88.00**	**88.00**	**60.00**	**60.00**	73.00	69.35
Mean	50.00	24.17	63.83	48.96	55.00	30.00	76.08	71.08	**66.25**	**70.83**	**83.17**	**80.88**
EgoDex												
E01	60.00	30.00	40.00	40.00	54.00	40.00	83.00	83.00	**76.00**	**88.00**	**91.00**	**91.00**
E02	60.00	40.00	40.00	36.80	56.00	50.00	75.00	75.00	**71.00**	**78.00**	**78.00**	**78.00**
E03	60.00	30.00	72.00	50.40	65.00	40.00	73.00	40.15	**76.00**	**75.00**	**90.00**	**88.20**
E04	60.00	**55.00**	**90.00**	**90.00**	**73.00**	38.00	85.00	85.00	66.00	45.00	88.00	88.00
E05	60.00	25.00	73.00	69.35	57.00	32.00	73.00	73.00	**82.00**	**78.00**	**90.00**	**90.00**
E06	60.00	25.00	40.00	40.00	59.00	30.00	73.00	73.00	**78.00**	**82.00**	**88.00**	**88.00**
E07	55.00	30.00	40.00	38.00	58.00	35.00	52.00	36.92	**72.00**	**72.00**	**81.00**	**76.95**
E08	50.00	45.00	68.00	56.76	55.00	**50.00**	37.00	22.94	**73.00**	45.00	**83.00**	**83.00**
E09	45.00	35.00	58.00	36.54	48.00	35.00	70.00	44.10	**71.00**	**70.00**	**95.00**	**95.00**
E10	50.00	30.00	68.00	61.88	57.00	34.00	65.00	59.80	**75.00**	**84.00**	**88.00**	**88.00**
E11	55.00	35.00	40.00	34.00	57.00	35.00	73.00	67.89	**76.00**	**86.00**	**83.00**	**83.00**
E12	50.00	30.00	29.00	7.54	52.00	45.00	30.00	18.30	**85.00**	**82.00**	**80.00**	**80.00**
Mean	55.42	34.17	54.83	46.77	57.58	38.67	65.75	56.59	**75.08**	**73.75**	**86.25**	**85.76**

B Per-Task Simulation Control Results

Table 4 compares task success and execution efficiency across baselines and our method with and without skills on DROID and EgoDex. With skills, our agent additionally solves D10 and D11 and reduces token usage on most tasks, while several failures remain. Task-specific success checks are also central to manipulation benchmarks such as robosuite, RLBench, and LIBERO (Zhu et al., 2020; James et al., 2020; Liu et al., 2023).

Table 4: **Per-task efficiency and success results** on DROID and EgoDex. Responses denote model responses, and tokens are reported in millions. Lower is better for response count, token usage, and time.

GPT-5.6 Sol xhigh

	DROID					EgoDex			
Task	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**	**Task**	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**
D1	✓	52	6.88	25.09	E1	✓	75	9.12	36.88
D2	✓	60	6.25	27.78	E2	✓	59	6.12	14.76
D3	✓	81	9.32	29.77	E3	×	62	8.25	13.15
D4	×	31	4.10	11.60	E4	✓	42	4.27	15.44
D5	×	53	6.44	28.45	E5	✓	53	5.93	23.16
D6	✓	56	4.83	23.43	E6	×	81	12.50	24.11
D7	✓	50	4.63	25.45	E7	×	82	10.31	25.19
D8	✓	62	6.69	29.81	E8	×	317	41.32	65.55
D9	×	39	5.53	16.91	E9	✓	92	10.00	24.86
D10	×	76	10.26	16.55	E10	×	55	7.22	16.64
D11	×	185	24.53	38.58	E11	×	133	18.77	28.91
D12	✓	112	13.37	35.22	E12	×	215	29.77	56.05

GPT-6 Astra Medium

	DROID					EgoDex			
Task	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**	**Task**	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**
D1	✓	42	1.88	10.04	E1	✓	22	1.00	6.70
D2	✓	20	0.85	11.04	E2	✓	21	0.87	6.37
D3	✓	19	0.75	6.03	E3	✓	21	0.98	8.37
D4	✓	32	1.86	9.37	E4	✓	17	0.73	4.35
D5	×	26	1.18	7.36	E5	✓	18	0.84	5.69
D6	✓	22	0.84	5.02	E6	×	61	3.98	17.42
D7	×	10	0.27	4.02	E7	×	24	1.22	7.70
D8	✓	16	0.75	4.35	E8	✓	34	1.95	9.37
D9	✓	47	2.29	11.04	E9	✓	21	1.01	6.36
D10	✓	30	1.89	8.03	E10	×	45	2.72	10.39
D11	✓	45	2.86	9.03	E11	✓	52	6.78	14.08
D12	×	31	1.65	8.37	E12	×	69	8.66	26.91

Ours

	DROID					EgoDex			
Task	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**	**Task**	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**
D1	✓	8	0.21	3.91	E1	✓	6	0.15	3.59
D2	✓	10	0.26	4.82	E2	✓	12	0.32	4.29
D3	✓	8	0.20	3.86	E3	✓	14	0.45	10.91
D4	✓	9	0.26	5.05	E4	✓	7	0.19	3.47
D5	×	17	0.60	10.51	E5	✓	9	0.26	4.82
D6	✓	9	0.25	4.95	E6	✓	16	0.52	8.38
D7	✓	12	0.40	17.32	E7	✓	23	0.93	12.61
D8	✓	14	0.45	7.30	E8	✓	25	1.40	13.52
D9	✓	20	0.70	9.75	E9	✓	14	0.52	8.27
D10	×	25	1.11	11.45	E10	✓	24	0.93	11.73
D11	×	38	2.19	10.94	E11	×	25	1.14	13.47
D12	✓	24	1.01	13.36	E12	×	25	1.15	13.96

Ours (w/ skills)

	DROID					EgoDex			
Task	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**	**Task**	**Success**	**Resp.↓**	**Tokens (M)↓**	**Time (min)↓**
D1	✓	7	0.17	3.45	E1	✓	6	0.15	3.00
D2	✓	9	0.23	3.62	E2	✓	10	0.25	3.74
D3	✓	6	0.15	2.86	E3	✓	10	0.27	3.90
D4	✓	8	0.23	3.97	E4	✓	7	0.18	3.43
D5	×	14	0.48	9.09	E5	✓	6	0.15	3.17
D6	✓	7	0.20	4.26	E6	✓	13	0.37	6.74
D7	✓	8	0.23	5.05	E7	✓	17	0.54	7.73
D8	✓	14	0.44	6.45	E8	✓	26	1.10	12.78
D9	✓	19	0.63	9.13	E9	✓	16	0.47	6.66
D10	✓	28	1.95	9.42	E10	✓	20	0.90	14.57
D11	✓	22	0.87	10.72	E11	×	17	0.56	8.38
D12	✓	18	0.58	7.76	E12	×	23	0.97	10.48



Figure 7: **Real-world manipulation trajectories.** The baseline and our method are shown across four tasks, with wrist-view insets. The shelf task shows our method before and after skill evolution.

C Real-World Execution Sequences

Figure 7 supplements the real-world experiments with manipulation sequences for the baseline and our method across four tasks. Wrist-view insets provide additional views of the interactions. For the shelf task, the figure also shows our method before and after skill evolution.

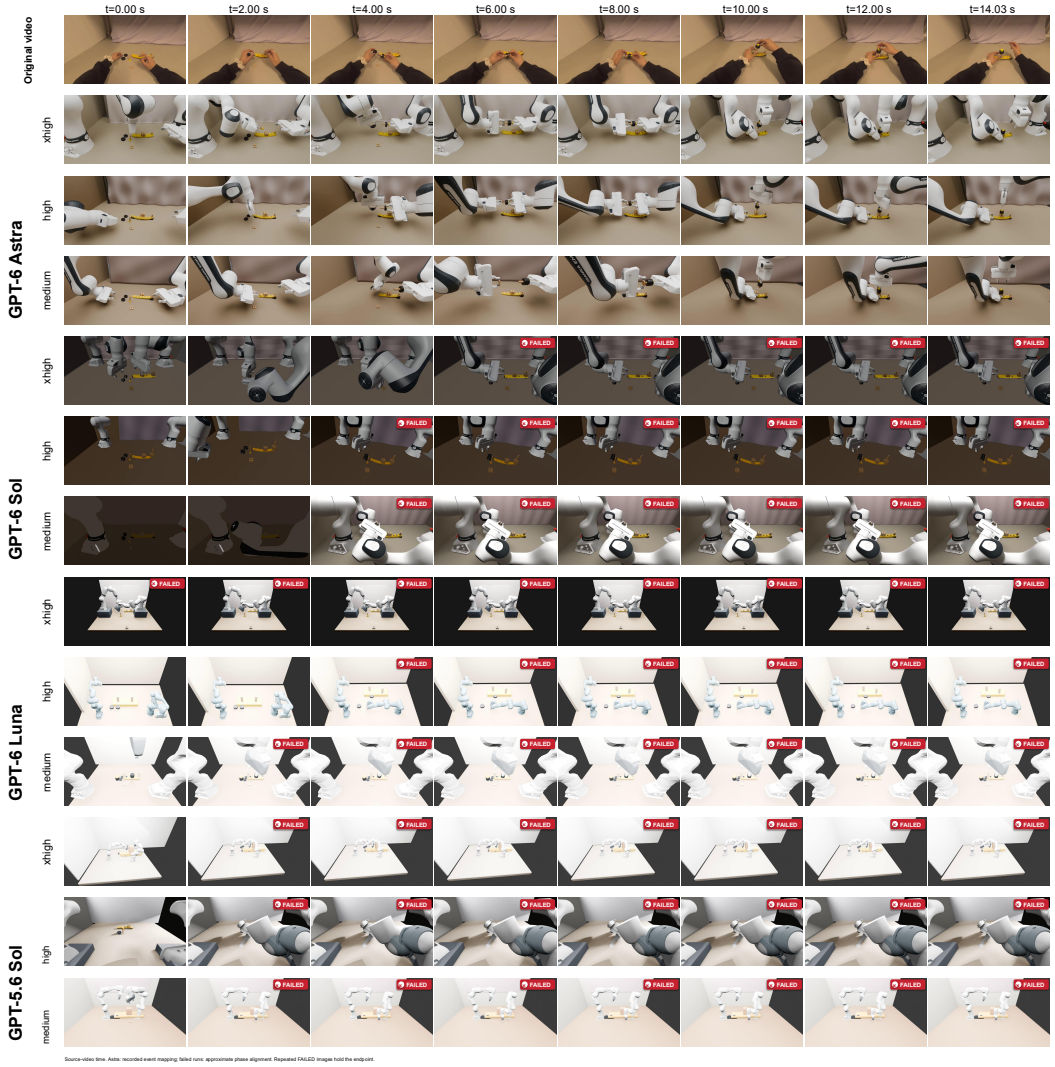


Figure 8: **Qualitative comparison across models and reasoning effort levels on an EgoDex assembly task.** The top row depicts the source human demonstration, while subsequent rows show the 12 experimental configurations. Columns are indexed by source-video timestamps. Successful trajectories are aligned using recorded event correspondences, whereas failed attempts are approximately aligned by the attempted action phase. Repeated frames marked with `FAILED` retain a frozen state for visual comparison and do not imply continued execution.

D Ablation on VLM Model and Reasoning Effort

We evaluate the effects of model choice and reasoning effort on a challenging bimanual assembly task from EgoDex. The source demonstration involves placing two wheels and a square nut onto a bolt, tightening the nut, and inserting the assembly into a base. Reproducing this sequence requires stable grasping, precise hole–shaft alignment, and contact dynamics coupling rotation with axial motion. Figure 8 compares four models across three reasoning effort levels (medium, high, and xhigh), yielding 12 configurations. GPT-6 Astra and GPT-6 Sol generally recover the main objects and scene layout in reasonable agreement with the source video, although Sol retains visual alignment and occlusion discrepancies. Only GPT-6 Astra passes the full-task simulation criteria at all three effort levels, completing an adapted execution that includes nut tightening and final insertion. GPT-6 Sol at high and medium effort fails during grasping or before assembling the first wheel, while xhigh reaches the second-wheel stage but fails to release it; none reaches successful nut

tightening. GPT-6 Luna and GPT-5.6 Sol exhibit larger discrepancies in scene geometry, object relationships, or viewpoint alignment. Their final candidates fail during initialization, early grasping, or local manipulation checks and do not reach the full assembly sequence. This case study highlights the gap between visually plausible scene reconstruction and physically executable, long-horizon manipulation.

E REAL2SIM EVALUATION CRITERIA AND PROMPT TEMPLATE

The following prompt specifies the scoring criteria for the Real2Sim results in Table 3. It assesses reconstruction and demonstration reproduction; the simulation success score defined here is distinct from the task-success outcomes reported in Appendix B. Model-based scoring is related to LLM-as-a-judge evaluation (Zheng et al., 2023); here, the explicit rubric is specific to reconstruction and execution evidence, and does not replace native task-success checks.

E0 Evaluation instructions

SHARED EVALUATION RULES

Using [source demonstration and reference data], [submitted reconstruction and execution evidence], and [shared task checklist], evaluate [method] on [scene]. Score content alignment, viewpoint alignment, action fidelity, and simulation success score on a 0-100 scale. Apply the same criteria and weights to all methods.

Human-to-robot substitution.

For human demonstrations, replacing the human arms with the prescribed target robot is permitted and must not itself incur a penalty. Still assess whether the task objects, scene layout, necessary contact relations, and causal relationships underlying the manipulation are preserved.

E1 1. CONTENT ALIGNMENT

EVALUATION CRITERION

Evaluation focus.

Assess correspondence between the reconstructed scene and the source demonstration in object identity and count, geometry, initial spatial layout, scale, appearance, materials, background, and task-relevant structures.

Scoring scheme.

The total score is 100, comprising source-to-MuJoCo content alignment (50 points) and source-to-Blender content alignment (50 points). Score each component separately using evidence from its respective reconstruction.

Scoring anchors.

Apply the following anchors to each component:

- 50: Key objects, layout, and task-relevant structures correspond almost completely, with minimal appearance differences.
- 40: The main content corresponds overall, with minor differences in local geometry, materials, or background.
- 30: The main objects and task setting are recognizable, but there is clear simplification, layout error, or missing content.
- 15: Only part of the key content corresponds, with substantial object omissions, substitutions, or structural errors.
- 0: Evidence shows that the key content does not correspond to the source demonstration.

Assessment constraints.

Inspect actual Blender and MuJoCo evidence; do not infer the quality of one engine's reconstruction from the other. Support intermediate scores with specific evidence.

E2 2. VIEWPOINT ALIGNMENT

EVALUATION CRITERION

Evaluation focus.

Assess correspondence to the source video in camera position, orientation, distance, field of view, framing, visibility of key objects, and occlusion relationships.

Scoring scheme.

Assign an overall score out of 100 using images generated by the submitted scene's existing camera.

Scoring anchors.

Apply the following anchors:

- 100: Viewing direction, coverage, and framing closely match the source; the image positions, scales, and occlusion relationships of key objects correspond almost completely.
- 80: The overall viewpoint is similar, with minor differences in position, angle, or field of view; key objects and interaction regions retain comparable visibility.
- 60: The same task region is visible, but viewing direction, distance, framing, or occlusion differs noticeably.
- 30: Only part of the task region or a subset of key objects corresponds; large viewpoint differences leave important interaction regions outside the view or occluded.
- 0: Evidence shows that the viewpoint is substantially inconsistent with the source, preventing visual correspondence of the key task region.

Assessment constraints.

Do not refit the camera or use cropping to conceal viewpoint differences. Image displacement across different views must not be interpreted as a calibrated 3D position error. Explain the source of deviations when assigning intermediate scores.

E3 3. ACTION FIDELITY

EVALUATION CRITERION

Evaluation focus.

Assess alignment with the source video's manipulation goals, objects, necessary action ordering, contact relations, motion paths, relative timing, and initial/final states, independently of task completion.

Scoring scheme.

Action fidelity is scored out of 100 by summing six components, subject to the causal-equivalence cap below:

- Goals and objects (10 points): Whether the same or task-equivalent objects are manipulated toward the same goal.
- Necessary actions and ordering (20 points): Whether the necessary operations and their ordering are preserved.
- Contact relations and manipulation mode (25 points): Whether grasping, pushing, support, and other contact relations retain their causal roles.
- Key motion paths and poses (25 points): Whether key translations, rotations, approaches, and releases correspond.
- Relative timing (10 points): Whether relative timing and coordination between actions correspond.
- Initial and final states (10 points): Whether the key object states before and after manipulation correspond.

E3 3. ACTION FIDELITY (continued)

Scoring anchors.

Determine the proportion of credit for each action-fidelity component using the following anchors:

- 100%: The observable key action relationships correspond completely.
- Approximately 80%: Overall correspondence is preserved, with only minor deviations.
- Approximately 60%: Clear simplification is present, but meaningful action correspondence remains.
- Approximately 30%: Only limited action correspondence remains.
- 0%: Evidence shows that the action relationships do not correspond.

Use five-point increments where appropriate; ten-point components may use 0, 2, 5, 8, and 10 points. Justify other intermediate values.

Assessment constraints.

Assign zero action fidelity if the overall manipulation goal is wrong or no corresponding action exists. If changes to necessary action ordering or contact causality make the manipulation logic non-equivalent, cap action fidelity at 40 and identify the violated requirement. Do not apply this cap solely because of human-robot embodiment differences, inverse kinematics, reasonable motion smoothing, or global speed scaling.

E4 4. SIMULATION SUCCESS SCORE

EVALUATION CRITERION

Evaluation focus.

Assess the extent to which execution both reproduces the source demonstration and completes the required manipulation. Completing a task through a process that does not align with the source should receive low credit.

Scoring scheme.

For each scene, compute Simulation success score = (Action fidelity * Task completion) / 100, where both component scores range from 0 to 100. Apply the confirmed physical-failure override described below. Average the resulting per-scene values; do not multiply dataset-level means.

Task completion is an intermediate quantity, not a separately reported metric. It assesses the required steps, intended final state, and physically plausible execution. Its total score is 100, comprising:

- Required task steps (40 points): Whether the required grasping, transport, placement, release, or other necessary operations are performed.
- Correct final state (40 points): Whether object positions, orientations, counts, assembly relationships, or opening/closing states satisfy the task requirements.
- Physical execution (20 points): Whether object motion results from robot control and physical contact, with appropriate collision, joint, support, and stability behavior.

Scoring anchors.

Derive evidence-checkable criteria for each component from the source demonstration and assign a weight to each criterion. Use identical criteria and weights for all methods on the same scene. Award full credit when a criterion is fully satisfied, partial credit according to the observed steps, object counts, or state requirements, and zero when the criterion is demonstrably unmet. Justify partial credit with evidence rather than an overall impression.

Assessment constraints.

Directly imposed object trajectories or hidden attachments do not establish a physical grasp. Ordinary position servos and replaying saved states for rendering do not, by themselves,

E4 4. SIMULATION SUCCESS SCORE (continued)

invalidate physical execution. Program termination or a generation timeout alone does not determine task completion.

Physical-failure rule.

Assign zero simulation success when a physical simulation failure, such as persistent penetration or jamming that breaks task execution, is confirmed. Ordinary task omissions, partial completion, or a generation timeout alone do not trigger this override.

F REAL2SIM BASELINE IMPLEMENTATION

The Real2Sim construction baselines are instructed to reconstruct Blender and MuJoCo scenes from the supplied demonstrations and to reproduce the demonstrated manipulation in simulation. The templates below specify the input observations, required manipulation sequence, and robot-model constraints for DROID and EgoDex. Bracketed fields are replaced with the corresponding task inputs. Related efforts provide diverse interactive scenes, including RoboCasa (Nasiriany et al., 2024) and InternScenes (Zhong et al., 2025), while MimicGen (Mandlekar et al., 2023) adapts demonstrations to new configurations. The prompts below instead specify reconstruction and execution for each supplied demonstration.

F.1 ROBOT DEMONSTRATIONS: DROID

C1 Robot Demonstrations: DROID

CONSTRUCTION PROMPT

Inputs and task.

Read [external-view videos], [wrist-view video], and [robot trajectory file]. Using [robot URDF] and its associated meshes, construct Blender and MuJoCo scenes for [task description].

Scene reconstruction and action reproduction.

Reconstruct the scene from the multi-view videos and robot trajectories. In MuJoCo, reproduce [required manipulation sequence] through physical simulation and execute the complete manipulation process. The manipulated objects, actions, and their ordering must follow the source videos.

Robot-model constraints.

Use the supplied Franka Panda arm and Robotiq 2F-85 gripper in the original scenes and all augmented scenes in both engines. Preserve the original URDF's link structure, joint types, axes, origins, limits, gripper coupling and mimic relationships, geometric dimensions, mesh scales, masses, centers of mass, and inertias. These structures and parameters must remain equivalent in the target engines after format conversion.

F.2 HUMAN DEMONSTRATIONS: EGODEX

C2 Human Demonstrations: EgoDex

CONSTRUCTION PROMPT

Inputs and task.

Read [egocentric video]. Using [dual-arm robot URDF] and its associated meshes, construct Blender and MuJoCo scenes for [task description]. The target robot consists of two Franka FR3 arms, each equipped with a Franka Hand.

C2 Human Demonstrations: EgoDex (continued)

Scene reconstruction and action transfer.

Reconstruct the scene from the video, transfer the demonstrated human manipulation to the target robot, and reproduce [required manipulation sequence] through physical simulation in MuJoCo. Assign manipulation and assistance roles to the two arms according to the task, ensuring that object motion results from robot control and physical contact.

Robot-model constraints.

Mount the two arms on separate bases. Base positions may be adjusted to accommodate the task workspace, but the dimensions and joint structures of the Franka FR3 arms and Franka Hands must not be changed.

G Agent Policy Baseline Prompts

The agent receives an observation-only user prompt followed by a task instruction. These prompts direct it to read `SKILL.md` for shared operating rules and `api/README.md` for the common API contract. The backend specified in `config/robot.json` determines which additional documents it reads: the simulation API and workflow files under `api/`, or `api/FRANKA.md` for the physical robot. Document paths are relative to `robot-skill/` unless shown with that prefix.

Below, P0–P1 present the user-facing prompts, while P2, F1, and P3–P4 summarize the documents read by the agent. These document summaries are condensed for readability and are not verbatim reproductions or separate user messages. The interface exposes available actions in the spirit of programmatic robot prompting (Liang et al., 2023; Singh et al., 2022), while repeated observation and execution feedback support the reasoning–action loop studied in ReAct (Yao et al., 2022).

G.1 Observation and Task Prompts

Observation-only introduction. The initial prompt asks the agent to inspect the configuration and current observations without moving the robot.

P0 Observation-only introduction

USER PROMPT / FIRST TURN

Please first read robot-skill/SKILL.md, robot-skill/api/README.md, and robot-skill/config/robot.json. Then call get_state() and only report the current state without sending any actions.

Task instruction. The following template is adapted from the supplied execution example by replacing the task description with a placeholder.

P1 Task instruction

USER PROMPT / SUBSEQUENT TURN

Complete the task "[task description]" and record the full execution video. Keep the warm-up brief and perform only the minimum necessary calibration before starting the task.

G.2 Shared Skill Instructions

The following panel summarizes `SKILL.md`, which defines the shared observation, execution, verification, and memory procedures and directs the agent to the applicable backend documentation. These operating instructions are distinct from the task experience evaluated in *Ours (w/ skills)*. Feedback-based refinement and experience reuse have antecedents in Self-Refine (Madaan et al., 2023), Reflexion (Shinn et al., 2023), and Voyager's executable skill library in Minecraft (Wang et al., 2023a).

P2 Shared Skill Instructions

SOURCE: SKILL.md

Configuration and backend selection.
Read SKILL.md, api/README.md, and config/robot.json under robot-skill/. Resolve configuration paths relative to this directory. For the Franka profile, read api/FRANKA.md and the referenced robot description, runtime frame information, and camera calibrations. For the simulation profile, read the corresponding simulation API contract and control workflow documentation.

Public interface.
Use only:

from robo import reset, get_state, post_actions

The operator starts the backend separately.
Do not start services, connect to hardware drivers, or inspect backend implementations. If the service is unavailable, report the failure.

Information boundary.
Use only the skill directory and observations returned by get_state(). Do not inspect external project source, simulator internals, datasets, task definitions, or backend logs to infer scene layout or task success.

Initial observation.
When instructed to observe only, call get_state() without arguments to obtain measured state and both exterior (agentview) and wrist images. Save and open both images before describing the scene. Report measured joints, end-effector pose, gripper opening, camera observations, and the relevant frame conventions and motion limits. Do not reset, send actions, perform physical warmup, or modify configuration. Wait for the task instruction.

Task execution.
After receiving the task, obtain fresh observations before planning. Follow the selected backend's warmup and control workflow. Alternate observation, action selection, execution, and verification. Use camera images to assess objects, alignment, contact, clearance, and task progress; use measured state to check motion feasibility.

Grasp verification.
After closing the gripper, inspect both camera images and the reported gripper state before lifting. Gripper closure or a successful API return alone does not establish that an object is held.

Limits and configuration.
Respect the configured joint, workspace, speed, and action limits. Do not modify safety settings, calibration, URDF files, or driver configuration during a task.

Persistent memory.
Save observations, actions, and warmup notes under memory/ using new timestamp-and-UUID records. Preserve historical records; append invalidation notes instead of overwriting or deleting failed warmups. Check configuration fingerprints and obtain fresh observations before reusing memory.

Outcome reporting.
Keep measured state separate from requested targets. Verify completion using fresh visual and measured-state evidence. Report failure or uncertainty when completion cannot be established; a successful post_actions() call alone is not task-success evidence.

G.3 SHARED API CONTRACT

The following panel summarizes `api/README.md`, which defines the common observation fields, action formats, and API return values. Backend-specific semantics are detailed in the subsequent profiles.

F1 Shared API Contract

```
    Observation.
    state = get_state(cameras=None)

    The default call returns robot state and both agentview
and wrist images. Camera selection also supports a single
camera, requested image sizes, or an empty list for numeric
state only.

    Common observation fields.
    {
        "joint": [q1, ..., q7],
        "ee_pose": [x, y, z, ax, ay, az],
        "gripper": [g],
        "image": {
            "agentview": png_bytes,
            "wrist": png_bytes
        },
        "timestamp": time_in_seconds
    }

    Joint angles are in radians. Pose translation is in metres
 and orientation is an axis-angle vector in radians.
Coordinate frames follow the selected backend. The gripper
convention is -1 for open and +1 for closed.

    Action submission.
    post_actions({
        "actions": [action_1, ..., action_N],
        "type": "joint" or "ee_pose",
        "freq": ""
    })

    Both profiles use absolute targets:
    joint:   [q1, q2, q3, q4, q5, q6, q7, g]
    ee_pose: [x, y, z, ax, ay, az, g]

    Return values. post_actions() returns 1 for a successful
call and 0 for rejection or failure. Always inspect subsequent
 observations to determine the actual result. Follow backend-
specific timing, reset, and failure semantics.
```

G.4 SIMULATION INSTRUCTIONS

The following panel consolidates two simulation-specific documents under `api/`: the API contract and the control workflow. The agent reads these files when the simulation backend is selected in `config/robot.json`. They specify simulation reset behavior, warmup, control modes, and action execution.

P3 Simulation Instructions

SOURCE: SIMULATION API AND WORKFLOW DOCUMENTS

Reset semantics.
reset() restores the simulation scene, including the robot and task objects. Use it before a new episode or during warmup setup, not as a recovery step during task execution.

Warmup.
Before task execution, load valid warmup memory matching the current setup. Otherwise, perform small isolated probes in the up, down, left, right, forward, and backward directions. Keep the gripper unchanged, observe after each probe, and return to the recorded neutral pose before the next probe. Validate the observed motion and save the results before proceeding. Warmup is a procedure using the public API, not a separate function.

Control preference.
Prefer joint-space control unless the configuration, warmup memory, or task provides a stronger reason to use end-effector control. End-effector targets use the simulation world frame.

Action chunks.
Use nearby intermediate targets for smooth motion. Prefer chunks of at least 10 low-level actions during normal task execution, except when shorter commands are needed for safety, precise probes, or handling rejections. The agent selects the chunk length.

Timing and mode selection.
An empty freq uses the default 20 Hz simulation pacing. Switching control type within an episode is rejected. Do not reset an ongoing task merely to change control mode.

Verification.
Observe after each chunk and verify visual and numeric effects. For rejected commands, inspect the cause and correct or reduce the command without bypassing limits.

G.5 Physical Franka Instructions

The following panel summarizes `api/FRANKA.md`, which the agent reads for the physical Franka backend. It supplements the shared instructions with TCP control, camera calibration, physical reset semantics, and fault-handling procedures. The distinction between a simulated reset and a physical reset reflects the reset and intervention challenges studied in autonomous robot learning (Eysenbach et al., 2017; Dulac-Arnold et al., 2019).

P4 Physical Franka Instructions

SOURCE: api/FRANKA.md

Robot and frames.
Read the configured arm URDF, physical runtime URDF, measured flange-to-TCP transform, and exterior/wrist camera calibrations. Check the backend and mode returned by get_state(). If physical runtime metadata is missing, report it rather than inventing a tool mounting transform.

TCP control.
Prefer absolute ee_pose targets for task motion. The pose is the Franka TCP expressed in the robot base frame, using metres and axis-angle radians. The backend converts TCP targets to flange targets and solves URDF inverse kinematics. Do not apply T_align or an additional model-gripper offset.

Camera observations.
Inspect exterior and wrist RGB images, natively 640 x 480. The API also returns camera intrinsics, extrinsics, and image timestamps. Wrist extrinsics depend on the current TCP

P4 Physical Franka Instructions (continued)

pose. Images and robot states are asynchronous; prefer observations after motion settles. The API does not provide depth or point clouds, and calibration alone does not establish metric depth.

Initial movement and contact.

Use a single target for the first movement and near contacts. Use short action chunks and inspect observations after motion. Do not impose the simulator's preference for at least 10 actions or its mandatory six-direction warmup on the physical robot.

Physical warmup.

For a new setup, record measured state, both camera images, and the configuration/runtime/URDF fingerprint. If a calibration probe is needed, choose a small movement in visibly clear space, retain the gripper opening, verify measured displacement, and save the result. Do not probe into surfaces, automatically open a held object, or repeatedly increase probe magnitude.

Reset semantics.

reset() only refreshes/checks observations. It does not move the arm home, change the gripper, restore objects, or clear a motion fault.

Execution semantics.

post_actions() blocks until the targets finish or a failure occurs. Each waypoint completes before the next begins. freq is an upper submission-rate bound, not the physical servo rate. Increasing it does not bypass motion limits.

Failure feedback.

A failed call may follow partial execution. Obtain fresh state and both camera images, then inspect control_feedback and effective_safety. Do not blindly replay the failed target sequence or remaining waypoints.

Recovery and replanning.

If feedback reports recovered_replan_required, plan from the new measured state under the returned limits. Do not assume recovery restored an earlier pose or completed the task. If feedback reports operator_required, stop issuing actions and report the error. Do not clear faults through another interface.

Geometric limitations.

Respect workspace and path checks, but do not treat the URDF as a certificate of collision-free motion: unverified tool collision geometry is not represented. Verify scene clearance visually.

Task completion.

Confirm grasping and placement using both camera views and measured gripper/state feedback. The physical backend provides no simulator reward or done signal.