

FORCA: A Force-Aware Coding Agent for Contact-Rich Robot Manipulation

Xiao Tan^{1,*}, Yixuan Wang^{1,*}, Zhexi Wang^{1,2}, Wenzhen Yuan¹

Abstract—Coding agents can translate language instructions into robot programs. For contact-rich manipulation, a key challenge is to generate and refine control logic that responds to physical interaction. We present FORCA, a coding agent that takes visual observations and wrist force/torque (F/T) measurements as input to write Python programs for contact probing and force-responsive motion. Between executions, the agent inspects force records alongside visual outcomes to diagnose failures and revise both motion and force-feedback control logic. FORCA requires no demonstrations, task-specific training, or simulation. Across five real-robot tasks, we demonstrate its effectiveness in three forms of contact-rich manipulation: improving manipulation accuracy, adapting to changes in object or surface properties, and regulating contact under surface variation and object motion. Compared with a vision-only variant of the same agent, FORCA improves grasp accuracy and task success and reduces contact loss in wiping and peeling.

Index Terms—Force and tactile sensing, manipulation planning, embodied agents, contact-rich control.

I. INTRODUCTION

Coding agents based on foundation models turn open-ended instructions into executable robot programs that compose perception, grasping, and motion skills [1]–[4]. They ground their decisions mainly in vision and proprioception, which is often insufficient for contact-rich manipulation: occluded or reflective surfaces yield uncertain geometry, mass and friction are not visible, and maintaining contact requires responding to surface variation and object motion.

Force and contact measurements complement vision by providing evidence about contact states and mechanical responses that are difficult to infer from visual observations alone. Learned policies exploit force and tactile signals for reactive control but require demonstrations or robot data [5]–[8]. Foundation-model agents have used force through compliant or force-thresholded primitives [9], [10], inferred physical properties largely from priors [11]–[13], and revised force targets after failed attempts [14]. Supporting these diverse tasks requires agents to determine when contact information is needed, how it should guide ongoing motion, and how unexpected contact outcomes should inform subsequent attempts.

We present FORCA, a training-free coding agent that uses wrist force/torque (F/T) sensing in both *program execution* and

program revision (Fig. 1). It writes complete Python programs with access to the measured wrench. During a rollout, the program uses measurements from contact probes to guide subsequent actions and uses live force feedback to regulate its motion. Between rollouts, the agent relates the recorded force traces to visual outcomes, diagnoses failures, and revises the program.

FORCA addresses three types of contact-rich manipulation: (i) *precision manipulation*, where force cues help refine grasp poses and improve manipulation accuracy; (ii) *physical adaptation*, where actions must be adjusted to changes in object or surface properties; and (iii) *contact regulation*, where motion must respond to surface variation and object movement. We evaluate FORCA on five real-robot tasks spanning these categories: drawer opening, door-handle operation, cube-to-target knocking, vase wiping, and cucumber peeling. Compared with a vision-only variant of the same agent, FORCA increases task success from 40% to 100% for drawer opening and from 10% to 100% for door-handle operation, while reducing contact-loss rates in wiping and peeling from 40.6–55.9% to at most 2.2%.

Our contributions are twofold. First, we introduce a coding-agent framework that generates task-specific motion and force-feedback control logic from language instructions. The resulting programs use measured contact to refine grasp poses and regulate motion during execution. Second, we extend program refinement to the force-feedback logic itself, enabling coding agents to revise how their programs respond to contact based on evidence from real-robot execution.

II. RELATED WORK

Foundation-model agents for manipulation. Foundation models generate robot programs and plans from language [1], [2], ground manipulation in value maps or keypoint constraints [3], [4], and revise plans from scene descriptions, success detectors, or language feedback [15]; recent work benchmarks coding agents for manipulation [16] and refines programs or policies from real-robot experience [17], [18]. Their feedback is visual, proprioceptive, or linguistic. Agents that do use force expose compliant, force-thresholded primitives to code generation [9], call force-feedback routines from a knowledge base [10], or have a VLM output target wrenches that it revises after a failed attempt [14]. To our knowledge, FORCA is the first coding-agent framework to jointly generate task-specific force-feedback control logic and revise that logic between real-robot rollouts using recorded F/T traces and visual outcomes.

*Xiao Tan and Yixuan Wang contributed equally.

¹All authors are with the University of Illinois Urbana-Champaign, Urbana, IL, USA.

{tomtan2, yixuan41, zhexiw, yuanwz}@illinois.edu.

²Zhuxi Wang is also with Tsinghua University, Beijing, China. wangzhexi23@mails.tsinghua.edu.cn.

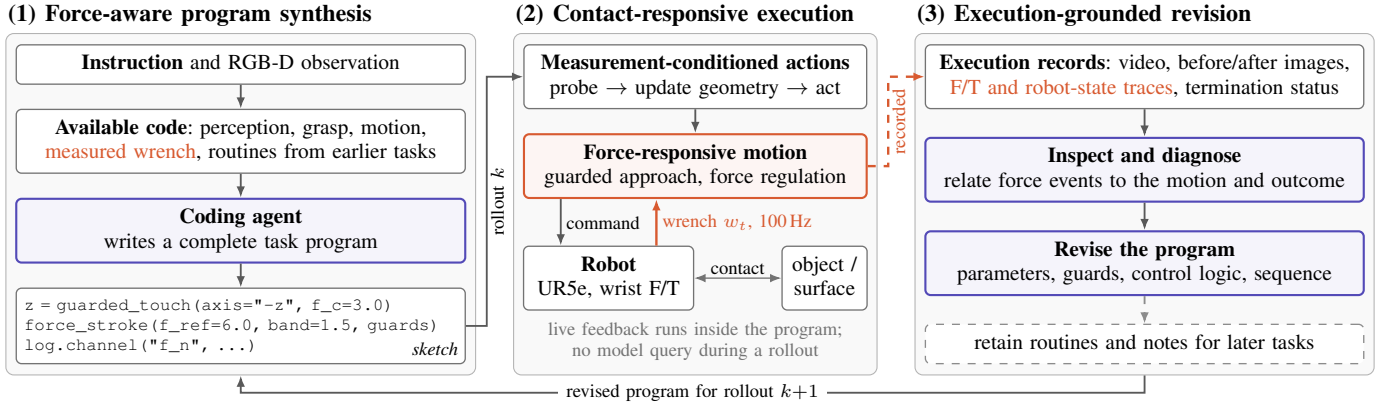


Fig. 1. **Overview of FORCA.** The agent generates a task program that uses force feedback during execution. Recorded force, robot-state, and visual observations inform revisions between rollouts. Generated routines and notes are retained for later reuse.

Contact-based perception and physical adaptation. Contact complements vision with local observations of geometry and interaction state: visuotactile estimators fuse vision, proprioception, and force under occlusion [19], and active tactile exploration selects informative contacts to estimate pose and shape [20]. Interaction also reveals latent properties, estimated by active force-based methods [21] or inferred across repeated trials by a learned memory [22]; these methods address a fixed estimation problem with exploration designed or learned for it. Foundation-model approaches infer properties largely from priors [11], [14], by matching observations in a physics engine [12], or through online system identification [13], and Matcha [23] lets an LLM choose exploratory actions for material recognition in simulation. In FORCA, the agent decides at deployment time when a visual estimate is unreliable and where to touch, and adapts a developed program to changed physical conditions from measured interactions, without task-specific training.

Force feedback for reactive manipulation. Force feedback lets robots adjust their motion as contact conditions change. It is widely used in compliant control and guarded motion [24]–[26]. Learned policies add force or tactile feedback to visuomotor control [5], [27]–[29] or fuse it into vision-language-action models [6]–[8]. Classical methods rely on thresholds and nominal traces specified per task, and learned policies require demonstrations or robot data. FORCA builds on the same control principles but has the agent write the force-responsive controller at deployment time and revise it from the recorded F/T trace.

III. METHOD

Given a language instruction and RGB-D observations, FORCA writes a Python code with access to wrist force/torque (F/T) measurements. The program executes on the robot without querying the foundation model. After execution, the agent reviews the recorded interaction and revises the code (Fig. 1). We refer to one execution of a program as a *rollout*.

A. Force-Aware Program Synthesis

For each task, the agent generates a Python task program that specifies the action sequence and the force-dependent motion logic. To construct this program, the agent has access to an initial skill codebase for perception, grasping, motion planning, and pick-and-place, assembled from open-source components under human supervision. These predefined modules provide object pose estimates, grasp candidates, and collision-free trajectories computed with PyRoKi [30]. Figure 2 illustrates these modules as the robot acquires a whiteboard eraser for the instruction “Wipe the vase.”

The agent writes the force-dependent logic for each task using wrist wrench measurements exposed by the robot interface. For example, this logic can specify when an approach should stop or how motion should respond to a force error. To implement these behaviors, the agent can modify existing routines or add measurement and control code as needed.

B. Contact-Responsive Execution

During a rollout, the generated program uses F/T measurements to guide subsequent actions and adjust ongoing motion. The conditions and feedback rules governing these responses are encoded in the program and executed without querying the foundation model.

Contact measurements can inform actions by refining the geometric estimates on which they depend. For example, the program can probe a surface before grasping, specifying both the probing motion and the force condition used to detect contact. The robot pose at contact, together with the tool geometry, provides an updated estimate of surface height or grasp position. Subsequent motions use this estimate within the same rollout.

For actions that require sustained contact, the program uses live force feedback to regulate motion. A host-side 100Hz loop reads the wrench and robot state over RTDE and updates Cartesian velocity commands according to the generated control logic. This logic can adjust motion based on the difference between measured and target force, or pause

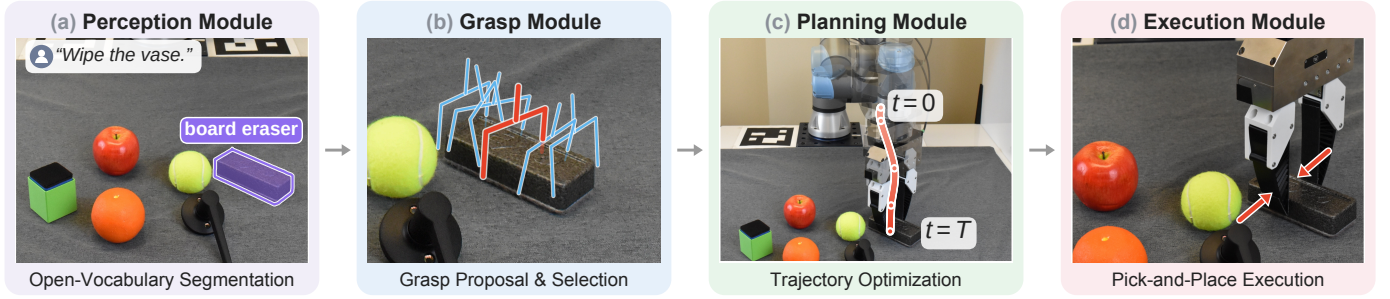


Fig. 2. **Predefined skill modules.** The agent is initialized with these modules, assembled from open-source components under human supervision. (a) SAM 3 [31] localizes the tool named in the instruction. (b) GraspGen [32] proposes candidate grasps, and the VLM selects one. (c) PyRoKi [30] plans the trajectory. (d) The robot executes the resulting routine to acquire the tool.

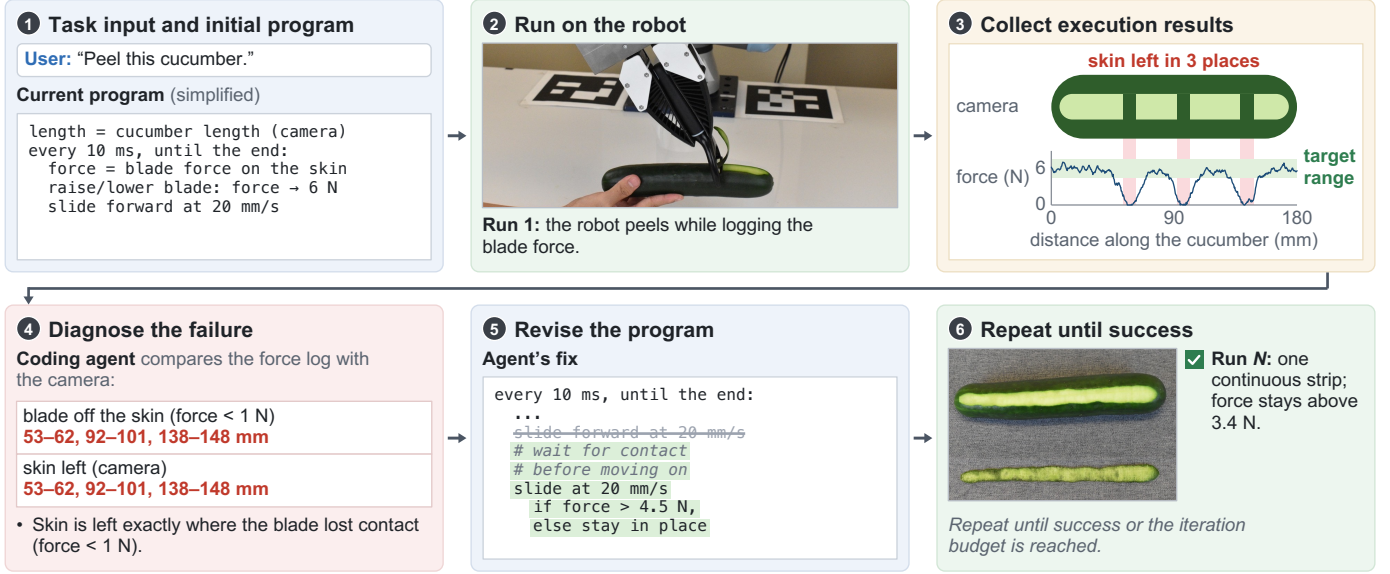


Fig. 3. **Illustrative program revision for cucumber peeling.** The agent compares force records with visible peeling gaps and adds a condition that pauses forward motion until contact is restored. Photos are from real runs; force traces and skin patches are idealized, and the code is simplified.

forward motion when contact is insufficient and resume it once contact is restored.

C. Execution-Grounded Program Revision

After each rollout, the agent inspects visual observations alongside the recorded force and robot-state traces. The agent can write analysis code to examine relevant time windows and relate force changes to robot motion and visible outcomes.

The agent uses these records to identify possible causes of failure and revise the relevant code. Changes can affect motion parameters or contact conditions, and can also modify the control logic or action sequence. Figure 3 illustrates one such revision in cucumber peeling. Skin remains exactly where the recorded force shows that the blade lost contact, so the agent makes the blade wait for contact before moving on.

Before execution, checks screen the revised code and planned motions for errors, reachability, and workspace violations. Revision stops when the agent declares the program converged or when either the rollout or wall-clock budget is exhausted. Development budgets and evaluation protocols are described in Sec. IV-A.

IV. EXPERIMENTS

We consider three ways in which force/torque feedback can benefit manipulation: (i) *improving manipulation accuracy*, by complementing visual estimates with contact-based geometric information to refine grasp poses and guide subsequent actions; (ii) *adapting to physical changes*, by providing evidence about how object properties affect action outcomes to adjust manipulation accordingly; and (iii) *regulating contact*, by using live force measurements to adjust ongoing motion and maintain contact as surfaces vary or objects move. Our experiments evaluate how effectively FORCA uses contact information across these three settings, combining comparisons against a vision-only agent with case studies of the generated programs. We additionally examine how FORCA revises its programs across iterations and the time and robot interactions required.

A. Experimental Setup and Protocol

Hardware and agent. Experiments use a UR5e arm with wrist force/torque sensing, a WSG-50 gripper, and a fixed RealSense

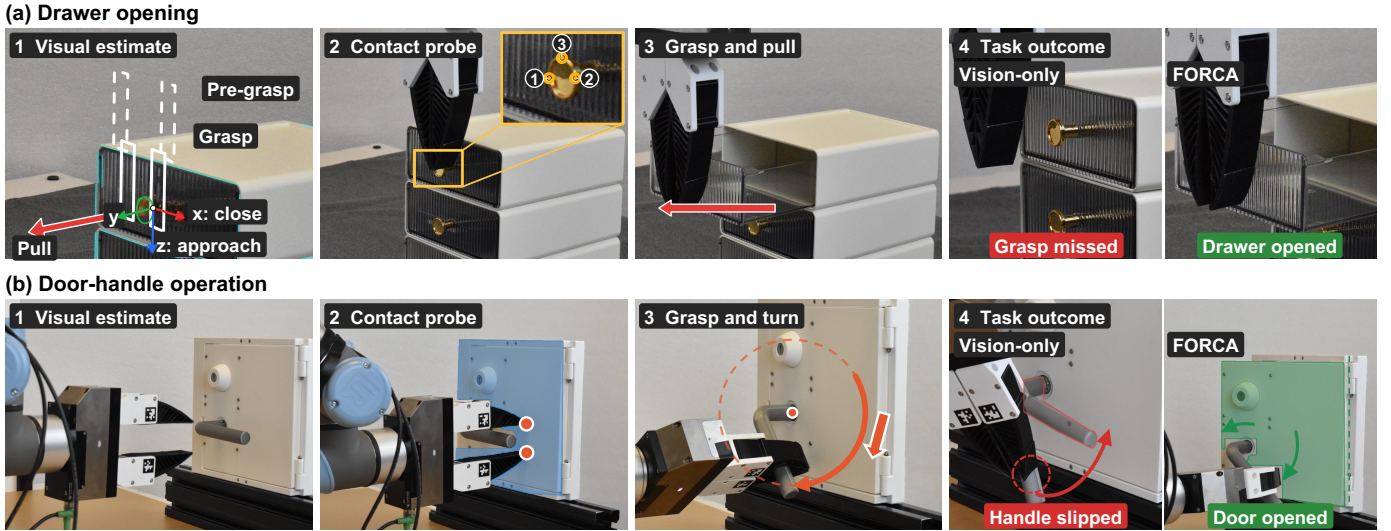


Fig. 4. **Example rollouts and outcome comparisons.** (a) Drawer opening and (b) door-handle operation. Panels 1–3 show FORCA estimating geometry visually, refining it through contact probing, and grasping to pull the drawer or turn the handle. In (a1), dashed and solid outlines mark the estimated pre-grasp and grasp poses; the axes indicate closing and approach directions. Numbered markers in (a2) identify three probe locations on the drawer knob. In (b2), orange markers indicate fingertip contacts, and blue shading marks the door-panel position confirmed by contact. Panels (a4) and (b4) compare example outcomes: the vision-only agent misses the drawer grasp and loses its grip on the door handle, while FORCA opens both mechanisms. The dashed circle and red arrow in (b4) mark the fingertip-only grasp and the handle’s rebound after slipping.

D455 RGB-D camera. The coding agent uses Claude Fable 5 for all analysis and code writing. The initial skills are described in Sec. III.

Execution and evaluation. The agent generates a complete task program, executes it, inspects the resulting records, and revises the program. One rollout consists of the agent writing or revising the task program and executing it once on the real robot. Each run is limited to 20 rollouts and 4 hours of wall-clock time. The run stops when the agent declares the program converged, after which we freeze the final program and run it over repeated trials and assess the physical task outcome; otherwise, it ends when either limit is reached. A rollout whose execution reaches the hard-coded 30 N force limit is stopped and counted as a failure. Human involvement is limited to execution authorization, scene operations, and final acceptance assessment.

Baseline policy. We compare FORCA with a **vision-only agent** that uses the same foundation model, task instructions, and basic skills, but has no access to force/torque data in program generation, revision, or execution: it receives no live or recorded wrench and no force-responsive control routines. Its programs guide motion with visual observations and robot state, and the agent can write visual feedback code and inspect visual outcomes when revising them. Robot-level safety protections remain active but do not expose force data to the agent, and the force recordings used for offline evaluation are withheld from it.

B. Results and Analysis

1) *Tasks Requiring Precise Manipulation* We evaluate whether force cues improve manipulation accuracy and task success on **drawer opening** and **door-handle operation**. Both

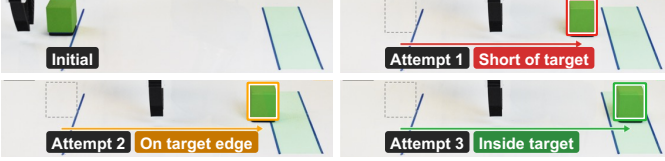
TABLE I
GRASP ACCURACY AND TASK SUCCESS FOR DRAWER OPENING AND DOOR-HANDLE OPERATION. GRASP ERRORS: MEAN $\pm$ STD. OVER 10 TRIALS.

Task	Method	Trans. err. (mm) $\downarrow$	Rot. err. ($^\circ$) $\downarrow$	Success (%) $\uparrow$
Drawer	Vision-only	8.3 ± 5.4	9.6 ± 3.2	40
	FORCA	5.3 ± 1.6	1.0 ± 0.3	100
Door	Vision-only	65.3 ± 16.9	55.5 ± 14.9	10
	FORCA	20.0 ± 0.9	0.6 ± 0.5	100

tasks require the robot to locate and grasp a handle before actuating the mechanism. The drawer is transparent, making its handle difficult to localize precisely from vision alone. Contact probing provides additional geometric information to refine the grasp pose. We compare FORCA with the vision-only agent using translational and rotational *grasp error* and *task success rate* (Table I). For each task and method, we execute the frozen program for 10 trials and report the success rate and grasp errors across these trials. FORCA achieves lower translational and rotational grasp errors and succeeds in all 10 trials on both tasks (100%), whereas the vision-only agent succeeds in only 4/10 trials on drawer opening and 1/10 on door-handle operation. Section IV-D examines how the door-handle programs were developed, with detailed failures and revisions documented in Table V.

2) *Tasks Requiring Adaptation to New Physical Conditions* We evaluate physical adaptation on a **cube-to-target** task, in which the robot knocks an object so as to make it slide to a target region. The robot must adjust its knocking motion to send the object far enough to reach the target without overshooting. The object’s mass and contact geometry affect its response

(a) Cube-to-target: landing position per attempt



(b) Object and surface variants

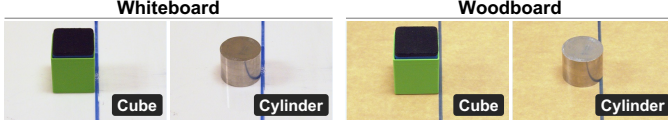


Fig. 5. **Cube-to-target adaptation.** (a) Object positions across successive attempts. The dashed outline marks the initial pose, and shading indicates the target region. (b) Object and surface configurations.

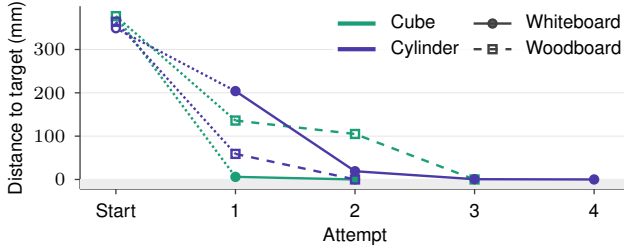


Fig. 6. **Reuse across physical conditions.** Distance to the target band per attempt (0: inside), starting from the program developed for the cube on the whiteboard.

to the impact, while friction with the supporting surface influences its subsequent deceleration and stopping distance. Consequently, the same knocking motion can produce different final positions when the object or surface changes.

We ask whether a program developed under one physical condition can be reused when the object’s response changes, and how many additional interactions it needs to complete the task again. We vary the object, using a cube and a cylinder that differ in both shape and mass (61.9 g and 663.6 g), and the surface, using a whiteboard and a woodboard that differ in friction (Fig. 5(b)). These variations change how far the object travels in response to a knock while preserving the task objective.

The evaluation has two stages. In the *development* stage, FORCA builds a knocking program from scratch for the cube on the whiteboard, the reference configuration (Table IV). In the *reuse* stage, we change the object or the surface and let FORCA continue from the developed program. After each attempt, it reviews the measured interaction forces together with the observed final object position and revises the program. Figure 6 shows the distance from the object to the target band after each attempt, and Fig. 5(a) shows illustrative landing positions across attempts. In each of the three new configurations, the retained program reached the target band within a few additional iterations, far fewer than developing it from scratch required (Table IV). The interaction experience of FORCA is thus kept in executable code, which remains usable, and can be revised, when physical conditions change.

TABLE II
VASE WIPING RESULTS AVERAGED OVER 10 TRIALS.

Vase	Method	Removal (%) $\uparrow$	Contact loss (%) $\downarrow$	Max. force (N) $\downarrow$
Static	Vision-only	73	42.6	11.8
	FORCA	99	0.7	11.3
Hand-held	Vision-only	21	55.9	17.5
	FORCA	93	1.6	12.3

TABLE III
PEELING A HAND-HELD CUCUMBER (AVERAGED OVER 10 TRIALS)

Method	Contact loss (%) $\downarrow$	Excessive force (%) $\downarrow$	Peeling ratio (%) $\uparrow$
Vision-only	40.6	13.4	37.4
FORCA	2.2	2.7	81.1

3) *Tasks Requiring Online Contact Regulation* We evaluate force-responsive execution on **vase wiping** and **cucumber peeling**. Wiping tests whether programs refined on a stationary surface remain effective when the object moves. Peeling tests whether the robot can maintain contact along an uneven surface while avoiding excessive force.

Vase wiping. The robot uses a whiteboard eraser to remove a marker line from a vase. We consider two settings: a *static* vase resting on a table and a *hand-held* vase that a person translates and rotates during wiping. We evaluate both methods after iterative program refinement in the static setting, then deploy the statically refined programs in the hand-held setting without further program revision.

Table II reports the *marker-removal ratio*, *contact-loss rate*, and *maximum force* for both methods in each setting. The marker-removal ratio is the percentage of the original mark removed, measured from before/after images. After iterative refinement, both methods achieve high marker removal on the static setup. The vision-only program improves by repeatedly adjusting trajectory offsets to the fixed surface, while FORCA uses force feedback to correct motion during contact. When transferred to the hand-held setting, the tuned offsets become inadequate as the vase moves, reducing removal and increasing contact loss. FORCA adjusts its motion to the moving surface and maintains more consistent contact. Figure 7 shows the wiping outcomes and corresponding force traces.

Cucumber peeling. The robot peels a longitudinal strip from a cucumber held by a person. We compare the vision-only agent and FORCA using *contact-loss rate*, *excessive-force rate*, and *peeling ratio* (Table III). The peeling ratio is the total newly peeled length divided by the intended strip length; separated peeled segments contribute their individual lengths, excluding unpeeled gaps. Excessive-force rate measures the proportion of active peeling time above an upper contact-force threshold, capturing sustained over-pressing that a peak-force value alone cannot characterize.

FORCA follows the cucumber surface with force-responsive motion and produces a more continuous peeled strip. The

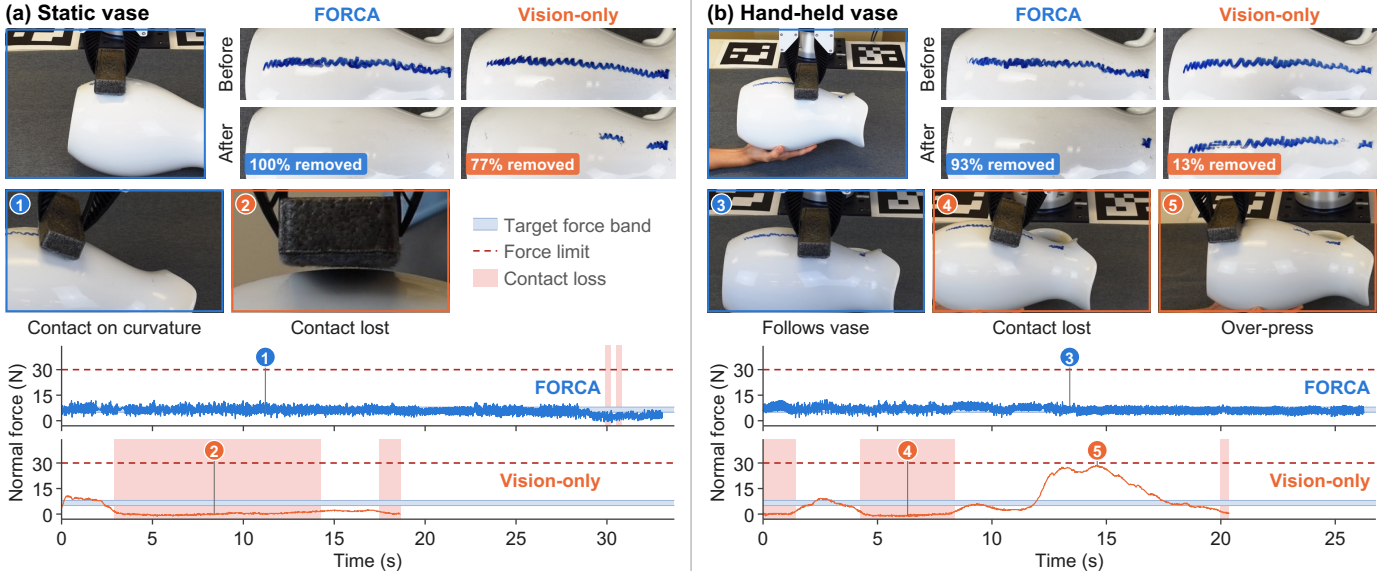


Fig. 7. **Vase wiping:** (a) static vase and (b) zero-shot transfer to a hand-held vase. Each panel shows representative outcomes (top), execution frames (middle), and force traces (bottom). Both programs are refined in the static setting and transferred without further revision. As the vase moves, FORCA maintains more consistent contact and high marker removal, whereas the vision-only program exhibits contact loss and large force variations, leaving substantial residual marks.

vision-only program exhibits contact interruptions and excessive pressing, leaving gaps in the peeled strip. Figure 8(a, b) shows representative peeling outcomes, and Fig. 8(c, d) relates the observed contact behavior to the force traces.

Contact metrics. Pressing force is the tool-normal force projection for wiping and the base-frame z force component for peeling. Contact-loss rate is the percentage of active task time spent in intervals where the pressing force stays below a contact threshold, 1.5 N for wiping and 0.8 N for peeling, for at least 0.2 s. Excessive-force rate uses $F_{\text{high}} = 10$ N, the upper threshold for acceptable peeling force, and maximum force is the largest pressing-force sample during wiping. Both rates include pauses for contact recovery and exclude the initial approach and deliberate lift-off. Force traces are recorded for offline evaluation with the same definitions for both methods; neither the vision-only agent nor its programs can access them.

C. Program Development Cost

Table IV reports the number of iterations FORCA used on each task and the mean time per iteration spent on agent reasoning and on robot motion. Agent reasoning takes most of each iteration, 4–10 min against 1–3 min of robot motion. Wiping and peeling used the full budget of 20 iterations.

D. Iterative Program Revision

During iterative revision, the agent uses execution feedback to diagnose failures and update its program between rollouts. To illustrate this process, we present door-handle operation as a case study and analyze one development run of each agent.

Door-handle case study. Figure 9 shows representative histories of iterative program development for FORCA and the vision-only agent. Each marker sits at the stage where that iteration’s execution failed, so reading the markers in order

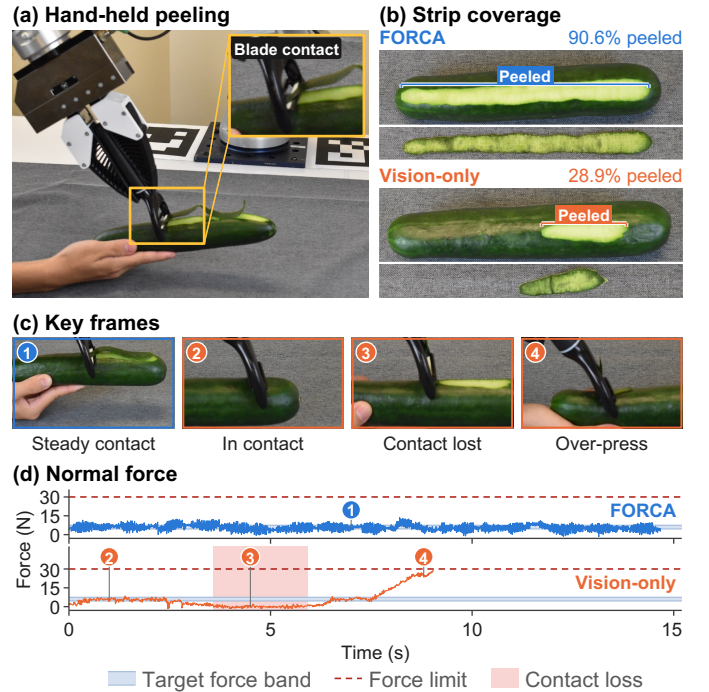


Fig. 8. **Cucumber peeling rollouts.** (a) Hand-held setup and a magnified view of the boxed blade–surface contact. (b) Peeled strips after one representative pass, covering 90.6% of the intended length for FORCA and 28.9% for the vision-only agent. (c) Execution frames at the marked instants. (d) Normal contact force. Shaded intervals indicate contact loss; the blue band marks the 4.5–7.5 N target range, and the dashed line marks the 30 N force limit.

shows how far each revision got. The dominant failure modes differ. FORCA failed mostly at *press down*: the handle turned but the latch did not release. It resolved this from the force recorded while pressing, where a sharp rise in resistance marks

TABLE IV
DEVELOPMENT ITERATIONS AND MEAN TIME PER ITERATION. FOR NEW CUBE CONFIGURATIONS, THE COUNT IS THE MEAN NUMBER OF REVISIONS AFTER THE FIRST EXECUTION, AVERAGED OVER THREE CONFIGURATIONS.

Task	Iterations	Agent reasoning	Robot motion
		(min / iteration)	(min / iteration)
Door handle	17	10	3
Drawer	8	5	1.5
Cube-to-target, from scratch	13	4	1
Cube-to-target, new condition	2	4	1
Vase wiping	20	4	1
Cucumber peeling	20	5	1.5

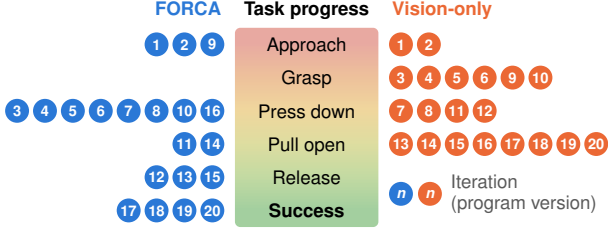


Fig. 9. **Door-handle development histories.** Each marker denotes an execution at its failure stage or at task completion. Executions 18–20 repeat FORCA’s final program. Detailed failures and revisions are listed in Table V.

the end stop, and then fixed the remaining failures in pulling and releasing. The vision-only agent failed mostly at *grasp* and *pull open*. It adjusted the grasp and commanded press angle using images alone and could press the handle down, but the handle repeatedly slipped out of the gripper before the door opened. Table V documents the failures and corresponding program revisions in both histories. The revision after execution 16 adds an angle condition for accepting an end stop and a force-drop condition before pulling. These checks change how contact evidence governs transitions between actions, illustrating refinement of the force-feedback logic itself.

V. LIMITATIONS

Our evaluation leaves several open questions about the cost and scope of deployment-time adaptation.

(i) *Iteration cost.* Each iteration takes several minutes, with agent reasoning accounting for most of the time (Table IV). The current refinement process also relies on real-robot roll-outs. Future work could investigate ways to reduce reasoning overhead and whether preliminary program refinement in simulation can reduce physical trials, while evaluating how well the resulting programs transfer to the robot.

(ii) *Contact sensing.* FORCA relies on wrist wrench measurements for contact feedback. These measurements capture the combined effects of contact but do not directly reveal where contact occurs or how forces are distributed across the fingers. This limits the contact information available to the agent. Future work could explore distributed tactile sensing and evaluate its usefulness for contact diagnosis and finer manipulation.

(iii) *Task complexity.* For tasks involving longer action sequences or tighter force constraints, the agent may struggle to diagnose failures and revise the program effectively, failing to produce a successful program even after repeated attempts. Future work could investigate whether improved reasoning and tool use help the agent solve such tasks.

VI. CONCLUSION

We presented FORCA, a coding agent that learns at deployment time by generating and revising robot programs using force feedback. Our real-robot experiments show that FORCA can refine geometric estimates through contact, adapt existing programs to changed physical conditions, and maintain contact during wiping and peeling, without task demonstrations or task-specific model training.

We view this as a step toward deployment-time learning in which physical experience accumulates in executable, inspectable programs. Over repeated deployments, an agent could retain useful contact routines together with evidence about when they work, then revise them as conditions change. Further progress requires more reliable reasoning from contact signals and a more efficient revision process, so that agents can solve complex tasks with fewer physical trials and manual resets. Policies learned through imitation or built on vision-language-action models could also serve as skills within these programs. The agent would coordinate sensing and skill execution, and revise the program based on execution feedback.

REFERENCES

- [1] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [2] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, “ProgPrompt: Generating situated robot task plans using large language models,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [3] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and F.-F. Li, “VoxPoser: Composable 3D value maps for robotic manipulation with language models,” in *Conference on Robot Learning (CoRL)*, 2023.
- [4] W. Huang, C. Wang, Y. Li, R. Zhang, and F.-F. Li, “ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation,” *arXiv preprint arXiv:2409.01652*, 2024.
- [5] Z. He, H. Fang, J. Chen, H.-S. Fang, and C. Lu, “FoAR: Force-aware reactive policy for contact-rich robotic manipulation,” *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5625–5632, 2025.
- [6] J. Yu, H. Liu, Q. Yu, J. Ren, C. Hao, H. Ding, G. Huang, G. Huang, Y. Song, P. Cai, C. Lu, and W. Zhang, “ForceVLA: Enhancing VLA models with a force-aware MoE for contact-rich manipulation,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- [7] P. Hao, C. Zhang, D. Li, X. Cao, X. Hao, S. Cui, and S. Wang, “TLA: Tactile-language-action model for contact-rich manipulation,” *arXiv preprint arXiv:2503.08548*, 2025.
- [8] J. Huang, S. Wang, F. Lin, Y. Hu, C. Wen, and Y. Gao, “Tactile-VLA: Unlocking vision-language-action model’s physical knowledge for tactile generalization,” *arXiv preprint arXiv:2507.09160*, 2025.
- [9] K. Burns, A. Jain, K. Go, F. Xia, M. Stark, S. Schaal, and K. Hausman, “GenChiP: Generating robot policy code for high-precision and contact-rich manipulation tasks,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 9596–9603.
- [10] R. Mon-Williams, G. Li, R. Long, W. Du, and C. G. Lucas, “Embodied large language models enable robots to complete complex tasks in unpredictable environments,” *Nature Machine Intelligence*, vol. 7, pp. 592–601, 2025.

TABLE V
DOOR-HANDLE ITERATIONS OF FORCA AND THE VISION-ONLY AGENT.

It.	Failed at	What happened	Revision
FORCA			
1–2	Approach	Camera put the door 3–4 cm too close; the probe missed it, then rejected a real touch.	Search farther; trust the touch and update the door depth.
3	Press down	First firm grasp, but at the 10 N force cap the handle turned only 18°, too little to unlatch.	Press along the handle’s arc about its axis.
4–5	Press down	Its own motion checks read slightly short moves as blocked; the handle never passed 5°.	Detect blocking by force; use the measured handle pose.
6–8	Press down	Still latched at 72.5°: grasp 21 mm and axis 15 mm off; handle twisted in the fingers.	Model the handle; press about a measured rotation axis.
9–10	Approach, press down	Setup faults: the cable plug, missing from the collision model, hit the door frame, and overly strict safety checks stopped motions; the press stopped at 49.7°.	Fix the setup; in the recorded press force, find latch release (61–65°) and end stop (jump past 74.5°); press to the stop.
11	Pull open	Latch released at 79.9°, but end-stop push-back used up the pull force budget.	Ease off one step from the end stop before pulling.
12–13	Release	Door opened (~26°); the handle-return move pushed on it; the retreat crossed the door.	Open the gripper and back straight out; replan the retreat.
14	Pull open	End stop not detected; push-back again used up the pull force budget.	Set the pull force limit relative to the push-back.
15	Release	Door opened and handle released; a leftover force reading blocked the retreat.	Re-zero the force sensor after letting go.
16	Press down	In one run, 49.7° was mistaken for the end stop; the robot pulled while still latched.	Accept end stops only past 65°; pull after the force drops.
17–20	Success	Door opened, handle released, and robot backed away in every rollout.	—
Vision-only			
1–2	Approach	A self-imposed ≥ 5 cm handle-standoff check failed at 4.9 cm; the robot never moved.	Lower the threshold.
3–6	Grasp	Bar landed at the fingertips (depth ~1 cm off, tilted gripper); it slipped at the first press.	Grasp deeper with less tilt; rebuild the handle model.
7–12	Press down	Fingers flexed, so a commanded 35° turned the handle only ~25°; the door stayed latched and the bar slipped out when pulled. In 9–10, the bar again landed at the fingertips.	Hook one finger behind the bar; raise the commanded press angle from 35° to 70°.
13–20	Pull open	Handle reached its end stop (~62°), but the bar slipped out of the grip during the pull.	Reorient and roll the gripper to hook the bar.

- [11] W. Xie, M. Valentini, J. Lavering, and N. Correll, “DeliGrasp: Inferring object properties with LLMs for adaptive grasp policies,” in *Conference on Robot Learning (CoRL)*, 2025, pp. 1290–1309.
- [12] A. Cherian, R. Corcoran, S. Jain, and D. Romeres, “LLMPhy: Parameter-identifiable physical reasoning combining large language models and physics engines,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2026.
- [13] B. Çiçek, M. K. Er, and Ö. S. Oğuz, “CoRAL: Contact-rich adaptive LLM-Based control for robotic manipulation,” in *Robotics: Science and Systems (RSS)*, 2026.
- [14] W. Xie, M. Conway, Y. Zhang, and N. Correll, “Unfettered forceful skill acquisition with physical reasoning and coordinate frame labeling,” 2025, arXiv:2505.09731.
- [15] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, “Inner monologue: Embodied reasoning through planning with language models,” in *Conference on Robot Learning (CoRL)*, 2023, pp. 1769–1782.
- [16] L. Fu, J. Yu, K. El-Refai, E. Kou, H. Xue, H. Huang, W. Xiao, G. Wang, D. Niu, F.-F. Li, G. Shi, J. Wu, S. Sastry, Y. Zhu, K. Goldberg, and L. Fan, “CaP-X: A framework for benchmarking and improving coding agents for robot manipulation,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2026, arXiv:2603.22435.
- [17] R. Lu, Y. Wu, E. Kou, L. Fu, W. Xiao, A. Mandlekar, Y. Xu, G. Shi, K. Goldberg, A. Chen, M. Chowdhury, Y. Zhu, L. Fan, and G. Wang, “ASPIRE: Agentic skills discovery for robotics,” *arXiv preprint arXiv:2607.00272*, 2026.
- [18] W. Xiao, J. Xie, T. Zhang, H. Lin, L. Fu, H. Xue, J. Lu, Y. Yang, C. Dai, Z. Wang, J. Wu, G. Wang, S. S. Sastry, K. Goldberg, L. Fan, Y. Zhu, and G. Shi, “ENPIRE: Agentic robot policy self-improvement in the real world,” *arXiv preprint arXiv:2606.19980*, 2026.
- [19] M. A. Lee, B. Yi, R. Martín-Martín, S. Savarese, and J. Bohg, “Multimodal sensor fusion with differentiable filters,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 444–10 451.
- [20] E. K. Gordon, B. Baraki, H. Bui, and M. Posa, “Active tactile exploration for rigid body pose and shape estimation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [21] J. Yuan, C. Choi, E. B. Tadmor, and V. Isler, “Active mass distribution estimation from tactile feedback,” *arXiv preprint arXiv:2303.01010*, 2023.
- [22] Y. Gao, J. Liu, S. Li, and S. Song, “Gated memory policy,” *arXiv preprint arXiv:2604.18933*, 2026.
- [23] X. Zhao, M. Li, C. Weber, M. B. Hafez, and S. Wermter, “Chat with the environment: Interactive multimodal perception using large language models,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Detroit, MI, USA, Oct. 2023, pp. 3590–3596.
- [24] N. Hogan, “Impedance control: An approach to manipulation: Part I—theory,” *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.
- [25] D. E. Whitney, “Quasi-static assembly of compliantly supported rigid parts,” *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 1, pp. 65–77, 1982.
- [26] P. M. Will and D. D. Grossman, “An experimental system for computer controlled mechanical assembly,” *IEEE Trans. Comput.*, vol. C-24, no. 9, pp. 879–888, Sep. 1975.
- [27] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu, “Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation,” in *Robotics: Science and Systems (RSS)*, 2025.
- [28] H. Fang, S. Tang, M. Mei, H. Qin, Z. He, J. Chen, Y. Feng, C. Wang, W. Liu, Z. He, C. Lu, and S. Wang, “Force policy: Learning hybrid force-position control policy under interaction frame for contact-rich manipulation,” in *Robotics: Science and Systems (RSS)*, 2026.
- [29] Z. Xu, Y. Wang, B. Abbatematteo, J. Preechayasomboon, S. Chan, N. Colonnese, and A. H. Memar, “Contact-grounded policy: Dexterous visuotactile policy with generative contact grounding,” in *Robotics: Science and Systems (RSS)*, 2026.
- [30] C. M. Kim, B. Yi, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa, “PyRoki: A modular toolkit for robot kinematic optimization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.
- [31] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. S. Coll-Vinent, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, J. Lei, T. Ma, B. Guo, A. Kalla, M. Marks, J. Greer, M. Wang, P. Sun, R. Rädle, T. Afouras, E. Mavroudi, K. Xu, T.-H. Wu, Y. Zhou, L. Momeni, R. Hazra, S. Ding, S. Vaze, F. Porcher, F. Li, S. Li, A. Kamath, H. K. Cheng, P. Dollár, N. Ravi, K. Saenko, P. Zhang, and C. Feichtenhofer, “SAM 3: Segment anything with concepts,” in *International Conference on Learning Representations*, 2026. [Online]. Available: <https://arxiv.org/abs/2511.16719>
- [32] A. Murali, B. Sundaralingam, Y.-W. Chao, W. Yuan, J. Yamada, M. Carlson, F. Ramos, S. Birchfield, D. Fox, and C. Eppner, “GraspGen: A diffusion-based framework for 6-DOF grasping with on-generator training,” *arXiv preprint arXiv:2507.13097*, 2025.