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 [1–4]. 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 [5, 6].

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 continues 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.

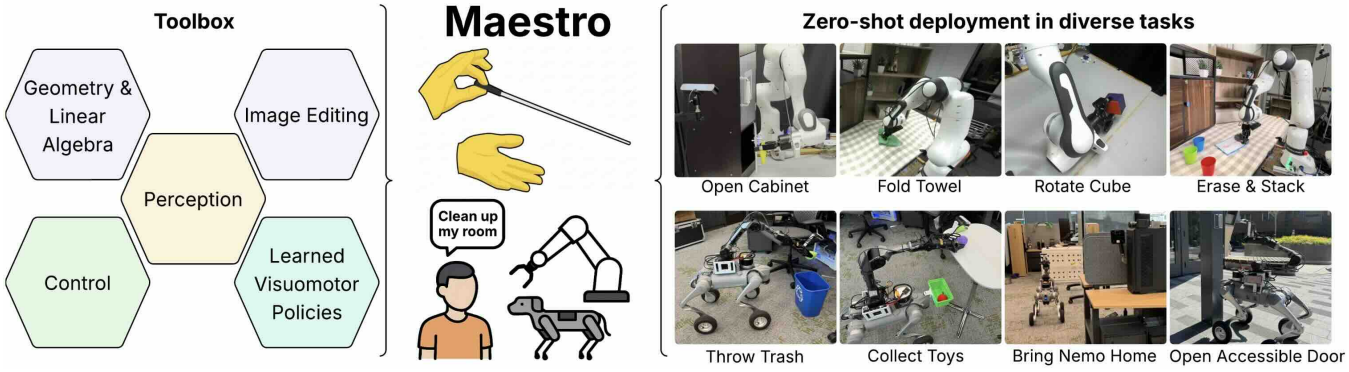

Fig. 1: **MAESTRO** receives language instruction and leverages a set of tools to complete diverse tasks in a zero-shot setting.

II. Related Work

We describe prior attempts at directly using VLMs and LLMs as robotic policies. See App. C 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, 4, 7–13]. 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 [14], 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 [15] 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 [15–17] is "open-loop".

More recent work [9, 18, 19] 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 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 general-purpose CaP-based manipulation 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 [20–22] or human videos [23–25]—the most performant methods still rely on massive real-world teleoperation data [1–4, 11, 13], which are costly and labor-intensive to collect.

In our experiments, we adopt the state-of-the-art $\pi_{0.5}$ model [11] 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

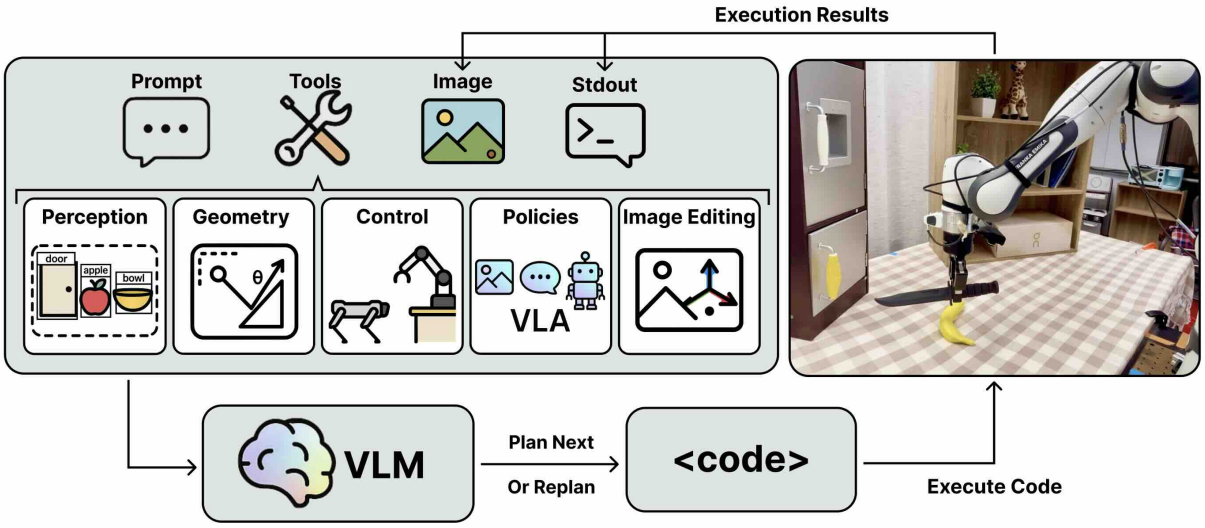

Fig. 2: Given prompt and images, VLM plans by writing and executing code that integrates perception, spatial reasoning, control, learned visuomotor policies, and image editing. Execution results (images and stdout) provide feedback for reacting and replanning, forming a closed-loop perception–action–learning 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 [30] 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 App. A 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 obervation 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, 7, 18]; Pointing [4]	Raw sensory inputs (RGB + proprioception); Segmentation centers; Pointing; **Active perception (zoom/look around with wrist camera)**; **FoundationStereo [26] depth**; **VLM-selected task-relevant keypoints (ReKep-inspired [27])**
Control	Cartesian control, gripper control [2, 4, 18]; Movement primitives [7]	Cartesian control, gripper control; **cuRobo collision-free motion planning**
Learned Visuomotor Policies	m2t2 grasp model [28]; Gemini Robotics 1.5 [4]	**GraspGen grasp model [29]**; **$\pi_{0.5}$ 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 [19]	Mobile base state estimation; **Active perception tools (look left/right/ground, view carry-on basket, log object location)**
Locomotion	Navigation [19]	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. In our

experiments, we use Gemini Robotics-ER 1.5 [4] as MAESTRO's VLM. 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 [30]: 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 7 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 [31]. 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 App. B for details.

B. Zero-Shot Tabletop Manipulation Results

Baselines. We benchmark MAESTRO against the strongest open-source 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 Robotics-ER 1.5 [4] and denote it as the **Gemini Robotics Agent**. For VLAs, we compare against the state-of-the-art models π_0 [1] and $\pi_{0.5}$ [11]. Specifically,

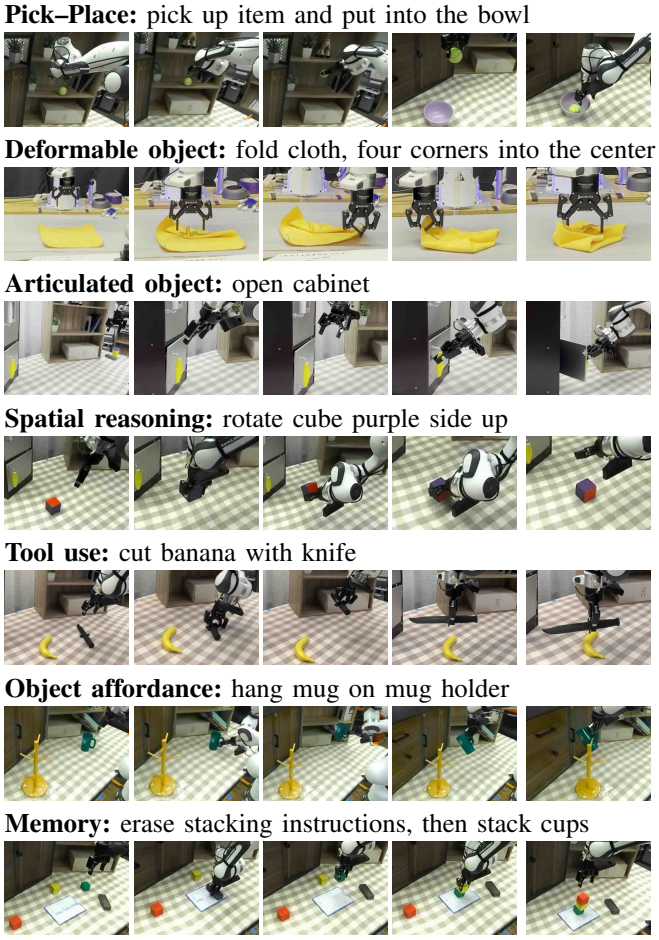

Fig. 4: **Tabletop manipulation evaluation tasks.**

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.

Tasks. We evaluate MAESTRO on seven 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**; *tool use*—**cut banana with knife**; *object affordance*—**hang mug on mug holder**; *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 six out of seven 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

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	$\mathbf{60.0 \pm 38.1}$
Tool use	Cut banana with knife	71.0 ± 28.8	30.0 ± 23.9	14.0 ± 6.5	$\mathbf{92.0 \pm 5.7}$
Object affordance	Hang mug on mug holder	46.0 ± 23.2	59.0 ± 30.7	$\mathbf{80.0 \pm 14.1}$	69.0 ± 9.6
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	$\mathbf{63.0 \pm 16.8}$

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 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. Nevertheless, the "hang mug on mug holder" task reveals MAESTRO 's current limitations: when complex chain-of-thought reasoning about object affordances and spatial orientation is required, it can still struggle. In this case, it fails to correctly align the mug handle's hole with the branch orientation of the holder.

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.

Long-horizon manipulation: collect all toys on table

Loco-manipulation: throw green ball into garbage can

Active exploration: search for object and return

Object affordance: press button to open door

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

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

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**.

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

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 geometry 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] G. R. Team, A. Abdolmaleki, *et al.*, *Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer*, 2025. arXiv: 2510.03342 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2510.03342.

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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/.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

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

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

[24] 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.

[25] 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.

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

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

[28] 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.

[29] 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.

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

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

[32] B. Sundaralingam, S. K. S. Hari, *et al.*, *Curobo: Parallelized collision-free minimum-jerk robot motion generation*, 2023. arXiv: 2310.17274 [cs.RO].

[33] J. Yang, H. Zhang, *et al.*, "Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v," *arXiv preprint arXiv:2310.11441*, 2023.

[34] K. Fang, F. Liu, *et al.*, "Moka: Open-world robotic manipulation through mark-based visual prompting," *Robotics: Science and Systems (RSS)*, 2024.

[35] C. Bai, T. Xiao, *et al.*, "Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4861–4868, 2022.

[36] S. Macenski, F. Martín, *et al.*, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 2718–2725.

[37] Y. J. Ma, W. Liang, *et al.*, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[38] Y. J. Ma, W. Liang, *et al.*, "Dreureka: Language model guided sim-to-real transfer," in *Robotics: Science and Systems (RSS)*, 2024.

[39] W. Liang, S. Wang, *et al.*, "Environment curriculum generation via large language models," in *Conference on Robot Learning (CoRL)*, 2024.

[40] L. Le, J. Xie, *et al.*, "Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model," *arXiv preprint arXiv:2410.13882*, 2024.

[41] G. J. Gao, T. Li, *et al.*, "Vlmgineer: Vision language models as robotic toolsmiths," *arXiv preprint arXiv:2507.12644*, 2025.

[42] W. Huang, C. Wang, *et al.*, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[43] H. Huang, F. Lin, *et al.*, *Copa: General robotic manipulation through spatial constraints of parts with foundation models*, 2024. arXiv: 2403.08248 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2403.08248.

[44] S. Patel, X. Yin, *et al.*, *A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards*, 2025. arXiv: 2502.08643 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2502.08643.

[45] M. Ahn, A. Brohan, *et al.*, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[46] W. Huang, F. Xia, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *arXiv preprint arXiv:2207.05608*, 2022.

[47] Y. Feng, J. Han, *et al.*, *Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation*, 2025. arXiv: 2502.16707 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2502.16707.

[48] S. Nasiriany, F. Xia, *et al.*, "Pivot: Iterative visual prompting elicits actionable knowledge for vlms," 2024. arXiv: 2402.07872 [cs.RO].

[49] W. Yuan, J. Duan, *et al.*, *Robopoint: A vision-language model for spatial affordance prediction for robotics*, 2024. arXiv: 2406.10721 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2406.10721.

[50] N. Di Palo and E. Johns, "Keypoint action tokens enable in-context imitation learning in robotics," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[51] Y. Yin, Z. Wang, *et al.*, "In-context learning enables robot action prediction in llms," in *ICRA*, 2025.

[52] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *arXiv preprint arXiv:2306.15724*, 2023.

[53] J. Duan, W. Pumacay, *et al.*, *Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation*, 2024. arXiv: 2410.00371 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2410.00371.

[54] Y. J. Ma, J. Hejna, *et al.*, *Vision language models are in-context value learners*, 2024.

APPENDIX

A. Method Full Technical Details

1) Tabletop Manipulation Modules: We equip our tabletop manipulation agent with a suite of tools spanning perception, reasoning, control, pre-trained visuomotor policies, and image editing. The tools are provided in **bolded text** below.

Perception. For perception tools, we adopt a "coarse-to-fine" approach. At the fastest and simplest level, the agent can access **raw sensory inputs** (RGB images and robot proprioception). Because raw depth is often noisy, we use FoundationStereo [26] to estimate more reliable **depth maps** from RGB. For **object center point**, given a language description, we employ Grounded-SAM to produce a mask and its center point, while a Gemini-based pointing tool returns single 2D points corresponding to language queries. For tasks requiring higher precision—such as grasping towel corners—we provide a **salient task-relevant points** tool inspired by ReKep [27]: after generating a segmentation mask, we overlay a uniform point grid and ask GPT-o3 to select the most relevant points. This tool hierarchy balances granularity and runtime: raw images are coarse but fast, while salient points are slower but more precise. MAESTRO autonomously reasons about which tools to use and how to use them for each task, achieving a balance between execution speed and task performance. As an alternative, we also provide **language-based pointing** using Gemini Robotics-ER 1.5 [4].

Reasoning. In our initial experiments, we observed that even with advanced perception tools, MAESTRO struggled to perform task-relevant spatial reasoning. We identified the key limitation as the lack of spatial chain-of-thought reasoning in current VLMs. To address this, we equipped MAESTRO with a small suite of simple but targeted tools designed to spark spatial reasoning and break complex tasks into intermediate steps. These include **measuring distances**, **constructing vectors using points**, **computing relative rotations between**

vectors, and **rotating a vector by specified angles**. Together, these tools dramatically improve MAESTRO's ability to reason about spatial relationships and act accordingly.

Control. For low-level manipulation, we provide a set of simple Cartesian end-effector control tools: **move gripper to**, **open gripper**, and **close gripper**. To ensure safe and reliable motion, we incorporate cuRobo [32] for point-cloud-based, collision-free motion planning, which greatly mitigates the risk of unintended contact with the environment and drastically improve object interaction performance.

Learned Visuomotor Policies. We equip MAESTRO with two types of learned visuomotor policies: a **grasp model** and a **VLA**. For grasping, we provide GraspGen [29], while for general-purpose low-level visuomotor control we provide the state-of-the-art $\pi_{0.5}$ model [11]. A key challenge when integrating end-to-end VLAs as tools is determining when to interrupt their execution, since they continue running inference until externally stopped. Because VLAs operate at high inference speeds, our framework requires an equally fast closed-loop monitor. To achieve this, we host Qwen-2.5-VL-72B-Instruct locally to check task completion at 2 Hz using a simple yes/no question, allowing rapid feedback and timely intervention.

Image Editing. Prior work [33, 34] shows that adding visual marks can improve the grounding and reasoning abilities of VLMs. Thus, we provide MAESTRO with image-editing tools that can **draw points** and **overlay 6D poses** on images, enabling clearer spatial references and richer visual reasoning during manipulation.

2) Additional Modules for Mobile Manipulation: To extend beyond tabletop tasks, we equip MAESTRO with additional tools that enable mobile manipulation.

Perception. For a mobile manipulator, obtaining the full **robot 6D state** requires more than raw proprioception. We therefore employ Faster-LIO [35], a lightweight LiDAR–Inertial Odometry method, to provide robust state estimation of the mobile base even under visually challenging conditions. This enhanced perception ensures reliable navigation and spatial reasoning across larger workspaces. In addition, mobile manipulation benefits from active perception to build a more complete understanding of the surroundings and gather task-relevant information on demand. To this end, we provide MAESTRO with active perception tools—**look left**, **look right**, **look to the ground**, and **view carry-on basket**, **remember object location**—which allow it to scan the environment, adjust its viewpoint, and access the basket contents as needed.

Locomotion. We supply two tools: (1) the **nudge** tool, which applies small velocity adjustments for precise positioning near a target location, and (2) the **navigation** tool, which leverages Nav2[36] to move the robot safely to a target pose on the map. This dual interface allows MAESTRO to fluidly switch between global navigation and fine-grained local control.

Manipulation. To support transport tasks, we provide a dedicated **put in basket** tool that enables the robot to efficiently carry multiple objects during long-horizon mobile manipulation. For unloading, MAESTRO leverages the **view carry-on basket** perception tool to inspect the basket's contents and then automatically generates the code needed to move objects out of the basket.

B. Task Evaluation Rubric

We performed rigorous evaluation on all tasks following STAR-Gen [31] to generate new trial for zero-shot generalization capabilities. 5 trials each task (1 initial setup following 4 generated by STAR-Gen), and for each STAR-Gen generated trial, vary all of the following:

- Object placement
- Object instance
- Scene / Lab setting
- Language instruction (paraphrase)

We designed a task completion tracking rubric to quantitatively evaluate the system performance, presenting results as a percentage of completion, where the maximum score is 100%. The overall task is decomposed into a sequential series of verifiable sub-steps. This metric moves beyond a simple binary success/failure and provides diagnostic detail on where the agent fails, which is critical for evaluating complex, long-horizon manipulation tasks.

The evaluation metrics along with STAR-Gen variations for table-top manipulation tasks are detailed below:

1) Pick-place (*"Put item in bowl."*)
   [25%] Approach the item.
   [50%] Grasp the item.
   [60%] Lift up the item.
   [75%] Approach the target location.
   [100%] Place correctly.
2) Deformable object (*"Fold the four corners of the towel into the center."*)
   For this task, STAR-Gen introduces significant geometric and interactive variability across trials (e.g., varying object shape and location), we could not rely on one fixed metric. Instead, the task completion rubric was dynamically fine-tuned for each generated trial, ensuring a fair evaluation of progress for sub-goals of each scenario.
   a) *"Fold the four corners of the towel into the center."*
      (color of the towel is different between two trials)
      [25%] One corner folded in.
      [50%] Two corners folded in.
      [75%] Three corners folded in.
      [100%] All four corner folded in successfully.
   b) *"Fold the T-shirt into a rectangle."*
      [15%] One t-shirt sleeve folded in.
      [30%] Both t-shirt sleeves folded in.
      [60%] Bottom of t-shirt folded in.
      [100%] Fold entire t-shirt inward successfully.
   c) *"Place one corner of the towel to its diagonal corner."*
      [30%] Approach one corner.
      [60%] Grasp and lift up the corner.
      [100%] Place at the diagonal corner successfully.
   d) *"Unfold the towel into a square."*
      [30%] Accurately approach the folded corner.

[60%] Grasp and lift up the corner.
[100%] Place at the table to finish unfolding.

3) Articulated object (*"Open cabinet."*)
[10%] Approach the cabinet.
[40%] Grasp the handle.
[60%] Attempt to pull.
[70%] Open the door slightly.
[100%] Open the door completely.
4) Spatial reasoning (*"Rotate cube purple side up."*)
[10%] Approach the cube.
[30%] Grasp the cube.
[60%] Rotate the cube.
[100%] Purple side faces up.
5) Tool use (*"Cut banan with knife."*)
[20%] Approach the knife.
[50%] Grasp and lift up the knife.
[60%] Approach the banana.
[80%] Position in a good cutting orientation.
[100%] Cut the banana.
6) Object affordance (*"Hang mug on mug holder."*)
[25%] Approach the mug.
[50%] Grasp and lift up the mug.
[75] Approach the mug holder.
[100%] Hang on to the stand bar.
7) Memory & long-horizon & semantic (*"Erase instructions on whiteboard, then follow instruction to stack cups."*)
[20%] Pick up the erasor.
[30%] Erase the white board partially.
[50%] Erase the white board completely.
[60%] Return the eraser.
[80%] Stack the second cube on top of the first cube.
[100%] Stack the third cube on top of the second cube.

The evaluation metrics for mobile manipulation tasks are detailed below:

1) Long-horizon manipulation (*"Collect all toys on table."*)
[25%] Collect one toy.
[50%] Collect two toys.
[75%] Collect three toys.
[100%] Collect all four toys.
2) Long-horizon loco-manipulation (*"Throw the ball into garbage can."*)
[16.7%] Find the ball.
[33.3%] Move to the ball.
[50%] Pick up the ball.
[66.7%] Find the trash can.
[83.4%] Move to the trash can.
[100%] Drop the ball into the trash can.
3) Active exploration (*"Search item and return when grasped."*)
[20%] Explore around the area.
[40%] See the object.
[60%] Move to the object.
[80%] Grasp the object.
[100%] Return to the initial position.
4) Object affordance (*"Press button to open door."*)
[33.3%] Identify the correct label.
[66.6%] Approach to the correct button.
[100%] Press button to open the door.

C. Extended discussion on prior work

A wide range of prior work has explored the use of large models — Vision-Language Models (VLMs) and Large Language Models (LLMs) — to support specific components of robotic systems. In the common paradigm, these models are embedded into manually designed, typically rigid, modular pipelines, where large models take on one or several well-defined roles while the remaining pipeline is built through "good old-fashioned engineering" by humans. These roles span reward and environment design [37–41], constraint-function design [27, 42–44], high-level planning [17, 45], self-reflection [46], visual question answering [34, 47–49], in-context learning [50, 51], and task-progress evaluation [52–54].

While effective within their intended scope, these approaches remain constrained by their rigid workflows. Because large models automate only a small portion of the pipeline, substantial manual effort is still required to design the rest. This rigidity also limits scalability: systems tuned and fixed for specific settings struggle to generalize to diverse, in-the-wild scenarios. As a result, these methods fall short of the requirements for scalable, general-purpose robotic manipulation.

We show that achieving this goal requires the opposite of prior practice: rather than increasing the complexity of the agentic system, especially its hand-engineered components, we reduce it. By stripping away rigid, manually defined workflows and instead scaling up the breadth and quality of the tools available to the agent, we create a more fluid and autonomous framework: one capable of dynamically deciding *how*, *when*, and *which* modules to employ.