RoboAuto: Autonomous Robot Program Search through Multi-Level Physical Understanding

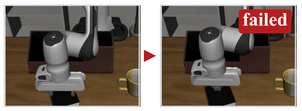
Buzhao Liu*, Jianing Ye*, Xiu Yuan*, Yangtong Feng, Chengsong Huang, Tyler Dinh, Yevgeniy Vorobeychik, Ning Zhang, Chongjie Zhang
Department of Computer Science & Engineering, Washington University in St. Louis *Equal contribution

Paper, videos & code
roboauto-ai.github.io

Introduction

Autonomous coding agents built on large language models can generate an executable solution, evaluate its outcome, and use the evidence to improve the next attempt. Code-as-Policy [1] brings this loop to robot manipulation: given "put the mug on the plate," the agent writes a program over perception and control primitives, runs it, and revises it from the rollout. CaP-Agent0 [2], ASPIRE [3], and RHO [4] show that execution feedback and retained experience help, but effective program search also requires **physical understanding**: how objects can be manipulated, how scene geometry constrains actions, and why an interaction succeeds or fails. Execution feedback alone says only *whether* a run succeeded, so the same mistake repeats in our baseline runs.

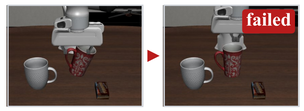
(a) Unknown affordance



descend on the book centre → pads close on the cover face

```
# wrong grasp strategy
grasp_pos = book_center
quat = top_down_quat
```

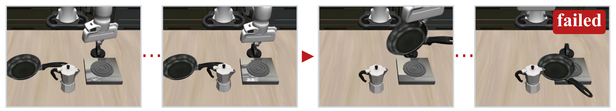
(b) Ungrounded geometry



descend to a hardcoded height → gripper closes on air

```
# height of other mug
grasp_xy = mug_center
grasp_z = top_z - 0.02
```

(c) Unverified effect

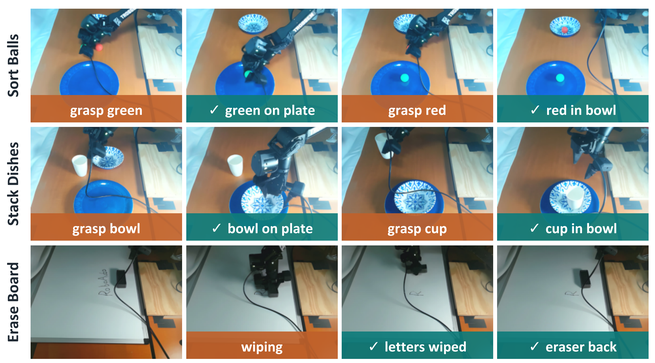


turn 1: knob, burner off → turn 3: knob, burner off → turn 4: moves on to the pan → result: pan placed on cold burner

```
close_gripper(); rotate_wrist(90)
place_pan(burner_pos)  # burner never checked
```

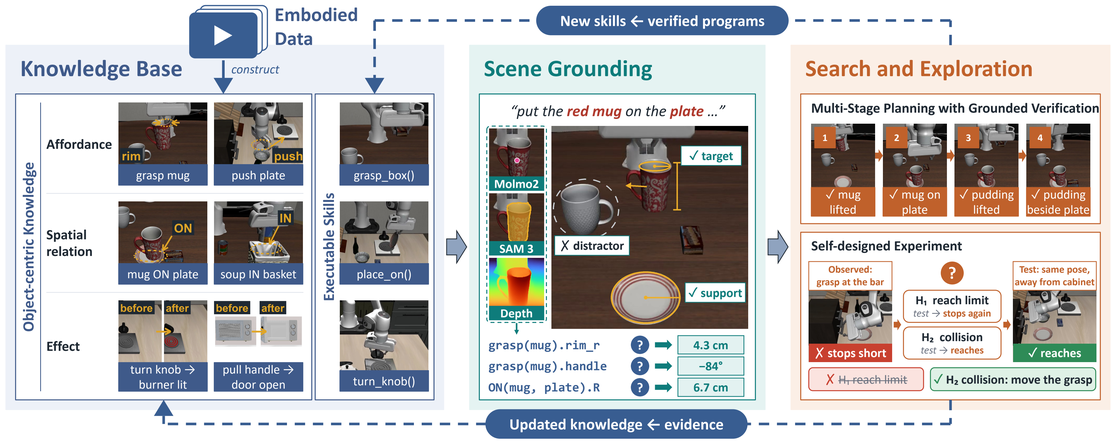
(a) CaP-Agent0 keeps grasping a standing book from the top after the first attempt topples it: missing affordance knowledge. **(b)** ASPIRE reuses a grasp height learned on another mug and the gripper closes on air: ungrounded geometry. **(c)** CaP-Agent0 places the pan although the burner never turned on: an unverified effect.

Real-Robot Evaluation



On a WidowX AI arm we compare RoboAuto, ASPIRE, and the DROID-finetuned $\pi_{0.5}$ [9] on three long-horizon precision tasks: **Sort Balls** places two slippery ping-pong balls in their receptacles, **Stack Dishes** puts a low-rimmed bowl on a plate and a cup upright in the bowl, and **Erase Board** erases "RoboAuto" and returns the eraser. Both search methods reuse their LIBERO knowledge; 20 trials per task with unseen object positions, scored by stage completion. **RoboAuto 98.6%**, ASPIRE 11.3%, $\pi_{0.5}$ 0%: knowledge learned in simulation transfers to unseen real-world tasks.

Method



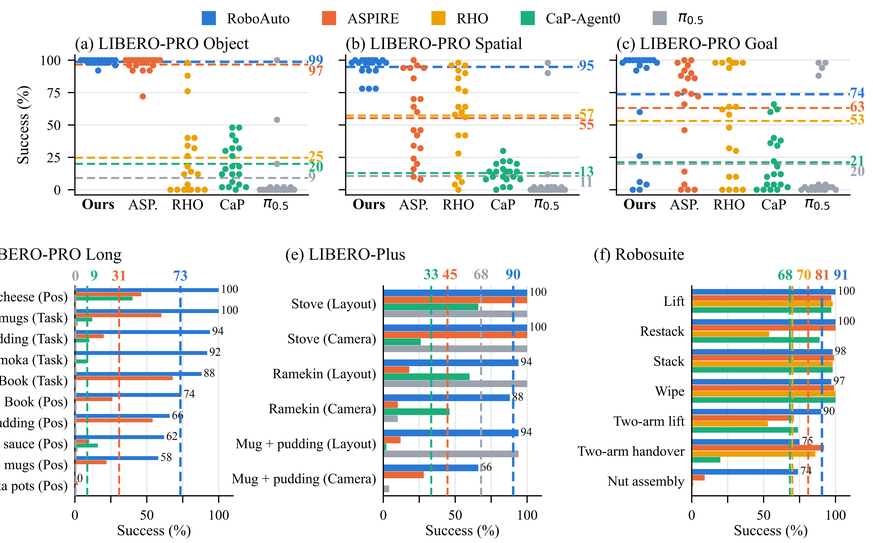
RoboAuto organizes physical understanding at **three levels**, each handled by one component: **objects**, what each kind of object affords and what an action must change; **scene**, the quantities the plan depends on, measured here and now; and **interaction**, whether each step produced its intended effect and, if not, why.

1. Object level: hierarchical knowledge base. A vision-language model watches robot demonstrations and distills reusable, object-centric knowledge: affordances such as "grasp a mug by the rim, not the handle," spatial relations, and action effects such as "a burner is on when its coil turns red." Programs that succeed are stored as executable skills. Entries record *which* quantities matter, never their values, so the same entry applies to any scene.

2. Scene level: physical scene grounding. For the task at hand, RoboAuto locates the relevant objects with Molmo2 pointing, SAM 3 segmentation, and depth cameras, then measures the open quantities each entry declares, such as a rim radius, a knob height, or a slot width, and moves the arm for a better view when the first observation is insufficient.

3. Interaction level: search and exploration. The task is planned as stages that each end in a checkable change of object state, searched sequentially so a verified prefix is kept and failures do not propagate. An independent auditor judges the videos and overrules optimistic self-checks. When the evidence does not explain a failure, the agent formulates competing hypotheses and writes a small **self-designed experiment**, such as trying three grasp poses, to decide between them; the evidence updates the knowledge entry.

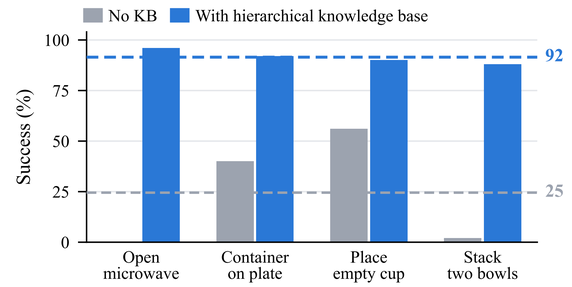
Simulation Benchmark Evaluation



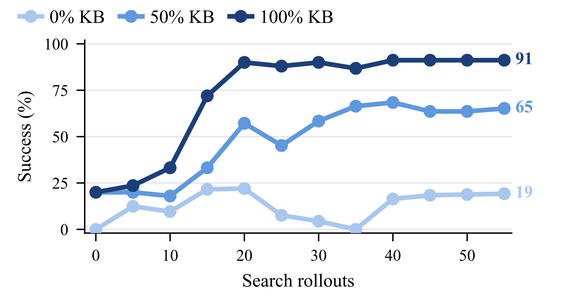
Mean success, %	**RoboAuto**	ASPIRE
LIBERO-PRO, 60 settings	**89.1**	71.7
LIBERO-PRO Long, 10	**73.4**	30.8
LIBERO-Plus, 6	**90.3**	44.7
Robosuite, 7	**90.6**	81.0

Setup. Benchmarks: LIBERO-PRO [5], LIBERO-Plus [6], Robosuite [7], RoboTwin 2.0 [8]. Baselines: ASPIRE, CaP-Agent0, RHO, $\pi_{0.5}$ [9]. Every Code-as-Policy method uses Claude Opus 4.6 as its coding agent and the same primitives: Molmo2 pointing, SAM 3 segmentation, Contact-GraspNet grasps, depth cameras. Search on 15 training scenes; final program tested on 50 disjoint scenes, 100 for Robosuite. **RoboAuto has the highest mean success on every family.**

Transfer and Scaling

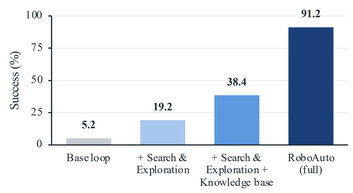


Transfer to a new simulator and embodiment. RoboTwin 2.0 changes the simulator from MuJoCo to SAPIEN, the robot to a bimanual arm, and the layouts, tasks, and primitives. Success rises from 24.5% without retained knowledge to 91.5% with the base learned in LIBERO: entries describe the physical requirements of an interaction, which RoboAuto re-grounds in the new scene and implements with the new robot's primitives.



Scaling with knowledge availability. With 0%, a random 50%, and 100% of the entries under the same budget, final success is 19.2%, 65.2%, and 91.2%. Half and full start at the same 20% but diverge as search proceeds, so additional entries help the agent interpret outcomes and choose revisions, not only write the first program. With the full base every task first reaches 80% within 19 rollouts.

Ablation Study



Components are added cumulatively to a base loop that rewrites the whole program from the results, on five tasks with the same seeds and budget. The base loop reaches 5.2%; Search and Exploration raises it to 19.2%; the Hierarchical Knowledge Base raises it to 38.4% by supplying strategies that parameter tuning never discovers; Physical Scene Grounding produces the largest gain, to 91.2%. Stored knowledge pays off only once it is grounded in the current objects and geometry.

References

[1] Liang et al. Code as Policies. ICRA 2023. [2] Fu et al. CaP-X. arXiv:2603.22435. [3] Lu et al. ASPIRE. arXiv:2607.00272. [4] Elmaaroufi et al. RHO. arXiv:2606.16458. [5] Zhou et al. LIBERO-PRO. arXiv:2510.03827. [6] Fei et al. LIBERO-Plus. CVPR 2026. [7] Zhu et al. robosuite. arXiv:2009.12293. [8] Chen et al. RoboTwin 2.0. arXiv:2506.18088. [9] Physical Intelligence et al. $\pi_{0.5}$. arXiv:2504.16054.