MAESTRO: Orchestrating Robotics Modules with Vision-Language Models for Zero-Shot Generalist Robots

Junyao Shi*, Rujia Yang*, Kaitian Chao*, Selina Bingqing Wan, Yifei Shao, Jiahui Lei, Jianing Qian, Long Le, Pratik Chaudhari, Kostas Daniilidis, Chuan Wen, Dinesh Jayaraman
*equal contribution
University of Pennsylvania
maestro-robot.github.io

***Abstract*— Today's best-explored routes towards generalist robots center on collecting ever larger "observations-in actions-out" robotics datasets to train large end-to-end models, copying a recipe that has worked for vision-language models (VLMs). We pursue a road less traveled: building generalist policies directly around VLMs by augmenting their general capabilities with specific robot capabilities encapsulated in a carefully curated set of perception, planning, and control modules. In MAESTRO, a VLM coding agent dynamically composes these modules into a programmatic policy for the current task and scenario. MAESTRO's architecture benefits from a streamlined closed-loop interface without many manually imposed structural constraints, and a comprehensive and diverse tool repertoire. As a result, it largely surpasses today's VLA models for zero-shot performance on challenging manipulation skills. Further, MAESTRO is easily extensible to incorporate new modules, easily editable to suit new embodiments such as a quadruped-mounted arm, and even easily adapts from minimal real-world experiences through local code edits. See our project site maestro-robot.github.io for videos and supplementary material.**

I. Introduction

The prevailing view in robotics today holds that achieving general-purpose capabilities requires training a single end-to-end model on massive, robotics-specific datasets, typically collected through labor-intensive manual teleoperation [1–3]. Compared to the abundance of text and image data available for language and vision models, the scarcity of robotics data is often cited as the key reason why robotic systems lag behind their generalist counterparts in these other domains. Hoping to bridge this gap, many efforts have been launched to massively scale up data collection for training vision-language-action (VLA) models.

Through careful experiments on a suite of challenging tasks, we demonstrate that, even without exploiting any of today's large robotics datasets, MAESTRO matches and in many cases surpasses state-of-the-art VLA models for zero-shot performance on tabletop manipulation skills, a domain that remains a benchmark for generalist robotic capabilities. MAESTRO therefore represents the **first competitive *modular* policy for generalist robotics**. In retrospect, this strength of MAESTRO is not surprising: it simply scales the traditional, tried-and-tested, paradigm of modular robotics system engineering by exploiting VLM capabilities to replace task-specific human engineering.

Going beyond its plain zero-shot capabilities, we show that MAESTRO inherits many of the advantages traditionally associated with manually engineered modular systems, such as interpretability, debuggability, and extensibility. For example, new tools, including VLA models themselves, can be integrated into MAESTRO's repertoire with minimal effort. Adapting the system to new embodiments such as mobile manipulators, or improving performance from limited experience, often requires only localized edits to the tool repertoire or policy code, rather than retraining or large-scale data collection. Moreover, in MAESTRO, robotics-specific data, training, and design are dealt with at the level of domain-specific modules, permitting learning from diverse pre-existing sources of training experiences (e.g., segmentation, pose estimation, grasp candidate generation datasets) rather than requiring all training data to fit into one common "observations-in-actions-out" straitjacket [4, 5].

These advantages do currently come at the cost of higher latencies and computational overheads compared to end-to-end VLA policies. We view this as a transitional limitation: as VLMs' inference hardware continue to evolve, we anticipate that systems like MAESTRO will become increasingly viable even for real-time, resource-constrained deployment—without sacrificing generality or adaptability. In the end, while more robotics data can only help, our results with MAESTRO suggest that it is far from the only way towards generalist robotic policies: pre-trained VLMs as robotic policies may offer a viable and attractive alternative route.

In summary, our contributions are:

1) We introduce MAESTRO, a novel VLM-driven agentic framework that leverage diverse robotics modules for general-purpose robot manipulation. We evaluate it extensively across tabletop and mobile embodiments, demonstrating that it outperforms state-of-the-art VLAs on diverse tasks.
2) We conduct systematic ablations to analyze design choices, providing insights into why MAESTRO substantially outperforms prior code-as-policy and VLA methods.
3) We demonstrate MAESTRO can further improve by evolving code programs from a small number of real-world trials.

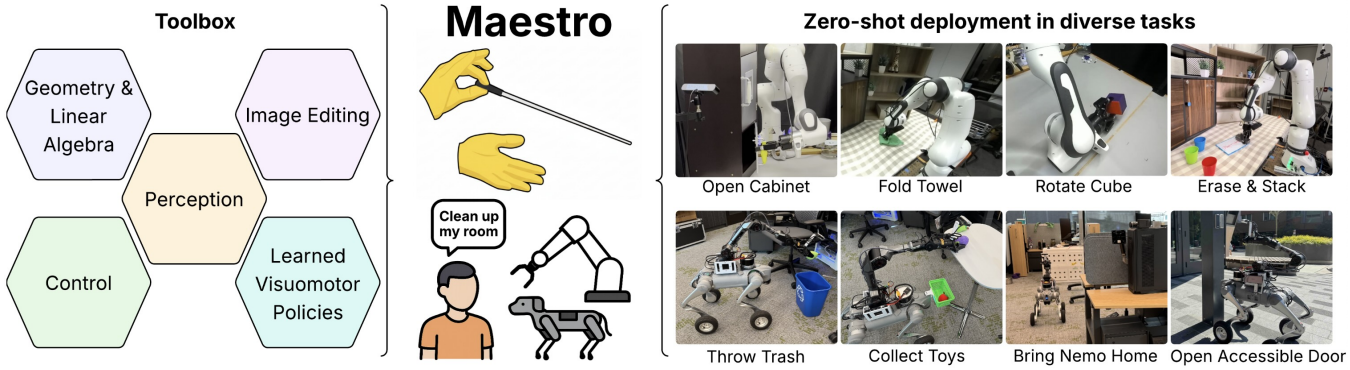


Fig. 1: **MAESTRO** receives language instruction and leverages a set of tools to complete diverse tasks in a zero-shot setting.

II. Related Work

We describe prior attempts at directly using VLMs and LLMs as robotic policies. See Supp. III on project site for extended discussion on prior work that utilizes VLMs and LLMs as modular components in robotics.

A. VLMs and LLMs as Robotic Policies

Modern vision-language-action (VLA) foundation models explicitly intended for robotics typically start from pre-trained LLM/VLM backbones before fine-tuning on large quantities of robotics-specific data [2, 6–12]. We are instead interested in studying the possibility of directly using off-the-shelf VLMs and LLMs in the robotic control loop.

Among such applications of "VLMs and LLMs as policies", directly generating low-level robotic actions has had limited success restricted to simple tasks [13], likely due to the large distribution shift from the web data used to train such models. Instead, the most successful approaches have converged to *generating "code as policies"*. The very first such approach CaP [14] demonstrated the remarkable capability of a text-based LLM to orchestrate a small set of perception and control APIs in a program to execute several robotic tasks. However, the program once generated is static and the LLM agent cannot respond to any unexpected scenarios that occur during execution — in this sense, a line of CaP work [14–16] is "open-loop".

More recent work [8, 17, 18] has explored the ability to "close the loop" with the VLMs, so that the robot actions within a trial do not have to be fully prescribed by one static program, by leveraging visual reasoning and closed-loop code generation based on visual feedback. While these results are encouraging, the general consensus within the field today is that off-the-shelf VLMs as generalist robotic policies are far behind their robot-data-trained VLA counterparts. A case in point is the recent release of Gemini Robotics VLA in March 2025 [2], which implements a closed-loop VLM-as-policy with a more capable perception API. While this is likely the most capable CaP-based system to date (we compare against our implementation of this system in our experiments), its performance is reported to be significantly inferior to their VLA model trained on tele-operation data. Broadly, dexterity and generalization have remained key challenges for such policies, particularly for tasks beyond pick-and-place on simple, symmetric objects.

We re-examine this consensus. We design a CaP system with two key characteristics: a more comprehensive set of robotics-relevant "tool" modules, and a simplified and streamlined closed-loop interface between the VLM and the APIs devoid of much manually imposed restrictive structure. Our choices permit the VLM to express itself more fully, and to benefit better from the best-in-class among tools produced by the large robotics research community over many years. As we will show in our experiments, the results surpass performance of today's state-of-the-art VLA models on challenging tasks at the frontier of today's generalist robotic capabilities.

B. Scaling up Data for Zero-Shot Robot Control

Data-driven approaches have recently become the dominant route toward general-purpose robotic manipulation. While some efforts have explored alternative data sources—such as simulation data [19–21] or human videos [22–24]—the most performant methods still rely on massive real-world teleoperation data [1–3, 10, 12], which are costly and labor-intensive to collect.

In our experiments, we adopt the state-of-the-art $\pi_{0.5}$ model [10] as a strong baseline. We demonstrate that, beyond simply scaling robot data, scaling the *right set of tools* for a robotics agent can also yield general-purpose manipulation capabilities. Furthermore, we show that MAESTRO can strategically leverage VLAs as callable tools in addition to its original set of tools, thus providing coverage in scenarios where VLAs struggle or face out-of-distribution inputs, while still maintaining the efficiency and strengths of VLAs themselves.

Taken together, these results indicate that large-scale robot data is not the only viable path to generalist robotic manipulation. By appropriately scaling the toolset and autonomy of code-based agents, it is possible to achieve and even surpass the performance of data-heavy approaches in zero-shot settings.

III. Method

We call our approach **M**anagerial **A**gent for **E**xecuting **S**ensorimotor **T**asks in **RO**botics, or **MAESTRO** for short. MAESTRO is a simple yet versatile robotic system centered on an agent that writes and executes code to leverage a rich

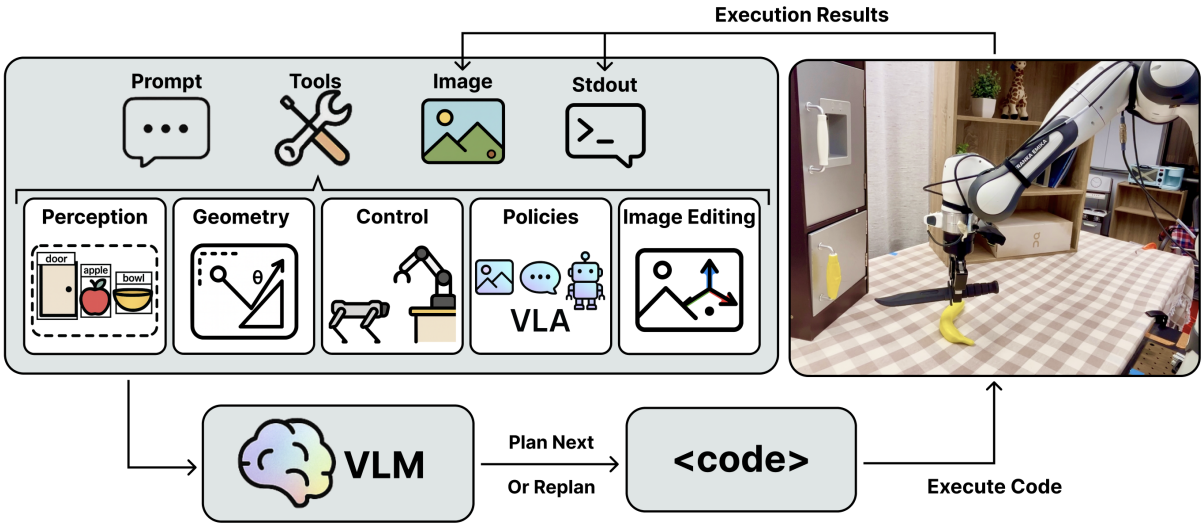


Fig. 2: Given prompt and images, VLM plans by writing and executing code that integrates perception, spatial reasoning, control, learned visuomotor policies, and image editing. Execution results (images and stdout) provide feedback for reacting and replanning, forming a closed-loop perception–action–learning cycle. This enables adaptive long-horizon manipulation, as illustrated in the tabletop example on the right (instruction: Grasp the knife by the handle and cut the banana in the middle).

toolkit spanning perception, geometry, control, pre-trained policies, and image editing. To specify a task, MAESTRO receives a system prompt, a scene image, and task instructions. Rather than invoking a VLM only once at the start of its attempt to perform the task, MAESTRO continually monitors the environment and calls the VLM as needed throughout execution, updating its code and actions in response to new observations and feedback in real time. We curate its set of tool modules to maximize coverage and capability, to provide a comprehensive foundation for manipulation tasks. This design forms an adaptive perception–action–learning loop. In the following sections, we detail the tool modules available (Sec. III-A), the monitoring system for closed-loop reaction and replanning (Sec. III-B), and the evolutionary improvement mechanism (Sec. III-C); an overview of this loop and the API is shown in Fig. 2.

A. *Principles for Building* MAESTRO *Module Toolset*

For tabletop manipulation, we experiment with the widely used DROID [29] platform visualized in Fig. 1, a 7-DoF Franka Panda robotic arm equipped with a Robotiq 2F gripper, supported by a wrist-mounted camera and a third-person camera. For this setup, Table I summarizes the modules present in MAESTRO compared to those used in prior work. Below, we highlight the key design principles that guided these choices (see Supp. I on project site for full technical details of each module).

"Coarse-to-fine" hierarchy of perception modules. Since different tasks require perceptual information about different regions of the scene at varying resolutions, we provide tools ranging from the fastest and simplest level (raw sensory input), medium level (mask centroid), to precise but slow (VLM-selected task-relevant keypoints). These tools give MAESTRO agency to autonomously select the right tools for the right uses, balancing execution speed and task performance.

Active perception module as an enabler improving other modules. Off-the-shelf tools, even when they are widely used and deployed, are often noisy and rely on acquiring the right informative observations of the scene. We believe that actively gathering better sensing (zoom in) or more sensory information (look around) with the wrist camera is essential for improving performances of vision-based tools (e.g. better point cloud for grasp model, better task-relevant keypoint selection) for downstream.

Geometry and linear algebra modules to scaffold spatial reasoning. Explicitly providing tools that construct vectors, measure Cartesian distances, measure rotation between two vectors, and compute vector rotation by an angle significantly improves MAESTRO's ability to reason step-by-step about object affordances and spatial relations.

Fast-inference VLM monitor enables VLA usage. Since VLAs run fast but are not trained to stop themselves after task completion, incorporating VLAs as modules into MAESTRO requires high-frequency interruption monitor. Locally hosted Qwen2.5-VL-72B-Instruct VLM is capable of generating "yes" or "no" output to check task-relevant conditions based on current image at 2HZ, allowing us to precisely interrupt VLA execution after completion or for replanning.

Collision avoidance key for object interactions. To perform robustly and generalizably in cluttered scenes, we add efficient point-cloud based collision-free motion planning.

Semantic map enables efficient long-horizon planning. By caching observed object locations, it supports persistent reasoning for mobile manipulation tasks.

B. *Plan, React, and Replan in a Loop*

Given a task instruction and image obvervation at the start of an episode, MAESTRO first ***plans***: it decomposes the task into smaller substeps and generates code for the

TABLE I: Comparison of modules used in prior work and in MAESTRO across tabletop and mobile manipulation. New or distinctive components in MAESTRO are shown in **bold**; cells marked *None* indicate no equivalent module in prior work.

Tool Category	Prior Work Examples	MAESTRO Modules
Tabletop Manipulation		
Perception	Raw sensory inputs (RGB + proprioception); Segmentation/Bounding Box centers [2, 6, 17]	Raw sensory inputs (RGB + proprioception); Segmentation centers; **Active perception (zoom/look around with wrist camera)**; **FoundationStereo [25] depth**; **Gemini pointing; VLM-selected task-relevant keypoints (ReKep-inspired [26])**
Control	Cartesian control, gripper control [2, 17]; Movement primitives [6]	Cartesian control, gripper control; **cuRobo collision-free motion planning**
Learned Visuomotor Policies	m2t2 grasp model [27]	**GraspGen grasp model [28]**; **$\pi_{0.5}$ VLA with high-frequency closed-loop monitoring (Qwen-2.5-VL)**
Geometry & Linear Algebra (new)	*None*	**Distance measurement, vector construction, vector rotation, relative rotation between vectors**
Image Editing (new)	*None*	**Draw points, overlay 6D poses to improve visual grounding**
Extra Modules for Mobile Manipulation		
Perception	Build global/local map [18]	Mobile base state estimation; **Active perception tools (look left/right/ground, view carry-on basket, log object location)**
Locomotion	Navigation [18]	Navigation; **Fine-grained "nudge" tool for local adjustment**

Initial Plan: Please finish the Receive Instruction, Describe the Scene, and Steps Planning process. At the end, you should enclose the first step code to be executed.

Robot Task Instructions: You are a helpful franka panda robot arm...

Task Execution Procedure: <1. Receive Instruction> ··· <2. Describe the Scene> ··· <3. Steps Planning> ··· <4. Steps Execution>

Robot Hardware and Physical Constraints: world frame orientation...

Robot API Documentation:

```
class RobotApi: """class variable defined that can be accessed at any point of the program""":
    def get_task_relevant_keypoints(self, object_name, prompt) → List[3d_points]
    def move_gripper_to(self, position, orientation)
    def get_grasp_pose(self, object_name, subtask_instruction, point_to_grasp)
    def rotate_by(self, current_orientation, rotation)
    ···
class ImageEditApi: """Interface for editing the image for a better understanding."""
    ···
```

Example Code

Here is an example code for the task 'Pick up the red cube and rotate.' You should learn how and when to use the functions in the API and try to imitate the code to complete the new task.

```python
points = robot.get_task_relevant_keypoints("red cube", "center of red cube")
grasp_results = robot.get_grasp_pose("cube", "pick up cube", points[0]['3d_position'])
···
target_rotation = robot.rotate_by(curr_orientation, [0, -np.pi, 0])
```

Replan: Describe the execution result and compare it with the goal and the expert demo, is this step successful? If successful, output the code for next step. If not, summarize what caused the failure, find the problem of your code, write code to return to some free state, and then rewrite a better code for the task.

Fig. 3: A summarized overview of MAESTRO's system prompt.

initial substep. After executing the code of the first substep, MAESTRO ***reacts***: it ingests the original instruction, code output, robot state, and the images from the last substep to assess whether the subgoal has been achieved. If successful, it proceeds to ***plan*** again by generating code for the next substep; if not, MAESTRO ***replans***: it diagnoses the likely cause of failure and rewrites improved code for the same substep. The ***plan***, ***react***, ***replan*** thus proceeds in a loop, allowing MAESTRO to continuously adapt its behavior to changes in the environment or its own mistakes until the overall task is complete. Note that in mobile manipulation settings, before performing failure analysis and rewriting code, MAESTRO is also prompted to actively look around and perceive the environment to build a more complete situational understanding. See Fig. 3 for a schematic of MAESTRO's system prompts for both initial plan generation and reacting to plan next step or replan.

C. Evolution Based on Previous Runs

Our evolution mechanism builds on a database that logs all past task executions. After each run, we store the generated code, standard output, and Gemini's success/failure analysis of the execution video. Before each new run, this accumulated record is supplied to Gemini as in-context examples, enabling it to draw on prior successes and failures to refine its code generation and improve performance over time.

IV. Experiments

Our experiments aim to study the following research questions:

- How well does MAESTRO perform zero-shot on various embodiments in various settings for various tasks, compared to state-of-the-art VLA and CaP generalist policies?
- Which system design choices are most important for MAESTRO's performance?
- Can MAESTRO improve from a small number of real-world trials, using the approach described in Sec. III-C?

A. Real-World Experiment Setup

We conduct a diverse set of real-world evaluations to demonstrate MAESTRO's versatility and robustness. Our

experiments are designed to showcase its ability to generalize across three critical axes: category of manipulation challenge, evaluation setting diversity, and robot embodiment. By varying these dimensions, we highlight how MAESTRO adapts to different hardware platforms, performs a wide range of manipulation skills, and maintains stable performance across distinct real-world contexts.

Embodiments. We evaluate MAESTRO across two distinct robot embodiments — one for tabletop manipulation and one for mobile manipulation — to rigorously test its generality. Our tabletop platform follows the DROID setup [29]: it is a 7-DoF Franka Emika Panda robotic arm equipped with a Robotiq 2F gripper, supported by a wrist-mounted camera and a third-person camera. This platform best supports experiments comparing MAESTRO to previously proposed generalist policies amongst which tabletop manipulation is the most widely studied task domain. We also deploy MAESTRO on a Unitree Go2-W wheeled quadruped outfitted with an AgileX Robotics PiPER manipulator arm mounted on top and a calibrated wrist-mounted camera for egocentric perception.

Tasks. We identify key axes of challenges for generalist robot policies and consolidate 5 tabletop and 4 mobile manipulation tasks. See Sec. IV-B and IV-C for details.

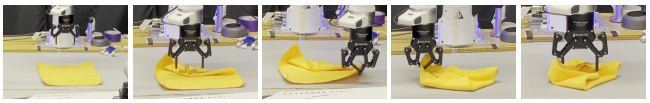
Evaluation Protocol. To ensure systematic experimentation, we adopt the STAR-Gen taxonomy of generalization for robot manipulation [30]. STAR-Gen formalizes generalization through systematic perturbations relative to a base task. Using the scenario generation tool provided by STAR-Gen, we prompt Gemini to generate perturbed task instances along four axes: ***visual changes to task-relevant objects***, ***changes to object poses***, ***changes to action verbs requiring new behavior***, **and** ***introducing entirely new manipulated objects***, resulting in a total of 5 evaluation trials for each task, held fixed across all compared methods. This approach ensures that every trial of our evaluation differs substantially and meaningfully from the rest, capturing realistic in-the-wild diversity and providing a rigorous test of MAESTRO 's robustness and adaptability. Additionally, since both MAESTRO and our baselines are capable of retries, we set a time limit for each task based on the task horizon. Following π_0 [1], we design **a score rubric that measures progress** on each task for quantitative results, see Supp. II on the project site for details.

B. Zero-Shot Tabletop Manipulation Results

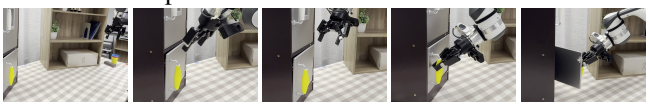
Baselines. We benchmark MAESTRO against the strongest generalist robot policies from both the Code-as-Policies (CaP) and VLA paradigms. For CaP, we adopt the approach recently described in the Gemini Robotics technical report [2], which uses Gemini to enable zero-shot robot control via code generation. While it has a similar closed-loop replanning structure, it contains only a limited set of simple tools (see Table I), thus restricting its capability to pick-and-place of simple, symmetric objects. We implement this using the latest Gemini 2.5 Pro and denote it as the **Gemini Robotics Agent**. For VLAs, we compare against the state-of-the-art models π_0 [1] and $\pi_{0.5}$ [10]. Specifically, we use the π_0-FAST-DROID checkpoint for π_0—π_0-FAST model fine-tuned on the DROID dataset—and the $\pi_{0.5}$-DROID checkpoint for $\pi_{0.5}$, which is similarly fine-tuned. We also include **MAESTRO + $\pi_{0.5}$**, where $\pi_{0.5}$ is incorporated as a callable module within our framework, allowing MAESTRO to leverage it dynamically.

Pick–Place: pick up item and put into the bowl

Deformable: fold cloth, four corners into the center



Articulated: open cabinet



Spatial reasoning: rotate cube purple side up

Memory: erase stacking instructions, then stack cups

Fig. 4: **Tabletop manipulation evaluation tasks.**

Tasks. We evaluate MAESTRO on five tasks, visualized in Fig. 4, that reflect the key challenge axes facing today's generalist tabletop manipulation policies: *pick-place*—**put item in bowl**; *deformable object*—**fold four corners of the towel into the center**; *articulated object*—**open cabinet**; *spatial reasoning*—**rotate cube to purple side up**; *memory & long-horizon semantic reasoning*—**erase the whiteboard instructions, then stack cups in the specified order**.

Results. Table II summarizes the tabletop manipulation results. Across all five tasks, MAESTRO substantially outperforms every baseline. This performance gap is most evident in tasks demanding semantic reasoning or trials with STAR-Gen semantic perturbations: while VLA baselines frequently fail under major changes in background and instructions, VLM-based agents remain robust. VLAs also lack any explicit memory mechanism, resulting in poor performance on memory tasks, and their training data, dominated by pick-and-place scenarios, leads to near-zero progress on more out-of-distribution tasks such as open cabinet or rotate cube.

VLM-based agents, by contrast, excel at high-level semantic reasoning and planning. However, our CaP baseline, Gemini Robotics Agent, still struggles to turn these high-level plans into effective low-level actions because it lacks modules for precise perception and control. It performs reasonably on simple pick-and-place tasks but fails in settings requiring richer visual information processing or specialized actions. For example, in the open cabinet task it cannot localize the handle without active perception, and its top-down grasping strategy no longer applies. Similar issues occur in fold towel

TABLE II: Tabletop manipulation results: average task progress (0–100; higher is better) across methods.

Challenge	Task Description	Gemini Robotics Agent	π_0	$\pi_{0.5}$	MAESTRO
Pick–Place	Put item in bowl	73.3 ± 46.2	74.0 ± 37.1	70.0 ± 41.1	$\mathbf{98.0 \pm 4.5}$
Deformable object	Fold the four corners of the towel into the center	40.0 ± 17.3	47.0 ± 25.1	70.0 ± 15.4	$\mathbf{71.3 \pm 21.4}$
Articulated object	Open cabinet	3.3 ± 5.8	8.3 ± 2.9	0.0 ± 0.0	$\mathbf{68.0 \pm 31.3}$
Spatial reasoning	Rotate cube purple side up	23.6 ± 3.5	29.0 ± 1.7	10.0 ± 0.0	$\mathbf{60.0 \pm 38.1}$
Memory & long-horizon & semantic	Erase instructions on whiteboard, then follow instruction to stack cups	26.7 ± 24.7	12.0 ± 12.0	22.0 ± 22.8	$\mathbf{63.0 \pm 16.8}$

TABLE III: Mobile manipulation results: average task progress (0–100; higher is better) for MAESTRO.

Task Category	Task Description	MAESTRO
Long-horizon manipulation	Collect all toys on table	85.0 ± 22.4
Long-horizon loco-manipulation	Throw green ball into garbage can	76.7 ± 14.9
Active exploration	Search item and put on table	96.0 ± 8.9
Object affordance	Press button to open door	93.3 ± 14.9

and erase whiteboard, where Gemini Robotics Agent can plan but cannot localize towel corners or correctly grasp and orient the eraser.

MAESTRO bridges this gap by coupling VLM-level semantic reasoning with a broad suite of specialized tool modules. It leverages grasp models for reliable pick-and-place, active perception modules for accurate point-cloud reconstruction of cabinet handles, task-relevant keypoints for towel corner localization, and geometric reasoning to rotate objects to precise orientations (e.g., turning a cube to place a target color face upward). When errors occur, MAESTRO 's "react and replan" mechanism analyzes execution history and images, then revises its code and module orchestration. For example, to pick up the tennis ball on the shelf, it first attempts a top-down grasp but, after detecting rolling and collision risks, switches to active perception to refine the point cloud and leverages the grasp model tool to generate a safer, reachable grasp pose.

By orchestrating its diverse tools in a "plan, react, replan" loop, MAESTRO fuses the semantic reasoning strength of VLMs with the precision and reliability of specialized modules. This enables it to surpass prior CaP systems in dexterity and task coverage, performing tasks that were previously challenging for code-as-policies approaches and typically considered better suited to VLA-style approaches.

C. Zero-Shot Mobile Manipulation Results

Tasks. Similar to the tabletop case, we evaluate on four tasks, visualized in Fig 5, that demonstrate key challenge axes for mobile manipulation: *long-horizon manipulation*—**collect all toys on table**; *long-horizon loco-manipulation*—**throw green ball into garbage can**; *active exploration*—**search for item and return**; *object affordance reasoning*—**press button to open door**.

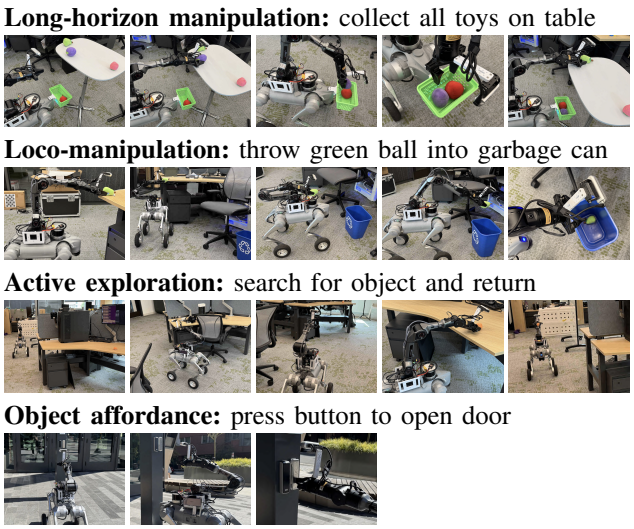


Fig. 5: **Mobile manipulation evaluation tasks.**

Results. The two *long-horizon* tasks show lower progress rates due to their multi-stage object interactions. A cached semantic map substantially improves performance by allowing the agent to re-track objects and complete sub-tasks without redundant search. Remaining failures primarily stem from low-level execution. The *throwing trash* task achieves 76.7% due to occasional inaccurate depth estimates of the garbage can. This led to invalid grasp poses that violated the IK constraints, causing aborted motions. Likewise, the reactive replanning mechanism sometimes entered oscillatory loops when no collision-free path was found. In contrast, *active exploration* shows high progress as we provide multi-view images at replan sessions to improve spatial reasoning. The *press button* task benefits from precise keypoints selection, reliably identifying pressing location from image input.

D. How important is each component of MAESTRO?

While Table I details the comprehensive list of improvements and additions we made over prior CaP systems, in the following experiments, we focus on two key category of design choices among them. **MAESTRO w/o advanced perception** evaluate the impact of *task-relevant keypoint* module and *active perception* module. **MAESTRO w/o geom-**

TABLE IV: Ablation results on the Fold Towel and Rotate Cube tasks (average task progress out of 100).

Method	Fold Towel	Rotate Cube
MAESTRO	71.3 ± 21.4	60.0 ± 38.1
MAESTRO w/o advanced perception	40.0 ± 7.1	25.0 ± 0.0
MAESTRO w/o geometry modules	67.5 ± 3.5	42.5 ± 31.8

etry modules tests the impact of the *geometry* and *linear algebra* modules. For systematic comparison, we fairly isolate each factor by following a "change one at a time" approach, creating variants where only one module category is reverted to the prior baseline while keeping all other components identical to MAESTRO.

Task. We use the "**fold towel**" and "**rotate cube**" tasks. In fold towel, robot must fold the corners into the center, thus the task requires reasoning about object geometry and specific points of interaction on the object. In rotate cube, the robot must rotate it until a particular color side faces up, requiring extensive spatial reasoning about rotation and object affordance.

Results. Table IV shows that both key module components are essential for strong performance across tasks. For example, the fold towel task requires precise interaction at the towel's corners, and **MAESTRO w/o advanced perception** fails to identify the correct interaction points. Likewise, the rotate cube task depends on constructing vectors based on task-relevant keypoints on the cube and the gripper to compute rotations, and therefore neither **MAESTRO w/o advanced pereption** nor **MAESTRO w/o geometry modules** can achieve progress. Overall, these results highlight that our carefully curated modules such as advanced perception and geometry tools are essential for MAESTRO to achieve high manipulation accuracy and robust performance across diverse tasks.

E. MAESTRO *improves from evolution across trials*

Using the method described in Sec. III-C, we evaluate MAESTRO 's ability to improve over successive trials by evolving its code from prior attempts on the same task. Starting with an the least successful run in our open-cabinet experiments, where MAESTRO achieved only 35% progress: it correctly identified the cabinet handle but attempted a top-down grasp, which failed. After one evolutionary update, MAESTRO adjusted its behavior, scanning around the handle and leveraging its grasp model to successfully grasp but still pulled the handle straight rather than along its rotation axis, reaching 70.0 ± 5.0 progress. By the third evolution, MAESTRO corrected this mistake, using vectors constructed from handle and hinge keypoints to compute the correct rotation and apply it, achieving 85.0 ± 7.4 progress.

V. Conclusion

We present MAESTRO, a simple yet powerful VLM-driven agent that orchestrates diverse robotic tools for general-purpose manipulation across both tabletop and mobile settings. Without relying on any robot training data, MAESTRO consistently outperforms state-of-the-art VLAs and prior CaP systems. Its performance will continue to scale with advances in both robotic modules and the underlying VLM.

Despite these strengths, MAESTRO still has limitations. Achieving more delicate and continuous control will require richer low-level behaviors than currently supported. Additionally, VLM API response times introduce pauses when MAESTRO reacts and replans, prolonging its overall runtime. Addressing these challenges is a focus for future work.

We view runtime as a transitional limitation: recent advances in VLM optimization, model distillation, and efficient code generation are already closing the gap in responsiveness and resource efficiency. As VLMs continue to evolve, we anticipate that modular systems like MAESTRO will become increasingly viable for a real-time deployment with limited resources – without sacrificing generality or adaptability. In fact, in the long run, modular generalist policies like MAESTRO may be *more resource-efficient*, due to only requiring resource allocation to match the current demands of the task, unlike monolithic one-size-fits-all VLA models.

References

[1] K. Black, N. Brown, *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[2] G. R. Team, S. Abeyruwan, *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[3] J. Bjorck, F. Castañeda, *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[4] A. Khazatsky, K. Pertsch, *et al.*, "DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset," *RSS*, 2024.

[5] L. collaboration, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," *ICRA*, 2024. [Online]. Available: `https://robotics-transformer-x.github.io/`.

[6] M. Zawalski, W. Chen, *et al.*, "Robotic control via embodied chain-of-thought reasoning," *arXiv preprint arXiv:2407.08693*, 2024.

[7] W. Chen, S. Belkhale, *et al.*, *Training strategies for efficient embodied reasoning*, 2025. arXiv: `2505.08243 [cs.RO]`. [Online]. Available: `https://arxiv.org/abs/2505.08243`.

[8] L. Shi, B. Ichter, *et al.*, "Teaching robots to listen and think harder," 2025, Published February 26, 2025; Physical Intelligence (Hi Robot project). [Online]. Available: `https://www.physicalintelligence.company/research/hirobot`.

[9] Q. Zhao, Y. Lu, *et al.*, *Cot-vla: Visual chain-of-thought reasoning for vision-language-action models*, 2025. arXiv: `2503.22020 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2503.22020`.

[10] P. Intelligence, K. Black, *et al.*, *$\pi_{0.5}$: A vision-language-action model with open-world generalization*, 2025. arXiv: 2504.16054 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2504.16054.

[11] J. Bjorck, F. Castañeda, *et al.*, "Gr00t n1.5: An improved open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025, GR00T N1.5 version; NVIDIA Labs. DOI: 10.48550/arXiv.2503.14734. [Online]. Available: https://research.nvidia.com/labs/gear/gr00t-n1_5/.

[12] J. Lee, J. Duan, *et al.*, *Molmoact: Action reasoning models that can reason in space*, 2025. arXiv: 2508.07917 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2508.07917.

[13] T. Kwon, N. D. Palo, and E. Johns, "Language models as zero-shot trajectory generators," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6728–6735, Jul. 2024, ISSN: 2377-3774. DOI: 10.1109/lra.2024.3410155. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3410155.

[14] J. Liang, W. Huang, *et al.*, "Code as policies: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[15] I. Singh, V. Blukis, *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530. DOI: 10.1109/ICRA48891.2023.10161317.

[16] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Proceedings of the 2023 Conference on Robot Learning*, 2023.

[17] J. Duan, W. Yuan, *et al.*, "Manipulate-anything: Automating real-world robots using vision-language models," *arXiv preprint arXiv:2406.18915*, 2024.

[18] P. Zhi, Z. Zhang, *et al.*, *Closed-loop open-vocabulary mobile manipulation with gpt-4v*, 2025. arXiv: 2404.10220 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2404.10220.

[19] M. Dalal, M. Liu, *et al.*, "Local policies enable zero-shot long-horizon manipulation," *International Conference of Robotics and Automation*, 2025.

[20] T. G. W. Lum, M. Matak, *et al.*, "DextrAH-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=S2Jwb0i7HN.

[21] T. Lin, K. Sachdev, *et al.*, "Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids," *arXiv:2502.20396*, 2025.

[22] J. Shi, Z. Zhao, *et al.*, "Zeromimic: Distilling robotic manipulation skills from web videos," in *International Conference on Robotics and Automation (ICRA)*, 2025.

[23] R. Yang, Q. Yu, *et al.*, *Egovla: Learning vision-language-action models from egocentric human videos*, 2025. arXiv: 2507.12440 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2507.12440.

[24] L. Y. Zhu, P. Kuppili, *et al.*, *Emma: Scaling mobile manipulation via egocentric human data*, 2025. arXiv: 2509.04443 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2509.04443.

[25] B. Wen, M. Trepte, *et al.*, "Foundationstereo: Zero-shot stereo matching," *arXiv*, 2025.

[26] W. Huang, C. Wang, *et al.*, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," *arXiv preprint arXiv:2409.01652*, 2024.

[27] W. Yuan, A. Murali, *et al.*, "M2t2: Multi-task masked transformer for object-centric pick and place," in *7th Annual Conference on Robot Learning*, 2023.

[28] A. Murali, B. Sundaralingam, *et al.*, "Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training," *arXiv preprint arXiv:2507.13097*, 2025. [Online]. Available: https://arxiv.org/abs/2507.13097.

[29] A. Khazatsky, K. Pertsch, *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," 2024.

[30] J. Gao, S. Belkhale, *et al.*, "A taxonomy for evaluating generalist robot policies," 2025.