BEYOND ARRIVAL: GEOMETRIC ALIGNMENT FOR AGENTIC HUMANOID LOCO-MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Loco-manipulation for humanoid robots in open environments requires both long-range navigation and whole-body loco-manipulation abilities. While recent works have demonstrated powerful navigation and manipulation capabilities, bridging the two within a unified framework remains an open problem. The primary difficulty is that the navigation goal and the initial state of manipulation are misaligned in terms of observations. Existing approaches typically rely on task-specific training or additional annotations, with limited generalization across tasks and scenes. We introduce a handoff phase to reduce observation misalignment and use object-relative geometry as a bridge between navigation and manipulation. Building upon this insight, we propose **GATE**, a **G**eometry **A**ligned **T**ask **E**xecution bridge that leverages geometry aware representations to bridge navigation and manipulation, requiring no extra training or markers. An agent selects position and orientation references from the manipulation objective; a geometry aware feedback controller adjusts the robot's position before handoff. Our approach effectively improves the overall success rate of loco-manipulation tasks and demonstrates clear advantages over alternative methods in generalization, stability, and robustness. Building upon this approach, we develop an agentic system and deploy it in the real world. It completes multiple complex whole-body manipulation tasks with substantially higher success rates than the baselines. We further evaluate GATE on the OVMM benchmark, providing evidence that the geometric handoff principle is applicable to a different embodiment and an independently trained manipulation policy.

1 INTRODUCTION

Recent advances in humanoid whole-body control (Luo et al., 2025; Li et al., 2025), vision-language-action policies (Jiang et al., 2025; Wei et al., 2026; Hu et al., 2026), and agentic navigation (Ding et al., 2025; 2026) lay the foundation for long-horizon loco-manipulation. However, successful navigation does not necessarily provide a suitable starting state for manipulation.

The central challenge lies in the handoff failures between navigation and manipulation. A navigation controller may stop near the target object while leaving the robot too far away, laterally displaced, poorly oriented, or unable to observe all relevant objects. An existing manipulation policy can therefore receive an unfavorable initial observation even when navigation has reached its own goal. The research question is whether task-conditioned geometric alignment can reduce the mismatch between the initial observation distribution induced by navigation and that expected by existing manipulation policies, thereby seamlessly bridging navigation and manipulation skills.

We use object-relative geometry to represent spatial conditions relevant to the manipulation handoff, including distance, lateral offset, and orientation. Through experiments, we find a separate reference geometry and design a general-purpose base-pose controller accordingly. The backend MLLM agent selects the target object based on the image and instruction, after which the controller performs feedback control by estimating the geometric relationship between the target object and the robot.

Building on GATE, we develop a practical agentic system that coordinates navigation, geometric alignment, and whole-body manipulation. Drawing inspiration from LaViRA, we adopt a dual-agent architecture. The high-level language agent analyzes the instruction, history, and current observation to determine an overall strategy, including which direction to take, which skill to use, and how to

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

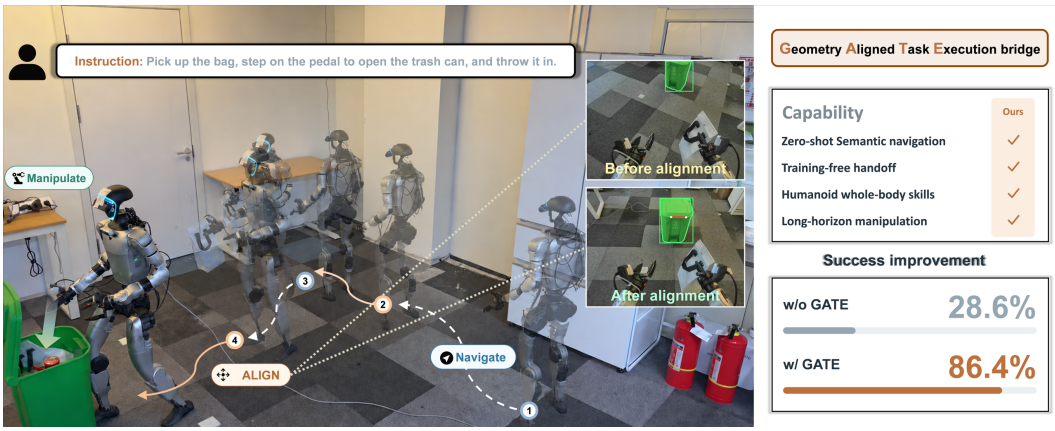


Figure 1: Long-horizon humanoid loco-manipulation with GATE. Our dual-agent framework coordinates semantic navigation and whole-body manipulation, while GATE provides a training-free geometric handoff by adjusting the robot's position and heading relative to task-relevant objects. The insets illustrate the resulting change in the manipulation view. Pooling full-system trials on T1–T4 and near-field trials on T1–T2, the observed task success rate is 28.6% without GATE and 86.4% with GATE.

manage the TODO list. A high-frequency vision agent handles low-level tasks, including grounding and state monitoring. Navigation, alignment, and manipulation are encapsulated as independent skills and converted into motor commands through a unified low-level motion control algorithm.

This hierarchical decomposition offers three key abilities. First, it possesses zero-shot navigation capabilities, enabling it to execute tasks across diverse locations in open environments based on different language instructions. Second, whole-body manipulation skills enable it to perform complex whole-body tasks. Third, our action alignment mechanism and agent design effectively bridge navigation and manipulation skills, demonstrating an exceptional overall success rate and generalization performance.

Our work is organized around three contributions:

- **GATE: geometric alignment for handoff.** We address navigation-to-manipulation handoff failure with object-aware geometry relations, closed-loop base adjustment, and readiness checks, without additional training or special annotations.
- **Agentic humanoid loco-manipulation.** We develop an agentic loco-manipulation system that coordinates navigation, alignment, and humanoid manipulation. The system exhibits robust generalization across diverse workspaces and tasks in real-world settings.
- **Long-horizon execution and transfer evaluation.** We evaluate the agentic system on real-world humanoid tasks, improve the overall success rate from from 35.0% to 82.5%, and we examine the applicability of GATE on the OVMM benchmark. These experiments assess the geometric handoff principle across different embodiments and downstream manipulation policies.

2 Related Work

Humanoid Loco-manipulation Skills. Humanoid loco-manipulation involves coordinated control of a humanoid robot to move and interact with the environment. SONIC provides a pretrained motion-token space for whole-body teleoperation and VLA control (Luo et al., 2025). Humanoid VLAs use varied action representations, from upper-body joint targets with locomotion commands (Jiang et al., 2025; Wei et al., 2026) to full-body joint targets with root motion (Hu et al., 2026). Training data and action representations distinguish these approaches. Our manipulation skills fine-tune a pretrained VLA using exclusively robot teleoperation demonstrations, with SONIC motion tokens and gripper commands as action labels. This reuses whole-body motor priors while grounding task learning in the target robot's capabilities.

Navigation skills. Vision-language navigation allows robots to perform locomotion, exploration, and object-search tasks in open environments following semantic instructions. NAVILA fine-tunes foundation models using simulated and video data, enabling the end-to-end generation of motion commands based on visual observations and instructions (Cheng et al., 2025). LaViRA and Uni-LaViRA utilize a hierarchical agentic architecture with two models of different sizes for planning and grounding and output goal waypoints to a low-level path planner in a completely training-free manner (Ding et al., 2025; 2026). Overall, the agentic paradigm currently demonstrates superior performance and scalability; motivated by this, we also adopt this approach.

Mobile Manipulation Systems. Mobile manipulation systems integrate planning, memory, and skill execution for long-horizon tasks. OWMM-Agent maintains multi-view memory and states to generate multimodal actions through function calling (Chen et al., 2025). ODYSSEY combines hierarchical vision-language planning with whole-body control for long-horizon exploration and manipulation (Wang et al., 2026). MoMaStage constrains skill composition through a skill-state graph and triggers replanning when execution deviates from the plan (Li et al., 2026). Following these principles of skill coordination and execution feedback, our dual-agent system integrates navigation and geometric alignment with whole-body humanoid loco-manipulation, while explicitly tracking progress through long-horizon tasks.

Navigation-to-Manipulation Handoff. Handoff failure is a key problem in loco-manipulation tasks. Prior work addresses this boundary through trainnning, annotation or zero-shot ways. MoMa-Kitchen provides large-scale, affordance-grounded supervision for final navigation positions and introduces NavAff, a trained policy to predict final positions (Zhang et al., 2025). HERMES uses an environment-specific PnP method to adjust positions (Yuan et al., 2025). MoManipVLA transfers pretrained fixed-base VLA policies to loco-manipulation by using their predicted end-effector waypoints to guide bi-level trajectory optimization (Wu et al., 2025b). MoTo selects task-relevant object and arm keypoints using vision-language models, aggregates target keypoints across views, and optimizes motion to obtain feasible docking configurations (Wu et al., 2025a). UniLM-Nav decomposes handoff positioning into reference-view selection, task-conditioned affordance grounding, and geometry aware base-pose reasoning with a shared multimodal language model (Zhang et al., 2026). Our geometric handoff module uses task-conditioned references and closed-loop geometric feedback, without requiring additional transition-policy training.

3 METHOD

3.1 SYSTEM OVERVIEW

We introduce a framework with dual agents and a unified skill interface to decompose planning, grounding, and action execution. Figure 2 illustrates the overall pipeline. A Language Agent (LA) invokes skills, makes plans, and manages memory, while a smaller Vision Agent (VA) handles simple visual tasks such as grounding and post-execution checks. A unified skill interface executes manipulation, navigation, and alignment skills. To execute these diverse tasks seamlessly on a single humanoid robot while avoiding falls, we use a unified humanoid locomotion layer to control the robot's motors.

Language Agent. LA receives a language instruction $\mathcal{I}$, a global observation O_t^g consisting of five images $\{I_{front}, I_{frontleft}, I_{left}, I_{frontright}, I_{right}\}$ and mission history $\mathcal{H}_t$ including the last action result O_{A_t} and at most five historical result images $\{O_t^r ... O_{t-5}^r\}$. Given the context, LA performs two tasks: selecting a language skill A_t^{lang} and updating a task list $\mathcal{L}_t$. The whole process can be formulated as:

$$(A_t^{lang}, \mathcal{L}_t) = LA(\mathcal{I}, O_t^g, \mathcal{H}_t). \tag{1}$$

The language action A_t^{lang} is a structured JSON response containing a state estimate, reasoning, the selected skill, predicted results, and arguments. The task list $\mathcal{L}_t$ serves as a simple text memory. Details and examples can be seen in Appendix C.2 and Appendix E.3.

Vision Agent. VA has two main tasks. First, it is responsible for extending a language action A_t^{lang} into a vision action A_t^{vis} including visual groundings g_t. This can be formulated as:

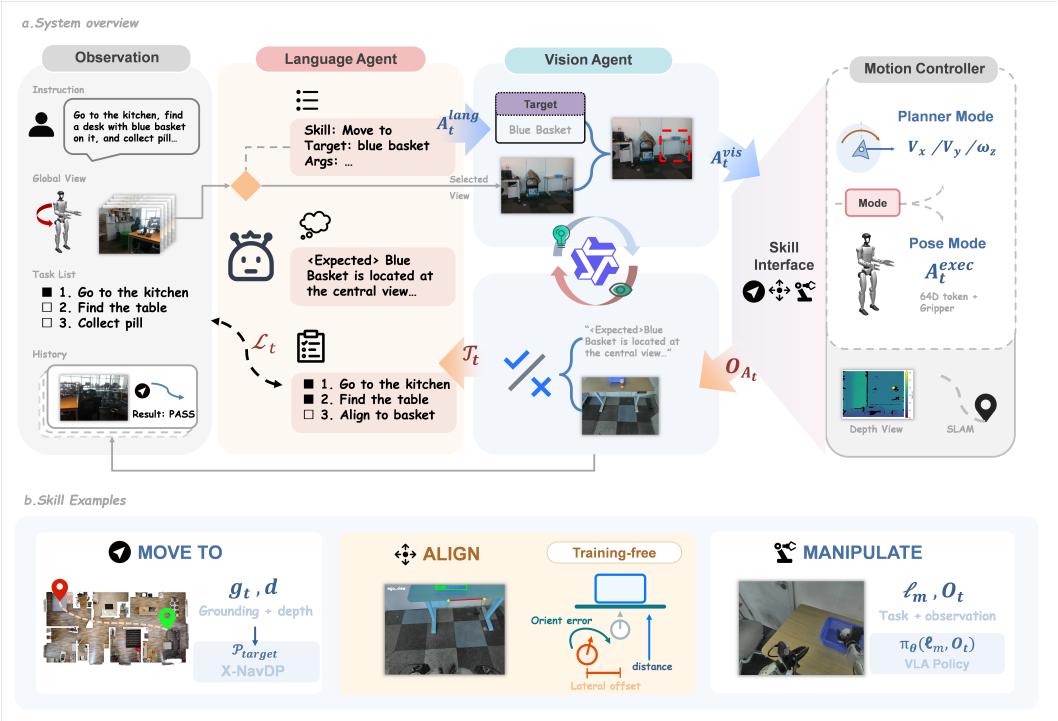


Figure 2: GATE framework: the Language Agent (LA) plans skills, and the Vision Agent (VA) grounds targets and checks execution outcomes. A unified skill interface coordinates navigation, geometric alignment, and manipulation through the SONIC whole-body controller. GATE provides training-free geometric alignment at the navigation-to-manipulation handoff.

$$A_t^{vis} = VA(A_t^{lang}, O_t^p). \tag{2}$$

The second task is to perform a post-execution check after a skill finishes. In this case, VA receives the action-result observation O_{A_t} and the selected language action A_t^{lang}. By comparing the predicted result with the observed outcome, it produces a task transition $\mathcal{T}_t \in \{PASS, REPLAN, FAIL\}$. This can be formulated as:

$$\mathcal{T}_t = VA(A_t^{lang}, O_t^p). \tag{3}$$

More details can be seen in Appendix C.3.

Skill interfaces. The skill interface is designed as a deterministic orchestration layer between the agents and robot controllers. It provides three categories of skills for LA and combines $(A_t^{vis}, \mathcal{T}_t)$ to generate skill goals or change the mission stage. The skill categories are as follows:

- `MOVE_TO`: LA selects a semantic waypoint and a direction. After the robot faces the selected direction, VA grounds the waypoint, and the skill layer constructs a local navigation goal.
- `ALIGN`: VA selects a geometric reference, as described in Section 3.3, based on the language action A_t^{lang} and the current observation; GATE then aligns the robot with that reference.
- `MANIPULATE`: Directly invokes one of the configured policies.

The skill interface implements a filter layer based on a state machine and harness, and integrates multimodal observation inputs to provide initial instructions for specific policies, forming an abstraction

layer that allows skill algorithms to be replaced without additional modifications. A detailed description of the skill state machine, harness, and transitions is provided in Appendix C.4.

Whole-body control layer. We implement SONIC as a unified whole-body control layer with two mutually exclusive modes. In *planner mode*, SONIC converts high-level speed commands into whole-body motion references. This is mainly used in navigation and alignment stages. In *pose mode*, whole-body motion references are directly provided by skill policies. We use SONIC's token space as the motion-reference space. Both modes share the same robot-state feedback and low-level whole-body controller. Further details of SONIC deployment are provided in Appendix C.5.

3.2 Navigation and Manipulation Skills

3.2.1 Navigation Skills

Navigation skills translate the identified visual target into speed commands. We use a multimodal method as the navigation skill implementation.

View Selection & Grounding. When selecting a navigation skill, LA outputs the selected view as input to VA O_t^p in $\{I_{front}, I_{frontleft}, I_{left}, I_{frontright}, I_{right}\}$. Then VA performs grounding and the robot turns toward the corresponding direction.

Object-to-Point Projection. Given a grounding bounding box g_t, we select a 7×7 pixel region at its center and perform spatial and temporal pooling to estimate depth d. We then unproject the center pixel into the camera's 3D coordinate system using the intrinsic matrix K:

$$[X_{target}, Y_{target}, Z_{target}]^\top = d \cdot K^{-1} \cdot [u_{target}, v_{target}, 1]^\top \tag{4}$$

Path Planning. Given the target point $\mathcal{P}_{target}$, the robot position $\mathcal{P}_{robot}$, and the RGB-D input O_{nav}, the navigation skill uses X-NavDP (Yang et al., 2026) to generate a robot trajectory. An MPC controller then converts the trajectory into speed commands with local obstacle avoidance. To obtain the positions of $\mathcal{P}_{target}$ and $\mathcal{P}_{robot}$ in world coordinates, we use a SLAM algorithm based on the onboard LiDAR (Xu et al., 2021; Sun et al., 2025). Further details are provided in Appendix C.6.

3.2.2 Manipulation Skills

For each manipulation skill, we independently fine-tune a pretrained $\pi_{0.5}$ policy (Physical Intelligence et al., 2025) to predict SONIC motion tokens and gripper commands. The resulting task-specific policies share a common observation and action interface and execute whole-body behaviors through the SONIC control layer.

Whole-body action prediction. Given a manipulation instruction ℓ_m, the policy receives an observation O_t comprising four RGB images from the head, chest, and two wrist cameras, together with the whole-body joint configuration $\mathbf{q}_t$ and the body-frame projected gravity vector $\mathbf{g}_t$. It predicts a sequence of H actions:

$$\mathbf{A}_t = \pi_\theta(\ell_m, O_t) = [\mathbf{a}_{t,0}, \ldots, \mathbf{a}_{t,H-1}], \qquad \mathbf{a}_{t,i} = \left[\mathbf{z}_{t,i}; \mathbf{u}_{t,i}^{\text{grip}}\right], \tag{5}$$

where $\mathbf{z}_{t,i}$ is a SONIC whole-body motion token and $\mathbf{u}_{t,i}^{\text{grip}}$ contains the left and right gripper commands. Demonstrations are collected through whole-body teleoperation, directly recording the executed SONIC tokens and gripper commands as action labels. We adapt the policy's action input and output projections to the concatenated action representation and fine-tune the model using the action flow-matching objective.

Online execution. Policy inference runs asynchronously while the current action chunk is streamed to the SONIC control layer at 50 Hz. When a new chunk arrives, the executor skips its elapsed prefix according to the measured delay from policy-call initiation to result consumption, then replaces the cached sequence. Details of the manipulation skills are provided in Appendix C.7.

3.3 GATE: Geometry Aligned Task Execution bridge

GATE separates task-level semantic reasoning from geometric feedback control. Selecting an appropriate alignment reference requires interpreting both the manipulation objective and the current

scene, whereas approaching that reference requires frequent geometric observations and timely motion corrections. These two processes require distinct capabilities. We therefore use the MLLM-based VA to select task-conditioned position and orientation references, and use real-time visual feedback and a staged low-level controller to perform closed-loop alignment. This division allows task semantics to determine what to align with, while geometric feedback determines how the robot should move.

Task-conditioned reference objects. Our key observation is that longitudinal, lateral, and angular deviations jointly determine whether the robot is suitably positioned and oriented for manipulation. Aligning solely with the manipulation object does not always sufficiently constrain all three. We therefore detect structural edges on an object in the manipulation workspace, which may or may not be the manipulated object itself, and use these edges as orientation references to reduce angular error. GATE expresses alignment through two reference roles selected by the VA from the current image and the manipulation instruction ℓ_m: a position reference r_p and an orientation reference r_ψ. The position reference is an object involved in the manipulation, such as an object to be grasped or a destination container. The orientation reference provides a visible structural edge and may be the same object or a supporting structure.

Visual geometry and adjustment. Text-prompted detection and segmentation based on YOLOE (Wang et al., 2025) locate the selected reference objects. On G1, position feedback is obtained from calibrated RGB-D observations of the selected head or chest camera. Position observations can switch cameras during recovery. We summarize the controller's adjustment objectives as

$$\mathbf{e}_t = \left[x_t - d^*_{c_t},\ r_t,\ e_{\mathrm{ori},t}\right]^\top, \tag{6}$$

where x_t and r_t are forward distance and rightward offset, c_t is the selected position camera, and $d^*_{c_t}$ is its configured standoff. Let $\mathbf{n}_t = (n_{x,t}, n_{y,t})^\top$ denote the selected edge's unit normal in the body horizontal plane, oriented toward the side containing the edge. The raw yaw correction is $e_{\psi,t} = \mathrm{atan2}(n_{y,t}, n_{x,t})$. The controller uses its filtered or heading-propagated counterpart $e_{\mathrm{ori},t}$. Appendix B details depth back-projection, camera-to-body calibration, planar edge-normal estimation, camera switching, and the staged state machine.

4 Experiments

We study three questions: whether GATE improves navigation-to-manipulation handoff, whether the integrated system supports completion of the four real-world tasks, and whether the geometric handoff design remains effective when adapted to Stretch on OVMM. Additional near-field experiments examine performance under varied initial configurations and scene backgrounds on T1 and T2. Appendices D.3 and D.5 provide the handoff analyses and near-field evaluation details.

4.1 Real-World Humanoid Experiment

Tasks. We design four tasks with varying difficulty and complexity, with detailed task definitions in Appendix D.1. T1 is a simple pick-and-place task. T2 emphasize whole-body manipulation. T3 requires to use body as a part. T4 evaluates repeated alignment across successive navigation and manipulation stages. Together, these tasks assess the role of a shared alignment module in supporting whole-body coordination and extended task execution.

Platform & Protocol. We evaluate our system on a Unitree G1 humanoid. In the main evaluation, each task–method configuration is evaluated over ten trials, varying both the manipulation environment and the robot's initial configuration. We report the success rate and the mean task progress for each task. Progress is determined by predefined milestones covering navigation arrival and manipulation completion. Hardware details and evaluation procedures are provided in Appendix A and Appendix D.

Baselines & Ablations. We compare three handoff variants: **Direct VLA**, which omits alignment; **GATE w/o Edge Alignment**, which disables edge detection and edge-based alignment while keeping all other system components and execution settings identical to Full VLN; and **VLM basepose**, which directly predicts the target base position. **LaViRA** and **NAVILA** provide alternative navigation baselines. Detailed configurations are reported in Appendix D.1.

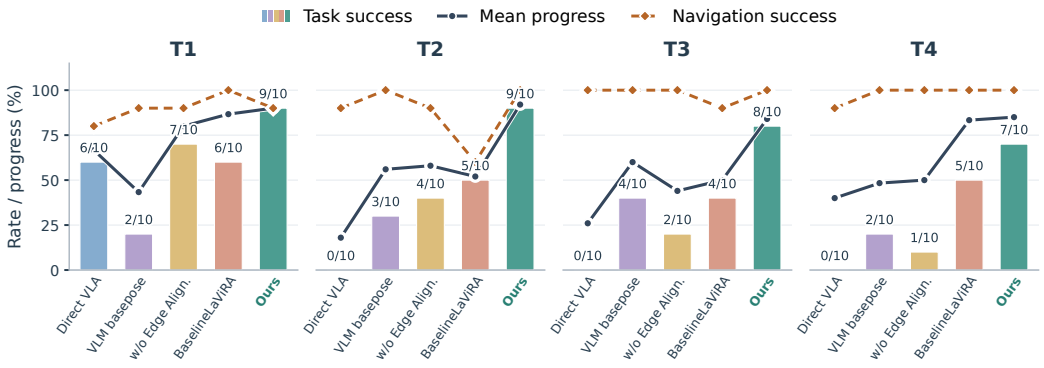


Figure 3: Real-world task success (bars), mean progress (solid line), and navigation success (dashed line), over ten trials per method and task. T4 uses first-leg navigation if the first VLA operation fails, and second-leg navigation otherwise. "w/o Edge Align." denotes GATE w/o Edge Alignment. Bar labels show successful trials / total trials; all plotted metrics are percentages. NAVILA did not successfully complete any of the four tasks (T1–T4).

Main Results. Figure 3 compares task success, mean progress, and navigation success across tasks; full values are reported in Table 15 in Appendix D. Our method achieves substantially higher success rates and task progress across all four tasks, with an average success rate of 82.5% (33 successful trials out of 40). These results demonstrate that the complete agentic system can perform complex loco-manipulation tasks in open environments. On a simple pick-and-place task such as T1, Direct VLA achieves a 60% success rate. However, on T2 and T3, which require whole-body manipulation, and on the more complex long-horizon task T4, Direct VLA does not succeed in any of the evaluated trials without an alignment module. VLM basepose improves on Direct VLA in T2–T4 but lowers T1 success from 60% to 20%, indicating that direct VLM pose prediction can also impair downstream execution in this G1 setting. GATE w/o Edge Alignment improves both success rate and task progress over Direct VLA across all tasks, but its success rates remain below the 70–90% achieved by GATE. This comparison evaluates the contribution of edge detection and alignment with the detected edge within the otherwise unchanged system.

T4 further tests alignment in a complex task that requires coordinated navigation and whole-body manipulation across repeated skill transitions. The full system and the LaViRA baseline, which uses the same alignment method, achieve task success rates of 70% and 50%, respectively, compared with 0%, 10%, and 20% for Direct VLA, GATE w/o Edge Alignment, and VLM basepose. In particular, the gap between Full VLN and GATE w/o Edge Alignment supports the benefit of edge detection and alignment with the detected edge across repeated navigation-to-manipulation handoffs.

NAVILA did not successfully complete any of the four tasks (T1–T4). Finally, comparisons with the LaViRA and NAVILA baselines further validate the advantages and stability of the complete system in long-horizon tasks, demonstrating the effectiveness and robustness of the overall design.

Analysis. We first examine whether GATE brings pre-manipulation observations closer to the policy's training examples. Using each task's frozen policy vision encoder, we measure the distance between spatially pooled image representations and the training-reference set. As shown in Figure 4(a–b), this distance decreases after GATE in 60/62 eligible T1 pairs and 59/60 T2 pairs from the full-system and near-field experiments, with median per-pair relative changes of -36.1% and -35.2%, respectively. Both trial sources exhibit the same downward trend. The cross-method comparison further shows that the full system achieves the lowest median pre-manipulation visual distance on both tasks (Figure 4(c–d)). Because GATE controls motion through geometric feedback without directly optimizing this representation distance, these results indicate that its geometric adjustments also bring the policy's visual inputs closer to training examples.

We next use geometric analysis to provide a spatial explanation. Figure 5 shows that handoff poses after GATE are more concentrated around the object-relative positions and headings represented in the training demonstrations. Under a training-neighborhood criterion that jointly considers lateral position, normal distance, and heading, the full system records 9/10 hits on T1 and 8/10 on T2,

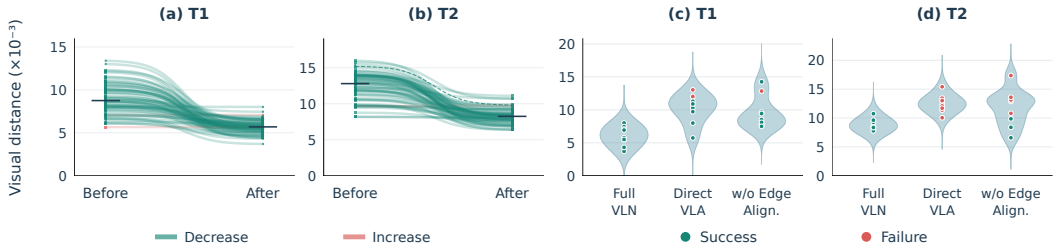


Figure 4: Visual distance to task-specific training references; lower values indicate greater visual similarity. (a–b) Paired observations before and after GATE, pooling full-system and near-field trials (T1: 62 pairs; T2: 60 pairs). Curves connect observations from the same trial, and dark bars indicate medians. (c–d) Pre-manipulation visual distances for three handoff methods in the full-system experiments, with points colored by physical task success or failure. "w/o Edge Align." denotes GATE w/o Edge Alignment.

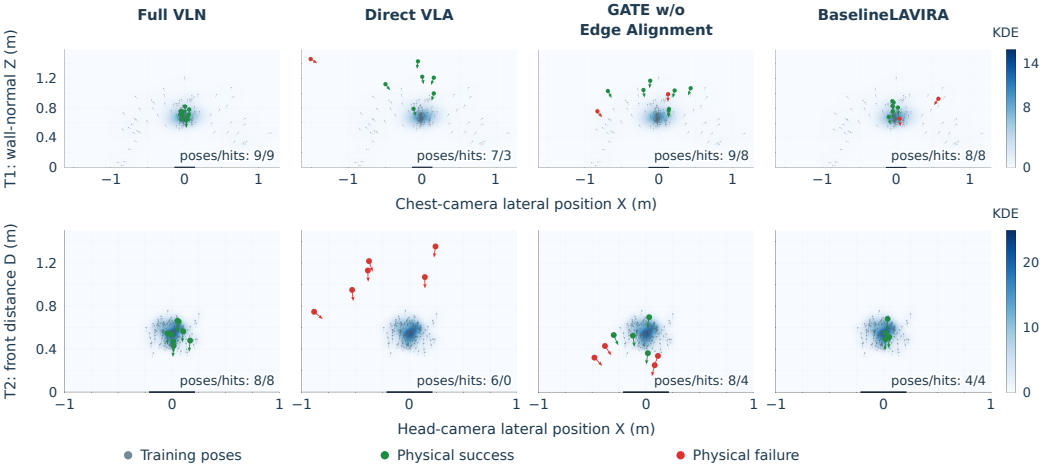


Figure 5: Geometry at the navigation-to-manipulation handoff for T1 (top) and T2 (bottom). Blue shading shows the training-pose density (m^{-2}), and gray markers show training poses. Evaluation positions and heading arrows are colored by physical task success (green) or failure (red). "Poses" counts trials with estimable geometry, and "hits" counts those trials whose poses fall within the training-pose neighborhood. Each method was evaluated over 10 trials per task.

compared with 3/10 and 0/10 for Direct VLA, and 8/10 and 4/10 for GATE w/o Edge Alignment. The larger gap on T2 suggests that adding edge detection and alignment with the detected edge helps bring handoff poses closer to the object-relative configurations represented in the training demonstrations.

Near-field experiments provide further support without repeating long-range navigation. Initial positions are selected from a region defined by the typical distribution of navigation endpoints. In these experiments, with the same task-specific manipulation policy, adding GATE raises T1 success from 48.0% to 83.3%, and T2 success from 18.6% to 92.5%. The initial-state maps in Appendix D.5 show that, particularly on T2, GATE + VLA succeeds across varied distances and lateral offsets, whereas Direct VLA successes are more concentrated at starting positions with smaller lateral offsets and shorter distances. Together with the full-system trials, these results support end-to-end robustness to variation in navigation endpoints. Together, the visual, geometric, and near-field results support GATE's role in improving the handoff conditions presented to an unchanged manipulation policy.

4.2 CROSS-EMBODIMENT TRANSFER ON OVMM

Setups. We evaluate GATE on OVMM to examine its applicability to a different robot embodiment and downstream manipulation system. In this benchmark, a Stretch robot must locate and grasp a target object, navigate to a specified receptacle, and place the object there in an unfamiliar environment with open-vocabulary object and receptacle categories (Yenamandra et al., 2023). The implementation adapts sensing and action interfaces while retaining the shared object-relative geometric alignment principle. Both GATE variants retain the official RL placement weights; the adapted variant additionally modifies inference-time action sampling and post-release settling.

Baselines. We compare GATE against representative baselines, including RL-based HomeRobot (Yenamandra et al., 2023), training-based MoManipVLA (Wu et al., 2025b), training-free MoTo (Wu et al., 2025a), and MLLM-based UniLM-Nav (Zhang et al., 2026).

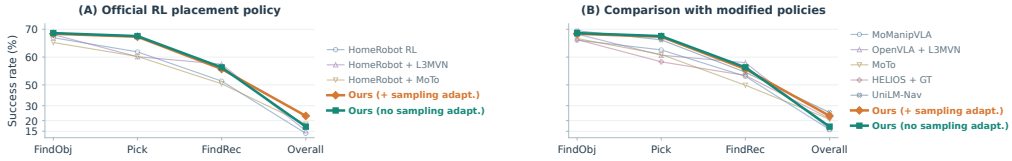


Figure 6: OVMM success rates on a shared nonlinear scale. Both Ours variants appear in each comparison. Full results, including Average, are in Table 18 (Appendix D.4).

Main results. Without changing the official RL placement weights or sampling behavior, GATE achieves 68.64% FindObj and 67.56% Pick. Overall success increases from 14.01% for the reproduced HomeRobot RL baseline to 17.18%. This improvement provides evidence that geometric handoff can benefit a different embodiment and downstream manipulation system.

With lightweight inference-time sampling adaptation, GATE achieves 22.19% Overall and 53.40% Average while retaining the official RL placement weights. Overall improves by 8.18 percentage points over the reproduced HomeRobot RL baseline and by 3.87 percentage points over HomeRobot + MoTo. Compared with UniLM-Nav, which uses an alternative manipulation policy, GATE's Overall and Average scores are lower by only 1.58 and 0.10 percentage points, respectively, indicating comparable benchmark performance.

Together with the real-world humanoid experiments, the OVMM evaluation supports the applicability of the geometric handoff principle across different embodiments and downstream manipulation systems. Object-relative position and orientation provide a shared alignment interface, while platform-specific adapters account for differences in sensing and action execution.

5 CONCLUSIONS

We propose GATE, an object-centric geometric method. GATE uses task-conditioned geometric feedback to reduce mismatch between navigation endpoints and manipulation starting conditions, without additional transition-policy training. Building on GATE, we develop a practical dual-agent loco-manipulation system that coordinates navigation, alignment, and whole-body manipulation. The system achieves higher observed task success than the tested baseline configurations. The OVMM evaluation provides additional evidence of the applicability of the geometric handoff design to a different embodiment and an independently trained manipulation system.

In the future, we will further improve the efficiency and generality of the alignment method. Additionally, we will optimize the agent system architecture to enable more complex and fine-grained instruction analysis and task orchestration, and integrate more powerful manipulation skills to enhance task generalization performance.

AI USE STATEMENT

We used AI tools to assist with modifying project code, analyzing and interpreting experimental results, and revising the manuscript for clarity and readability. We also used AI tools for presentation edits to existing images, such as removing the background from robot photographs in Figure 7. We did not use AI to generate images from scratch. The AI agents used for navigation and alignment are components of the robotic system described in the paper, distinct from these uses of AI during research and manuscript preparation. The authors take responsibility for the final text, code, analyses, and figures.

REFERENCES

Katrina Ashton, Chahyon Ku, Shrey Shah, Saumit Vedula, Tingrui Zhang, Wen Jiang, Kostas Daniilidis, and Bernadette Bucher. HELIOS: Hierarchical exploration for language-grounded interaction in open scenes. *arXiv preprint arXiv:2509.22498*, 2025. URL https://arxiv.org/abs/2509.22498v2.

Junting Chen, Haotian Liang, Lingxiao Du, Weiyun Wang, Mengkang Hu, Yao Mu, Wenhai Wang, Jifeng Dai, Ping Luo, Wenqi Shao, and Lin Shao. OWMM-Agent: Open world mobile manipulation with multi-modal agentic data synthesis. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4330. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/bc97207e3979d1cc23109db0be0e8ed2-Abstract-Conference.html.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, et al. NaVILA: Legged robot vision-language-action model for navigation. In *Robotics: Science and Systems*, 2025. URL https://navila-bot.github.io/.

Hongyu Ding, Ziming Xu, Yudong Fang, You Wu, Zixuan Chen, Jieqi Shi, Jing Huo, Yifan Zhang, and Yang Gao. LaViRA: Language-vision-robot actions translation for zero-shot vision language navigation in continuous environments. *arXiv preprint arXiv:2510.19655*, 2025. URL https://arxiv.org/abs/2510.19655.

Hongyu Ding, Sizhuo Zhang, Ziming Xu, Jinwen Guo, Hongxiu Liu, Xingzhi Cheng, Zixuan Chen, Haifei Qi, Duo Wang, Hao Xu, Jieqi Shi, Yifan Zhang, Jing Huo, Jian Cheng, Yang Gao, and Jiebo Luo. Uni-LaViRA: Language-vision-robot actions translation for unified embodied navigation. *arXiv preprint arXiv:2605.27582*, 2026. URL https://arxiv.org/abs/2605.27582.

Yingdong Hu, Haodong Zhu, Boyuan Zheng, Yihang Hu, Tong Zhang, Zunhao Chen, Junming Zhao, Ruiqian Nai, and Yang Gao. OpenHLM: An empirical recipe for whole-body humanoid loco-manipulation. *arXiv preprint arXiv:2606.22174*, 2026. URL https://arxiv.org/abs/2606.22174.

Haoran Jiang, Jin Chen, Qingwen Bu, Li Chen, Modi Shi, Yanjie Zhang, Delong Li, Chuanzhe Suo, Chuang Wang, Zhihui Peng, and Hongyang Li. WholeBodyVLA: Towards unified latent VLA for whole-body loco-manipulation control. *arXiv preprint arXiv:2512.11047*, 2025. URL https://arxiv.org/abs/2512.11047.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Chenxu Li, Zixuan Chen, Yetao Li, Jiapeng Xu, Hongyu Ding, Jieqi Shi, Jing Huo, and Yang Gao. MoMaStage: Skill-state graph guided planning and closed-loop execution for long-horizon indoor mobile manipulation. *arXiv preprint arXiv:2603.08383*, 2026. URL https://arxiv.org/abs/2603.08383.

Yitang Li, Zhengyi Luo, Tonghe Zhang, Cunxi Dai, Anssi Kanervisto, Andrea Tirinzoni, Haoyang Weng, Kris Kitani, Mateusz Guzek, Ahmed Touati, Alessandro Lazaric, Matteo Pirotta, and

Guanya Shi. BFM-Zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning. *arXiv preprint arXiv:2511.04131*, 2025. URL https://arxiv.org/abs/2511.04131.

Zhengyi Luo, Ye Yuan, Tingwu Wang, et al. SONIC: Supersizing motion tracking for natural humanoid whole-body control. *arXiv preprint arXiv:2511.07820*, 2025. URL https://arxiv.org/abs/2511.07820.

Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Axiang Sun, Junfeng Zhu, Yihao Xu, Xin Li, Zhongxia Zhao, Gaohao Zhou, Dongliang Li, Xiang An, Huajie Tan, and Ziyong Feng. FAST_LIO_LOCALIZATION_HUMANOID. Source code, 2025. URL https://github.com/deepglint/FAST_LIO_LOCALIZATION_HUMANOID.

Ao Wang, Lihao Liu, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. YOLOE: Real-time seeing anything. *arXiv preprint arXiv:2503.07465*, 2025. URL https://arxiv.org/abs/2503.07465.

Kaijun Wang, Liqin Lu, Mingyu Liu, Jianuo Jiang, Zeju Li, Bolin Zhang, Wancai Zheng, Xinyi Yu, Hao Chen, and Chunhua Shen. ODYSSEY: Open-world quadrupeds exploration and manipulation for long-horizon tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18602–18610, 2026. doi: 10.1609/aaai.v40i22.38927. URL https://ojs.aaai.org/index.php/AAAI/article/view/38927.

Songlin Wei, Hongyi Jing, Boqian Li, Zhenyu Zhao, Jiageng Mao, Zhenhao Ni, Sicheng He, Jie Liu, Xiawei Liu, Kaidi Kang, Sheng Zang, Weiduo Yuan, Marco Pavone, Di Huang, and Yue Wang. Ψ_0: An open foundation model towards universal humanoid loco-manipulation. *arXiv preprint arXiv:2603.12263*, 2026. URL https://arxiv.org/abs/2603.12263.

Zhenyu Wu, Angyuan Ma, Xiuwei Xu, Hang Yin, Yinan Liang, Ziwei Wang, Jiwen Lu, and Haibin Yan. MoTo: A zero-shot plug-in interaction-aware navigation for general mobile manipulation. *arXiv preprint arXiv:2509.01658*, 2025a. URL https://arxiv.org/abs/2509.01658.

Zhenyu Wu, Yuheng Zhou, Xiuwei Xu, Ziwei Wang, and Haibin Yan. MoManipVLA: Transferring vision-language-action models for general mobile manipulation. *arXiv preprint arXiv:2503.13446*, 2025b. URL https://arxiv.org/abs/2503.13446.

Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. FAST-LIO2: Fast direct LiDAR-inertial odometry. *arXiv preprint arXiv:2107.06829*, 2021. URL https://arxiv.org/abs/2107.06829.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.09414.

Tianyu Yang, Yiming Zeng, Wenzhe Cai, Yuqiang Yang, Jiaqi Peng, Hui Cheng, Jiangmiao Pang, and Tai Wang. X-NavDP: Generalizing navigation diffusion policy to novel behavior and embodiments with group q-score reweighted matching. *arXiv preprint arXiv:2607.28560*, 2026. URL https://arxiv.org/abs/2607.28560.

Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, et al. HomeRobot: Open-vocabulary mobile manipulation, 2023. URL https://ovmm.github.io/.

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3MVN: Leveraging large language models for visual target navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023. doi: 10.1109/IROS55552.2023.10342512. URL https://arxiv.org/abs/2304.05501.

Zhecheng Yuan, Tianming Wei, Langzhe Gu, Pu Hua, Tianhai Liang, Yuanpei Chen, and Huazhe Xu. HERMES: Human-to-robot embodied learning from multi-source motion data for mobile dexterous manipulation. *arXiv preprint arXiv:2508.20085*, 2025. URL https://arxiv.org/abs/2508.20085.

Pingrui Zhang, Xianqiang Gao, Yuhan Wu, Kehui Liu, Dong Wang, Zhigang Wang, Bin Zhao, Yan Ding, and Xuelong Li. MoMa-Kitchen: A 100k+ benchmark for affordance-grounded last-mile navigation in mobile manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6315–6326, 2025. URL https://arxiv.org/abs/2503.11081.

Zhuofan Zhang, Tianxu Wang, Guoxi Zhang, Yixiong Lin, Xilin Wang, Hongming Xu, Qing Li, Song-Chun Zhu, and Lifeng Fan. UniLM-Nav: A unified framework for zero-shot last-mile navigation. *arXiv preprint arXiv:2607.06537*, 2026. URL https://arxiv.org/abs/2607.06537v2.

A HARDWARE

This section documents the humanoid platform setup and hardware used in data collection, trainning and inference.

A.1 EMBODIMENT SETUP

The real-world embodiment is built on a Unitree G1 EDU version with two Unitree Dex1-1 grippers. To enhance the overall performance of navigation and manipulation, we have implemented a series of modifications mainly on camera.

The system uses four RGB views: head, chest, left wrist, and right wrist. During the navigation and alignment stages, we primarily use RGB-D inputs from the head and chest cameras, whereas during the VLA execution phase, we utilize RGB inputs from all four cameras. Table 1 and Figure 7 present the detailed camera specifications and their mounting configurations.

In addition to camera inputs, we also utilize the G1's onboard MID360 LiDAR for self-localization. To minimize latency, we currently use a wired connection between the robot and the PC. In the future, we plan to utilize an onboard Jetson equipped with a wireless network card to enable a wider operational range.

Table 1: Camera details.

Location	Sensor model	Input used	Resolution	Frame rate
Head	Orbbec Gemini 345Lg	RGB-D	480*640	30Hz
Chest	Realsense D435i	RGB-D	480*640	30Hz
Left wrist	Realsense D405	RGB	480*640	30Hz
Right wrist	Realsense D405	RGB	480*640	30Hz



(a) Chest camera **(b) Head camera**

Figure 7: Mounting locations of the (a) chest and (b) head cameras on the Unitree G1.

A.2 TRAINING AND INFERENCE PLATFORM

We perform inference for the entire system on a local PC with 1*NVIDIA RTX 3090(24G) and 32GB RAM, including the VLA policy, motion controller, navigation policy, and the associated image recognition and self-localization algorithms, while we use internet API for all agent request. Furthermore, we utilize a cloud server equipped with 8*A800 GPUs for all VLA training, and conduct all data collection using the PICO 4 Ultra.

B GATE: ALGORITHM AND IMPLEMENTATION DETAILS

This section specifies GATE on G1 and its current default settings. The module takes a manipulation instruction and camera observations, selects task-relevant geometric references, and adjusts the

base before requesting a manipulation handoff. The Stretch/OVMM adaptation is summarized in Section B.4.

B.1 TASK-CONDITIONED GEOMETRIC REFERENCES

Reference roles. For each `ALIGN` attempt, the VA receives the manipulation instruction ℓ_m and the current chest-camera image to select a position reference r_p and an orientation reference r_ψ. The orientation reference supplies a visible straight structural edge. It may be the position reference itself or a supporting object. A regular, clearly visible non-floor support is preferred when available. The roles describe two constraints and need not denote two distinct objects. Navigation landmarks, unrelated objects, and objects inferred only from prior state are excluded.

Grounding interface. The VA returns the two names, visibility flags, visual evidence, and a tight position-reference bounding box in normalized $[0, 1000]$ image coordinates. A `FOUND` result requires a valid position box and both roles to be visibly identifiable; otherwise the result is `NOT_FOUND`. The names configure text-prompted YOLOE detection and segmentation. The position box is passed to the control service. The tracker initializes from text detections. An unavailable grounding result does not start base motion and is returned to the handoff/recovery logic.

For example, when the instruction is to put a pill bottle into a blue basket on a desk, the basket can supply r_p and the desk can supply r_ψ. This selection prepares the base relative to the destination and its support; it does not make the bottle an additional alignment reference.

B.2 COORDINATE FRAMES AND GEOMETRIC ESTIMATION

RGB-D coordinates and calibration. Depth is aligned to the RGB image and converted to meters using the sensor depth scale. The implementation checks image dimensions, depth alignment, and a positive depth scale. The camera frame uses right, down, and forward axes; the body frame uses forward, left, and up axes. For intrinsics (f_x, f_y, c_x, c_y), a pixel (u, v) with depth z is mapped as

$$\mathbf{p}_C(u, v, z) = \left[\frac{(u - c_x)z}{f_x}, \frac{(v - c_y)z}{f_y}, z\right]^\top, \qquad \mathbf{p}_B = R_{BC}\mathbf{p}_C + \mathbf{t}_{BC}. \tag{7}$$

The implemented rotation is $R_{BC} = R_z(\gamma)R_y(-\beta)R_x(\alpha)R_0$, where (α, β, γ) are the configured camera roll, pitch, and yaw, and $R_0\mathbf{p}_C = (z_C, -x_C, -y_C)^\top$. The translation is the configured forward and lateral offset, with zero vertical offset in this base-position model. Thus the reported rightward displacement is the negative body-left coordinate. Positive base v_x moves forward, positive v_y moves left, and positive angular velocity turns left. For yaw estimation, this calibrated transformation is applied before retaining the body forward and left coordinates. Here R_{BC} describes camera mounting calibration; it is not an estimated robot-to-object rotation.

Position statistics. The position-reference mask is eroded, and the pixels within the eroded mask are filtered for validity and depth. Let $\mathcal{S}$ denote the retained pixels. The forward measurement is the arithmetic mean of their transformed forward coordinates. Lateral centering instead uses the bounding-box horizontal center u_b, the median retained vertical coordinate $\tilde{v}$, and the median retained depth $\tilde{z}$:

$$x_t = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}[\mathbf{p}_{B,i}]_x, \qquad r_t = -[R_{BC}\mathbf{p}_C(u_b, \tilde{v}, \tilde{z}) + \mathbf{t}_{BC}]_y. \tag{8}$$

Here $r_t > 0$ means the target is to the robot's right. This lateral anchor is not the centroid of the segmented point cloud. Empty masks, insufficient valid depth, or incompatible RGB-D inputs produce invalid observations rather than zero position errors.

Depth-based planar edge orientation and yaw feedback. For the orientation-reference mask, the estimator retains the largest connected component, fills holes, and dilates it. It then extracts RGB edges using Gaussian smoothing, Canny detection, and a probabilistic Hough transform within the mask region. When the roles differ and a position-reference mask is available in the same head-camera image, its dilated neighborhood is excluded. This exclusion is omitted when that mask is unavailable or the roles coincide; a chest-image mask is not applied to the head image. Candidate segments must satisfy length and depth-validity requirements. Among these candidates, the segment

with the smallest median depth is selected, breaking depth ties in favor of a longer segment. After selecting the reference segment using the rules above, we use its valid depth samples to compute orientation feedback. Each sample is back-projected using camera intrinsics and transformed into the body frame. We then retain only the forward and left components to estimate the segment direction and its normal in the body horizontal plane. Let $\mathcal{S}_{\ell,t}$ denote the valid depth samples associated with the selected image segment. For sample i with pixel coordinates (u_i, v_i) and metric depth z_i, define

$$\mathbf{q}_i = \Pi_{xy}\left(R_{BC}\mathbf{p}_C(u_i, v_i, z_i) + \mathbf{t}_{BC}\right), \qquad \Pi_{xy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{9}$$

Here $\mathbf{p}_C$ is given by Eq. (7), and $\mathbf{q}_i$ contains body-forward and body-left coordinates. The camera-to-body rotation is applied before the horizontal components are selected. Let $\mathbf{c}_t$ be the mean of these planar samples and $\mathbf{d}_t = (d_{x,t}, d_{y,t})^\top$ their unit line direction, obtained from the dominant right singular vector of the centered sample matrix:

$$\mathbf{c}_t = \frac{1}{|\mathcal{S}_{\ell,t}|}\sum_{i\in\mathcal{S}_{\ell,t}} \mathbf{q}_i, \qquad A_t = [(\mathbf{q}_i - \mathbf{c}_t)^\top]_{i\in\mathcal{S}_{\ell,t}}, \qquad A_t = U\Sigma V^\top, \quad \mathbf{d}_t = V_{:,1}. \tag{10}$$

Projected edge lengths below 0.20 m are rejected. The two signs of the line direction represent the same unoriented edge. We form a normal and choose its sign using the edge center:

$$\mathbf{n}_t^{(0)} = (-d_{y,t}, d_{x,t})^\top, \qquad \mathbf{n}_t = \begin{cases} -\mathbf{n}_t^{(0)}, & (\mathbf{n}_t^{(0)})^\top \mathbf{c}_t < 0, \\ \mathbf{n}_t^{(0)}, & \text{otherwise.} \end{cases} \tag{11}$$

The raw yaw correction is

$$e_{\psi,t} = \operatorname{atan2}(n_{y,t}, n_{x,t}). \tag{12}$$

This angle is measured from the body forward axis to the edge normal directed toward the edge. Positive values command a left turn; zero corresponds to the body forward axis aligning with that normal and hence being perpendicular to the edge in the horizontal plane. Yaw feedback is computed from the selected edge normal in the body horizontal plane after depth back-projection and camera-to-body calibration. The intermediate camera coordinates have three components, but the orientation calculation uses only the body forward and left coordinates. It neither estimates a full object-relative pose nor uses image-plane edge inclination directly as yaw.

Small changes in the edge-normal yaw correction $e_{\psi,t}$ are filtered with an exponential moving average. An inter-frame raw yaw-error change exceeding 10° resets the filter to the new value; position measurements use the latest observation directly. Let $\bar{e}_{\psi,t}$ denote the filtered yaw error and θ_t the measured robot heading. When both are available, the desired heading is refreshed as

$$\theta_t^* = \operatorname{wrap}(\theta_t + \bar{e}_{\psi,t}), \qquad e_{\mathrm{ori},t} = \operatorname{wrap}(\theta_t^* - \theta_t). \tag{13}$$

During a brief edge loss, the last desired heading is retained and the error is propagated using fresh robot heading. Without heading telemetry, a valid depth-based edge-normal estimate still supplies direct geometric yaw feedback. A held yaw error without a fresh valid edge-normal estimate or usable heading is marked untrusted and cannot satisfy the yaw completion test.

Let $c_t \in \{\mathrm{head}, \mathrm{chest}\}$ denote the camera currently supplying position. Its calibrated RGB-D observation supplies x_t and r_t, while the selected head-camera reference edge and its depth samples supply yaw feedback through the edge normal in the body horizontal plane. When chest position is selected, its position estimate is paired with head-camera edge-normal yaw feedback; the two position estimates are not averaged. The implementation-level counterpart of Eq. (6) is

$$\mathbf{e}_t = [x_t - d_{c_t}^*,\ r_t,\ e_{\mathrm{ori},t}]^\top. \tag{14}$$

Here $d_{c_t}^*$ is the standoff configured for the selected position camera. Both position cameras use the same standoff and approach cutoff. A change of position source refreshes the position estimate and resets position-related stability counters.

Recorded bounding-box, mask, and edge overlays. Figure 8 shows a paired chest/head capture from an archived alignment run. The green bounding boxes identify the basket used as the position reference, while cyan shading marks the table mask. The head view additionally shows the table-edge segment in red and its endpoints in yellow, illustrating the visual evidence for orientation feedback. The chest view provides a clear target observation; the basket is partially visible at the top of the head view.



(a) Chest: target bbox and table mask (b) Head: bbox, mask, and table edge

Figure 8: GATE perception examples from paired chest and head views.

B.3 Alignment Procedure and Agent Handoff

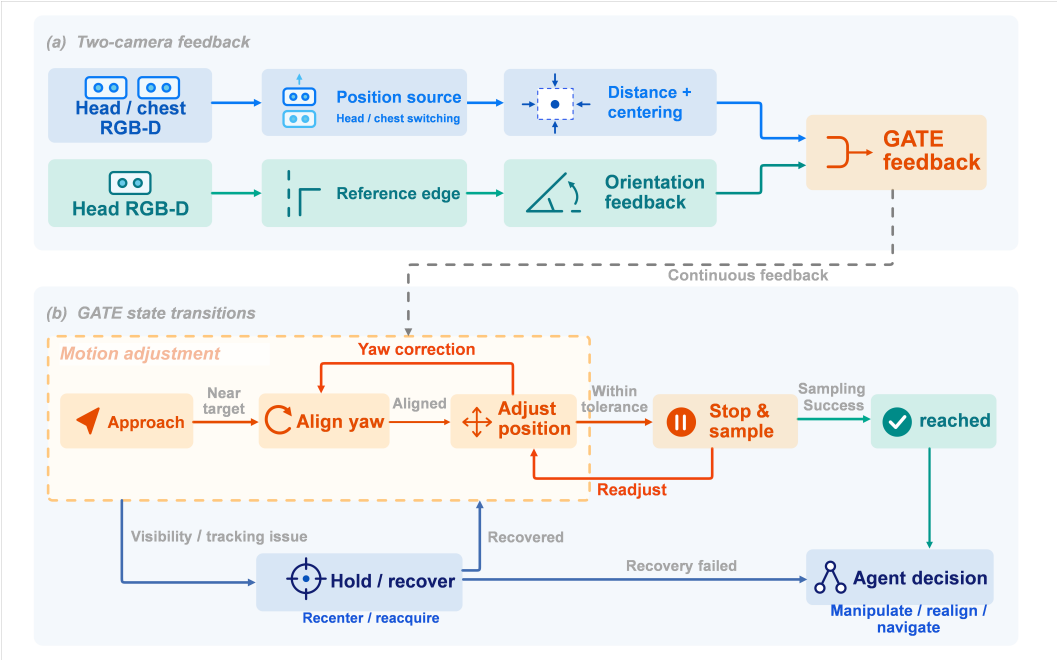


Figure 9: GATE feedback and simplified state transitions. (a) Position feedback can switch between the head and chest RGB-D cameras. (b) Geometric adjustment includes correction and observation-recovery loops, followed by stationary sampling with a budget of 30 samples. Persistent deviations trigger readjustment. The controller returns its outcome to the agent, which decides whether to manipulate, realign, or resume navigation. Detailed switching and sampling rules are provided in this section.

Staged control. Figure 9 illustrates the camera feedback and state transitions, and Algorithm 1 summarizes geometric alignment and the return of its result to the agent. When the position reference is farther than the approach cutoff of the selected position camera, the robot approaches using bounding-box centering without requiring edge orientation. Near the reference, coarse and fine yaw correction are interleaved with visibility recovery; three consecutive trusted, in-tolerance yaw observations permit translation. Translation prioritizes lateral correction and then forward/backward standoff correction, avoiding simultaneous longitudinal and lateral commands in this stage. If an edge is lost during translation, a correction using the retained desired heading can precede further

translation; a final yaw correction is also made when position is within tolerance but orientation is not. These are conditional state transitions, not a fixed action sequence traversed in every attempt.

Camera selection and switching. GATE checks text detections independently in both cameras and initially prefers the head camera when its position reference is detected. This initial eligibility check does not require a valid orientation edge. During control, it retains the current position source while its observation is valid. If position observation fails and the current state permits substitution, it tries the other camera without restarting the control stage. Chest-camera position is combined with yaw feedback computed from the selected head-camera edge normal in the body horizontal plane. Once this combination is established, a dropped chest detection is handled by the persistence rules in Table 2 rather than an immediate switch on one missed frame.

A position-source substitution differs from restarting a camera tracking attempt. If recovery requires a new attempt, GATE stops motion, reinitializes tracking and the controller, and retries text detection in head-then-chest order; exhaustion of that recovery cycle returns a failure. During chest tracking, a background head monitor can supply orientation without changing the position source. When the roles differ, head-camera orientation does not require the position object to remain visible in that view. Reacquiring the position object in the head view therefore does not by itself force a valid chest-position/head-orientation observation to be abandoned.

Algorithm 1 G1 GATE geometric alignment and return to the agent

Require: Manipulation instruction ℓ_m, RGB-D streams, calibration, default settings
Infer (r_p, r_ψ) and the position box from ℓ_m and the current image
if grounding is available **then**
    Initialize per-camera text tracking; prefer an eligible head-camera position source
    while the alignment skill is active **do**
        Acquire position from the selected RGB-D camera; back-project depth samples of the selected head-camera edge, apply camera-to-body calibration, and compute planar edge-normal yaw
        Apply state-dependent source switching, loss handling, and observation-age checks
        Recover tracking or restart an attempt as needed
        Compute position errors and trusted yaw feedback from valid planar edge-normal estimates or heading propagation
        Execute the active approach, recentering, yaw, translation, or stopped state
        Resume adjustment if the stopped-pose deviation condition is met
        Terminate on completion, unrecoverable failure, cancellation, or applicable timeout
    end while
    Stop motion and retain the controller's termination state and reason
else
    Record unavailable grounding without starting the controller
end if
Return the controller result to the agent (`reached` on geometric completion)
The agent decides whether to manipulate, realign, or return to navigation

Control and observation recovery. During approach, if the position-reference bounding-box center is near the image boundary, the robot rotates to recenter it while continuing to approach. During visual yaw adjustment, visibility recovery instead uses lateral motion. A restart from chest to head tracking can additionally trigger a short forward motion to recover visibility if the target center is in the top 10% of the head image.

Table 2 summarizes observation loss during motion adjustment. The specialized chest-position and head-edge loss windows apply after chest position has been paired with head orientation. Loss counts refer to processed frames, not durations implied by the command publication rate. Observation events older than 0.40 s are discarded; a gap of more than 0.40 s since the last accepted observation or invalid-observation event produces a zero-command hold until updates resume.

Completion and stopped-pose sampling. After position and orientation adjustment, five consecutive processed observations must meet the forward, lateral, and trusted orientation tolerances before stopped-pose sampling begins. The default sampling budget is 30 observations; ten consecutive out-of-tolerance observations trigger readjustment.

Table 2: Observation-loss handling in GATE on G1.

Condition	Controller response
Position observation invalid, substitution permitted	Try the other camera; a valid replacement updates the position source without restarting the stage.
Chest position target missing after pairing with head orientation	Counts 1–9 retain the current command under transient-loss handling; 10–19 hold zero. At 20, select head position within a head tracking attempt, or restart detection through the head camera if the active attempt is chest-based.
Head edge missing while chest position remains valid	Counts 1–9 allow heading propagation when telemetry is usable; 10–19 hold zero. At 20, restart reference detection. Chest edges do not replace head orientation.
Other invalid observations	Hold zero; persistent loss or hard errors invoke recovery or terminate the controller. Tracking IDs may be reacquired before a camera restart is necessary.

Handoff to the agent. Once geometric alignment is complete, GATE stops the base and returns `reached` to the agent. The agent then determines whether to proceed with manipulation, realign, or return to navigation.

B.4 Parameters and Platform Adaptations

Current defaults. Table 3 reports G1 launch defaults and controller constants. The alignment module requires no additional transition-policy training; it still depends on calibrated sensing and configured tolerances.

Table 3: Current G1 defaults. Frame counts refer to processed observations, not necessarily 20-Hz camera measurements.

Parameter	Default value
Target standoff	0.90 m
Approach cutoff	1.10 m
Forward / lateral tolerance	0.10 / 0.08 m
Orientation-error tolerance	3°
Minimum nonzero translation speed	0.40 m/s
Maximum lateral speed	0.40 m/s
Coarse / fine yaw speed	0.30 / 0.10 rad/s
Minimum nonzero yaw speed	0.10 rad/s
Approach recentering yaw speed	0.30 rad/s
Horizontal guard / recovery fraction	0.25 / 0.30
Command publication rate	20 Hz
Observation-event freshness threshold	0.40 s
Controller motion runtime limit	180 s
Stopped samples / deviation streak	30 / 10 frames
YOLOE confidence / input size	0.25 / 640 pixels
Convergence / yaw-lock count	5 / 3 frames
Visibility-recovery count / approach recenter interval	3 frames / 0.50 s
Chest-position / head-edge loss: hold / recovery begins	10 / 20 frames
Edge-normal yaw EMA weight / reset threshold	0.35 / 10°
Heading telemetry age limit	0.25 s

The RGB edge extractor uses a 5×5 Gaussian kernel, Canny thresholds 50/150, Hough vote threshold 25, and a maximum line gap of 12 pixels. The default segmenter checkpoint is `yoloe-26m-seg.pt`. These perception settings are shared across task instructions.

Stretch/OVMM adaptation. Both embodiments expose object-relative position and orientation feedback to an alignment controller before manipulation, but their observation and execution interfaces differ. G1 uses text-prompted YOLOE tracking from head/chest RGB-D and continuous base commands executed through SONIC. The OVMM adapter obtains target and support observations from simulator RGB-D and visible semantic masks. It converts servo velocities into bounded

waypoint displacements while accounting for the simulator's minimum executable translations and turns. It also contains reference reacquisition and placement-surface geometry specific to Stretch. Benchmark-specific execution rules and downstream placement-policy settings are described in Section D.4.

C Agent and Skills Details

C.1 Runtime Configuration

We leverage internet-scale model APIs to serve as the backend for the LA and VA modules. Table 4 records some key API settings for LA and VA.

Table 4: API settings for LA and VA.

Setting	Reference configuration
LA / VA model identifiers	`qwen3.8-max` / `qwen3.5-27b`
Thinking mode	`no_think`
Temperature	0
Maximum output tokens	LA: 1800; VA 768

C.2 Language Agent (LA)

LA plans navigation, alignment, and manipulation throughout the mission using current observations, a task checklist, and execution feedback. Table 5 summarizes its response; the prompt appears in Appendix E.4.1.

Table 5: LA response for multi-skill missions.

Field	Meaning
`global_target`	Current navigation objective or last relevant target; updated as the mission progresses.
`progress_analysis`	Assessment of task progress.
`updated_todo_list`	Markdown checklist of completed and pending actions.
`reasoning`	Rationale for the next action.
`decision`	`EXECUTE`, `COMPLETE`, or `FAIL`.
`skill`	`MOVE_TO`, `ALIGN`, or `MANIPULATE`; null for `COMPLETE`/`FAIL`.
`skill_args`	`MOVE_TO`: `view_direction` and `target`; `ALIGN`: `skill_key` and `task`; `MANIPULATE`: `skill_key` from the latest successful ALIGN; empty for `COMPLETE`/`FAIL`.
`expected_postcondition`	Expected physical result of the selected action.

Image order. LA receives current views in the order front, front-right, right, left, and front-left (front only in fixed-front mode), followed by up to five accepted navigation-result images from oldest to newest, with a label after each image. Figure 10 illustrates the image layout with original inputs from a test run.

Task memory. LA records the mission as a checklist of navigation, alignment, and manipulation steps and carries it into each planning round. Using this checklist together with current images and execution feedback, LA selects the next unfinished step and may revise the remaining plan without changing completed steps. It returns `COMPLETE` when the mission is finished and all steps are checked off. Detailed examples are provided in Appendix E.3.

Current observations: five directional views at planning step 4



Front ($0°$) | Front-right ($-45°$) | Right ($-90°$) | Left ($+90°$) | Front-left ($+45°$)

Retained history: oldest to newest



PLAN-3 | PLAN-2 | PLAN-1

Figure 10: Example input to the Language Agent: five current directional views in message order (top), together with three retained observations from previously accepted navigation steps, ordered from oldest to newest (bottom).

C.3 VISION AGENT (VA)

Table 6 specifies the three VA response modes: target grounding, alignment-reference grounding, and post-execution checking.

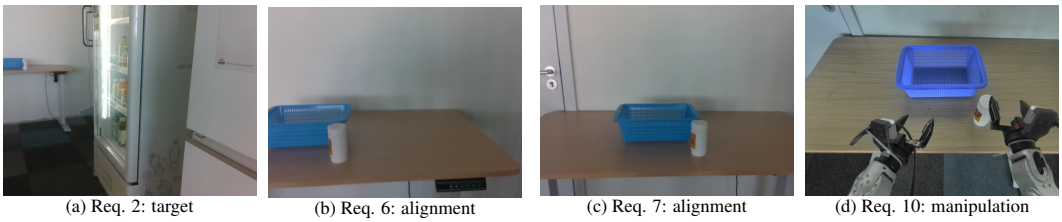
Table 6: Core VA response fields and skill transitions. G: target grounding; A: alignment-reference grounding; P: postcheck. Each mode requires only its applicable top-level fields.

Field	Mode	Meaning and constraints
Shared fields		
`mode`	G, A, P	`GROUNDING`, `ALIGN_GROUNDING`, or `POSTCHECK`.
`status`	G, A	`FOUND` or `NOT_FOUND`; A requires both references visible and a valid position box for `FOUND`.
	P	`SATISFIED`, `NOT_SATISFIED`, or `UNKNOWN`.
`visual_evidence`	A, P	Text supporting reference selection or postcondition checking.
Mode-specific fields		
`bbox_2d`	G	$[x_1, y_1, x_2, y_2]$; null for `NOT_FOUND`.
`target`, `yaw_align_target`	A	Position and orientation reference objects. Both contain `name` and `visible`; only `target` also contains `bbox_2d`. Names are English detector labels and visibility is boolean. The position box uses the coordinates above and is null when invisible.
`transition`	P	Skill-dependent value below; `UNKNOWN` status requires `UNKNOWN` transition.
Skill transitions (all also allow `UNKNOWN`)		
`MOVE_TO`	—	`CONTINUE_NAVIGATION`, `READY_TO_ALIGN`; computed from grounding, depth, and controller feedback.
`ALIGN`	—	`RETRY_ALIGN`, `RETURN_TO_NAVIGATION`, `READY_TO_MANIPULATE`.
`MANIPULATE`	—	`SATISFIED` maps to `TASK_COMPLETE` (current operation); `NOT_SATISFIED` maps to `CONTINUE_MANIPULATION`.

Visual decision semantics. `NOT_FOUND` means that the requested target or alignment references cannot be grounded from the visual evidence. `NOT_SATISFIED` means the requested condition is unmet, whereas `UNKNOWN` denotes insufficient evidence. For manipulation, `SATISFIED` completes the operation and `NOT_SATISFIED` continues execution. Navigation and alignment readiness re-

quire a complete passing camera view; partial evidence from two cameras is not combined into one passing view.

Recorded VA input images. Figure 11 shows a example input list for each VA calling. Request 2 grounds "the desk with the blue basket" after the robot turns toward the selected direction. Request 6 selects alignment references from the manipulation instruction. Request 7 checks the selected "blue basket" and "desk" roles in a fresh chest view after alignment. Request 10 checks the manipulation postcondition in a fresh head view while VLA execution continues. The panels illustrate input context and do not independently establish task success.



(a) Req. 2: target grounding (b) Req. 6: alignment references (c) Req. 7: alignment postcheck (d) Req. 10: manipulation postcheck

Figure 11: Example single-image inputs to the Vision Agent for (a) target grounding, (b) alignment-reference selection, (c) alignment verification, and (d) manipulation monitoring. Panels (a–c) use chest-camera views; panel (d) uses a head-camera view captured during manipulation.

C.4 Skill interface

The skill interface connects LA decisions to navigation, GATE alignment, and VLA execution. It constructs skill-specific goals and combines controller outcomes with VA evidence to determine whether to proceed or return to planning.

- **MOVE_TO:** turn toward the selected view, ground the requested waypoint in a fresh image, and use calibrated depth to construct a navigation goal; controller completion and visual/depth checks determine arrival.
- **ALIGN:** select a manipulation skill and pass its semantic description to VA to choose position and orientation references for GATE; manipulation readiness requires geometric convergence and visibility of both references in at least one camera view.
- **MANIPULATE:** execute the skill selected by the latest successful ALIGN; VA checks its physical postcondition and reports continuation, operation completion, or insufficient evidence.

In the current multi-skill loop, LA may interleave navigation and operations, including starting with ALIGN when already near the target. Every manipulation requires its own fresh alignment. Operation completion returns control to LA; `COMPLETE` ends the whole mission only after its checklist is finished. The corresponding prompt is given in Appendix E.4.1.

C.5 SONIC details

We use the `sonic_v1_1` policy variant. The two modes are mutually exclusive, so only one motion reference source controls the robot at a time. Table 7 and Table 8list the input and control frequency of two SONIC control mode.

Planner mode. Navigation and alignment execute in planner mode. Upstream modules provide planar velocity and heading commands at 20 Hz, and SONIC converts them into continuous whole-body motion references. The kinematic planner generates references at 30 Hz, which are resampled to the 50 Hz tracking rate.

Pose mode and mode transitions. Before manipulation, the runtime keeps SONIC in planner mode while smoothly moving the upper body to the VLA initial pose over approximately 3 s at 50 Hz. It then transfers control to pose mode, where VLA action chunks are streamed to SONIC at 50 Hz. After manipulation, the runtime stops the VLA stream and returns SONIC to planner mode.

Table 7: SONIC mode inputs. Both modes share the downstream whole-body tracking controller.

Mode	Field	Dimension	Role
Planner	`mode`	1	Selects the locomotion or posture mode.
Planner	`movement`	3	Movement-direction vector (x, y, z).
Planner	`facing`	3	Desired facing-direction vector.
Planner	`speed`	1	Scalar translational speed.
Planner	`height`	1	Desired body-height command.
Planner	`upper_body_position`	17	Optional upper-body joint-position targets.
Planner	`upper_body_velocity`	17	Optional upper-body joint-velocity targets.
Planner	`left_hand_joints`	7	Optional left-hand joint-position targets.
Planner	`right_hand_joints`	7	Optional right-hand joint-position targets.
Pose	`token_state`	64	Continuous SONIC whole-body motion token.
Pose	`left_hand_joints`	7	Left-hand joint-position targets.
Pose	`right_hand_joints`	7	Right-hand joint-position targets.

Table 8: Command and reference timing used by the two SONIC deployment modes. The VLA request wait excludes inference time; online scheduling is detailed in Appendix C.7.

Mode	Quantity	Value
Planner	Upstream command input	20 Hz
Planner	Generated reference	30 Hz
Pose	VLA request wait after previous result	0.5 s
Pose	VLA action horizon	50 steps
Pose	Upstream action input	50 Hz
Both	Reference tracking	50 Hz

C.6 Navigation skills

Navigation goal construction. Given the target bounding box, we use its center pixel and the chest-camera intrinsics to determine the target direction. Metric depth is predicted from the chest RGB image using Depth Anything V2 (Yang et al., 2024). We collect five consecutive depth frames, compute the median valid depth within a 7×7 window around the box center for each frame, and require at least three valid measurements. Their ranges are averaged, while the target direction is obtained from the middle frame. The resulting direction and range define a two-dimensional navigation goal in the robot base frame.

X-NavDP deployment. We use X-NavDP (Yang et al., 2026) to predict a robot-local planar trajectory from the chest RGB-D observation and a point goal. At the beginning of a navigation segment, the measured base-frame point goal is transformed into the FAST-LIO2 world frame (Xu et al., 2021) and kept fixed. We use the G1 localization implementation of Sun et al. (2025). At each subsequent inference step, the current Fast-LIO pose is used to transform this fixed goal back into the robot frame. The resulting local goal, the synchronized RGB-D observation, the camera intrinsics, and the pose at the image timestamp are passed to X-NavDP. Table 9 summarizes the deployed interface and trajectory-processing parameters.

MPC controller. The predicted trajectory is transformed to the world frame using the Fast-LIO pose closest to the image timestamp. Our nonlinear model predictive controller then tracks this world-frame reference with the unicycle state $\mathbf{x} = [x, y, \theta]^\mathsf{T}$ and control $\mathbf{u} = [v, \omega]^\mathsf{T}$. Its dynamics are $\dot{x} = v\cos\theta$, $\dot{y} = v\sin\theta$, and $\dot{\theta} = \omega$, discretized by fourth-order Runge–Kutta integration. The controller solves

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k\in\mathcal{S}} \left\|\mathbf{x}_k - \mathbf{x}_k^{\mathrm{ref}}\right\|_Q^2 + \sum_{k=0}^{N-1} \|\mathbf{u}_k\|_R^2, \qquad \mathcal{S} = \{0, 3, \ldots, 30\}, \tag{15}$$

subject to the discretized dynamics and the control bounds

$$|v_k| \le 0.4\,\mathrm{m/s}, \qquad |\omega_k| \le 0.7\,\mathrm{rad/s}. \tag{16}$$

The position and control weights are $Q = \mathrm{diag}(10, 10, 0)$ and $R = \mathrm{diag}(5.0, 0.03)$, respectively. Algorithm 2 shows the receding-horizon procedure, and detailed parameters are given in

Table 9: X-NavDP deployment, interface, and trajectory-processing parameters.

Parameter	Value	Description
Valid depth range	$[0.1, 5.0]$ m	Values outside the interval are set to zero.
Depth encoding scale	10^4	Metric depth encoded as 16-bit PNG.
Goal range, forward	$[0, 10]$ m	Clipping range for local goal x.
Goal range, lateral	$[-10, 10]$ m	Clipping range for local goal y.
Goal tolerance	2.0 m	Distance for declaring the fixed world goal reached.
Stop threshold	-4.0	Stop threshold supplied when resetting X-NavDP.
Trajectory output	$N \times 2$	Robot-local planar waypoints.
Initial points removed	3	Removes unstable points close to the robot.
Interpolation ratio	50	Linear densification before MPC tracking.

Algorithm 2 Asynchronous receding-horizon MPC for tracking the X-NavDP trajectory.

Require: Local trajectory τ, image-time pose $\mathbf{x}^I$, current pose $\mathbf{x}_t$

1: Remove the first three points from τ
2: Transform τ to the world frame using $\mathbf{x}^I$ and interpolate it by $50\times$
3: Find the closest reference point to $\mathbf{x}_t$ and advance by 10 points
4: Estimate path length L and maximum smoothed curvature $\kappa_{\max}$
5: $v_L \leftarrow v_d \min(L/L_{\mathrm{ref}}, 1)$
6: $v_\kappa \leftarrow c_\kappa \omega_{\max} / \max(\kappa_{\max}, \epsilon)$
7: $v_{\mathrm{ref}} \leftarrow \max(v_{\min}, \min(v_{\max}, v_L, v_\kappa))$
8: Sample 11 reference states at arc-length spacing $v_{\mathrm{ref}}(3\Delta t)$
9: Solve the 30-step constrained nonlinear program using the previous solution as a warm start
10: Apply the first optimized control (v_0, ω_0) as $(v_x, v_y, \omega_z) = (v_0, 0, \omega_0)$

Table 10. The MPC output (v, ω) is converted by the runtime into the SONIC body-frame command $(v_x, v_y, \omega_z) = (v, 0, \omega)$.

New trajectories update the world-frame reference asynchronously, while the MPC is solved at 10 Hz and the latest valid velocity command is published at 20 Hz. This decouples policy inference latency from the continuous navigation-control loop.

Table 10: Parameters of the deployed nonlinear MPC trajectory controller.

Parameter	Value	Unit or role
MPC update frequency	10	Hz
Navigation command frequency	20	Hz
Prediction horizon N	30	steps
Integration interval Δt	0.1	s
Prediction duration	3.0	s
Reference sampling gap	3	MPC steps
Number of reference states	11	states
Look-ahead offset	10	interpolated points
Desired linear velocity v_d	0.4	m/s
Linear velocity bound $\lvert v \rvert$	0.4	m/s
Angular velocity bound $\lvert \omega \rvert$	0.7	rad/s
Reference-length scale L_{ref}	2.0	m
Minimum desired velocity $v_{\min}$	0.0	m/s
Position weights Q	$\mathrm{diag}(10, 10, 0)$	x, y, and yaw
Control weights R	$\mathrm{diag}(5.0, 0.03)$	linear and angular
Curvature speed gain c_κ	0.15	speed adaptation
Heading preview duration	0.5	s
IPOPT maximum iterations	100	iterations
IPOPT acceptable tolerance	10^{-8}	solver tolerance

C.7 MANIPULATION SKILLS

This section details demonstration collection, task-specific policy training, and online execution on G1. All manipulation policies use the observation and action interface defined in Section 3.2.2 and the SONIC control interface described in Appendix C.5.

C.7.1 DATA COLLECTION

We collect demonstrations using the PICO whole-body teleoperation setup described in Appendix A. SONIC remains in pose mode throughout collection, allowing the operator to coordinate locomotion, torso motion, arm motion, and grasping within each demonstration.

Each observation contains RGB images from the head, chest, and two wrist cameras, a 43-dimensional whole-body joint configuration $\mathbf{q}_t$, and a three-dimensional body-frame projected gravity vector $\mathbf{g}_t$. The latter two form a 46-dimensional proprioceptive state. We record the 64-dimensional SONIC motion token used during teleoperation and concatenate it with the left and right gripper commands to obtain a 78-dimensional action label. Gripper commands are recorded directly without an additional latent encoder. State and action records follow the 50 Hz control cadence; camera specifications are provided in Appendix A.1.

Demonstrations begin near the workspace approached by the navigation skill and include the local approach needed for manipulation. We vary the initial base position around typical navigation endpoints to broaden the demonstrated starting configurations. The navigation controller's 2.0 m arrival tolerance (Appendix C.6) provides a reference scale for collection, rather than a specified sampling boundary. The robot generally starts facing the task objects, consistent with the orientation sought by GATE. We allocate more demonstrations to skills requiring precise interaction or coordinated whole-body motion. Table 11 lists the demonstration counts and total recorded durations for the T1–T4 training datasets.

Table 11: Demonstration counts and total recorded durations per manipulation skill. T4 uses separate policies for bag pickup and disposal. Duration is the sum of episode frame counts divided by the 50 Hz recording rate, counting each demonstration once across camera views.

Task	Manipulation skill	Demonstrations	Duration (h)
T1	Pill-bottle pick-and-place	238	1.68
T2	Bag packing and box carrying	179	2.15
T3	Pedal-bin operation and bowl emptying	146	1.86
T4	Gift-bag pickup and turning	77	0.41
T4	Pedal-bin opening and bag disposal	188	1.20

C.7.2 TRAINING

Each skill policy is initialized from the pretrained $\pi_{0.5}$ checkpoint (Physical Intelligence et al., 2025) and fine-tuned independently, with all model parameters updated. Images from each camera are resized to 224×224 pixels. Numerical inputs and action targets are normalized, and the 46-dimensional proprioceptive state is discretized and incorporated into the language input through the $\pi_{0.5}$ state-conditioning interface. Each language instruction describes the corresponding manipulation task.

To accommodate the 78-dimensional action representation, we resize and reinitialize the action input and output projection layers, whose pretrained configuration supports 32-dimensional actions, while loading the remaining pretrained weights. The input projection maps the noisy concatenated action vector into the action expert's hidden representation, and the output projection maps it back to the 78-dimensional action space. The flow-matching objective is applied jointly to the motion-token and gripper-command targets.

Each prediction contains $H = 50$ actions with an interval of $\Delta t = 0.02\,\text{s}$, giving a 1-s prediction horizon at the 50 Hz execution rate. Under the approximately 200 ms inference latency measured on our deployment platform, this leaves a remaining action sequence after temporal compensation. We train each policy for 30,000 steps using AdamW with a batch size of 32. The learning rate increases

to 2×10^{-5} over the first 1,000 steps and then follows a cosine decay to 2×10^{-6} at step 30,000. All evaluations use the final checkpoint from the corresponding training run. Training hardware is specified in Appendix A.2.

C.7.3 Inference and Temporal Offset Compensation

Online inference uses the same observation representation as training, with joint and orientation feedback obtained through the G1 SDK. Policy requests run asynchronously while the executor streams the currently cached action chunk at 50 Hz. Each newly received chunk replaces the cached sequence after its starting index is adjusted for the measured delay.

Let τ_t denote the elapsed time from initiation of the policy call to consumption of its result by the execution loop, measured using a local monotonic clock. Given the action interval Δt, the starting index is

$$k_t = \mathrm{clip}\left(\mathrm{round}\left(\frac{\tau_t}{\Delta t}\right), 0, H-1\right). \tag{17}$$

The executor skips the first k_t actions and uses the remaining suffix:

$$\mathbf{A}_t^{\mathrm{exec}} = [\mathbf{a}_{t,i}]_{i=k_t}^{H-1}. \tag{18}$$

This advances execution past the action steps corresponding to the measured policy-call and result-delivery interval. It does not explicitly compensate for sensor latency before the policy call.

On the RTX 3090 platform, policy inference takes approximately 200 ms, and sensor acquisition introduces approximately 30 ms of additional latency. The configured 0.5-s request interval is measured from receipt of the preceding result; consequently, the effective observation-update period also includes the subsequent inference time.

Mode transitions and initialization are described in Appendix C.5, while visual completion checks and execution limits are specified in Appendices C.3 and D.1.

D Experiment Details

This chapter documents the real-world evaluation protocol, navigation success rates and execution times, geometric and visual handoff analysis, OVMM configuration, and additional near-field experiments.

D.1 G1 Task Definitions and Evaluation Procedure

Tasks and progress. Table 12 specifies the VLA tasks and the milestones used for manual Progress annotation. The VLA task descriptions state the physical manipulation objectives, including the local approach; navigation instructions specify the route to the corresponding workspace. Each completed milestone contributes one point. Progress is the number of completed milestones divided by the total number of milestones for the task, expressed as a percentage. Task-level mean Progress averages this percentage over all ten trials, including failures.

Quantitative results. Table 15 summarizes task success, mean Progress, navigation success, and execution time for all task–method pairs.

Initial conditions. For each task–method evaluation, ten trials start from five distinct robot positions, with two different object-placement and background configurations tested at each starting position. At two positions, the target is initially visible and the route leads directly to it; at one position, the target is initially visible but the route includes an intermediate landmark; at the remaining two positions, the target is initially out of view. The task-specific navigation instruction describes the required route.

Execution limits and termination. The experiment launcher starts a trial and its timer on the operator's start command. The archived configurations for Full VLN, Direct VLA, GATE w/o Edge Alignment, and BaselineLAVIRA specify a 600-s total limit, including navigation, alignment, manipulation, model waiting, and retries. An independent supervisor enforces this limit and a 180-s VLA limit measured from the first valid action. Neither a model retry nor a return to navigation resets the trial budget.

Table 12: G1 task descriptions and manually annotated Progress milestones.

Task	VLA task	Progress definition
T1	Move forward to approach the blue basket on the desk, pick up the pill bottle, and place it inside the blue basket.	1. Reach the navigation goal. 2. Pick up the pill bottle. 3. Place the bottle in the basket.
T2	Move forward to approach the open cardboard box, pick up the bag and put it into the box, then carry the box to the open area and place it on the ground.	1. Reach the navigation goal. 2. Pick up the bag. 3. Put the bag in the box. 4. Lift the box. 5. Carry the box to the open area and place it on the ground.
T3	Move forward to approach the bowl on the desk, pick up the bowl, step on the trash-bin pedal to hold the lid open, empty the bowl into the bin, and put the bowl back on the desk.	1. Reach the navigation goal. 2. Pick up the bowl. 3. Press the pedal to open the lid. 4. Empty the contents into the bin. 5. Return the bowl to the desk.
T4	Pick up the white gift bag on the desk, turn while holding the bag to bring it to the side of the body, carry it to the green rubbish bin, step on the pedal to open the lid, and throw the bag into the bin.	1. Navigate to the white gift bag. 2. Pick up the white gift bag. 3. Turn while holding the bag to bring it to the side of the body. 4. Navigate to the rubbish bin. 5. Press the pedal to open the lid. 6. Throw the bag into the bin.

- **Agent and model calls.** The agent allows at most 50 planning steps. LA and VA requests have a configured timeout of 180 s per request. The model-call wrapper makes up to three attempts; request exceptions incur a 10-s wait before another attempt, whereas invalid structured responses are retried with format feedback. Exhausted attempts terminate the run; the independent trial deadline takes precedence where enabled.
- **Navigation and alignment.** Waiting for an individual navigation or alignment controller segment is limited to 180 s; navigation heading correction has a separate 10-s limit. Full VLN and GATE w/o Edge Alignment share the 180-s alignment controller runtime limit, dual-camera switching, target-loss handling and recovery, and agent handoff checks. A returned alignment failure can be evaluated by the agent for another alignment attempt or a return to navigation; an unhandled controller-wait timeout terminates the run. The VLM basepose alignment module has a 180-s deadline.
- **Observation and VLA startup.** Camera acquisition has a 15-s timeout. VLA startup must acknowledge readiness within 6 s; otherwise the agent stops the request and terminates the run.
- **Manipulation and completion checks.** During manipulation, the VLA policy continues executing while the VA periodically checks task completion. Full VLN, Direct VLA, GATE w/o Edge Alignment, and BaselineLAVIRA enforce an independent 180-s limit from the first VLA action, including model waiting; VLM basepose instead checks its 180-s manipulation timeout between execution windows. Execution also terminates when the completion-check or uncertainty limits in Table 13 are reached.
- **NAVILA branch.** NAVILA allows at most 50 navigation actions, with a 30-s model-request timeout and a 3-s limit for confirming that the stop command has been published by the executor. An unsupported action, request timeout, or missing stop confirmation terminates the run. VLA is started only after geometric alignment succeeds. This branch does not call LA or VA; manipulation stops on an operator decision, cancellation, a fault, or the 180-s action budget. Budget expiry records a completion candidate, not automatic physical success.

Outcome annotation. For methods with VA completion checks, VA outputs the task-completion decision, which is manually verified against the physical result. The manually verified outcome

Table 13: Manipulation completion-check settings shared by Full VLN, Direct VLA, GATE w/o Edge Alignment, BaselineLAVIRA, and VLM basepose.

Parameter	Setting
Wait before each VA completion check	5 s; VLA continues executing
Maximum completion-check windows	36
Additional check after an inconclusive result	1
Consecutive inconclusive windows before termination	3, each remaining inconclusive after the additional check

is used for success-rate computation; a software completion flag alone is insufficient. NAVILA outcomes are manually annotated because that branch has no VA. Progress is manually annotated using Table 12. Any configured VLA continuation after the VA completion decision is excluded from the logged completion time described in Section D.2.

Method configurations. Table 14 summarizes the navigation and handoff implementations. The G1 platform, camera suite, VLA manipulation interface, and SONIC execution layer are shared; each task uses its corresponding manipulation policy. The method names follow Table 15. GATE w/o Edge Alignment is an ablation of Full VLN that disables only edge detection and alignment with the detected edge; all other system components and execution settings are shared.

VLM basepose. This baseline adapts the alignment prompts from UniLM-Nav (Zhang et al., 2026) to an MLLM backend and outputs the target position directly. It takes the current RGB-D observation and historical RGB-D observations from navigation, together with their corresponding robot poses, and performs multi-stage reasoning to predict the target base position. The predicted target is executed through the G1 controller.

Table 14: Navigation, alignment, and execution differences in the G1 comparison.

Method	Navigation	Alignment before VLA	Handoff and completion
Full VLN	Dual-agent navigation with X-NavDP and MPC.	GATE: task-conditioned position and orientation references with RGB-D feedback.	Agent checks geometric/visual readiness; VA checks completion.
Direct VLA	Same navigation as Full VLN.	No alignment; invoke VLA after navigation arrival is accepted.	Bypass alignment and its handoff check; retain navigation and completion checks.
GATE w/o Edge Alignment	Same navigation as Full VLN.	Same as Full VLN, with edge detection and alignment with the detected edge disabled.	Same agent handoff checks and VA completion checks as Full VLN.
BaselineLAVIRA	LaViRA waypoint selection, iPlanner, and Pure Pursuit.	Same GATE alignment module.	Retain navigation, alignment, and completion checks.
VLM basepose	Shared agentic navigation with X-NavDP and MPC.	Use multi-stage VLM reasoning to predict a target base position, then execute it through the G1 controller.	Retain agent handoff and VA completion checks; use the separate runtime limits above.
NAVILA	Image-conditioned discrete navigation actions.	GATE controller with position/orientation references specified by the trial configuration.	Require controller alignment success; no LA/VA checks; manual outcome annotation.

D.2 Real-World Quantitative Results

Table 15 combines task and navigation outcomes with the mean logged duration of successful trials. Ours denotes Full VLN.

Table 15: Real-world results over ten trials per task and method. Task SR gives successes/trials (rate); time is the mean logged duration of successful trials, with a dash when none succeeds. T4 navigation is reported as selected (first leg, second leg). Bold denotes the best success, progress, and navigation values within each task. NAVILA completed none of T1–T4.

Task	Method	Task SR ↑	Progress (%) ↑	Nav SR (%) ↑	Time (s)
T1	Direct VLA	6/10 (60%)	66.67	80	143.70
	VLM basepose	2/10 (20%)	43.33	90	164.53
	GATE w/o Edge Alignment	7/10 (70%)	80.00	90	229.82
	BaselineLaViRA	6/10 (60%)	86.67	**100**	282.27
	Ours	**9/10 (90%)**	**90.00**	90	274.38
T2	Direct VLA	0/10 (0%)	18.00	90	—
	VLM basepose	3/10 (30%)	56.00	**100**	168.92
	GATE w/o Edge Alignment	4/10 (40%)	58.00	90	174.99
	BaselineLaViRA	5/10 (50%)	52.00	60	176.30
	Ours	**9/10 (90%)**	**92.00**	**100**	195.82
T3	Direct VLA	0/10 (0%)	26.00	**100**	—
	VLM basepose	4/10 (40%)	60.00	**100**	222.67
	GATE w/o Edge Alignment	2/10 (20%)	44.00	**100**	144.72
	BaselineLaViRA	4/10 (40%)	50.00	90	287.48
	Ours	**8/10 (80%)**	**84.00**	**100**	290.64
T4	Direct VLA	0/10 (0%)	40.00	90 (100, 40)	—
	VLM basepose	2/10 (20%)	48.33	**100** (100, 50)	250.01
	GATE w/o Edge Alignment	1/10 (10%)	50.00	**100** (100, 60)	300.89
	BaselineLaViRA	5/10 (50%)	83.33	**100** (100, 100)	302.84
	Ours	**7/10 (70%)**	**85.00**	**100** (100, 90)	374.42

D.3 Geometric and Visual Analysis of the Handoff

Scope and observation times. The geometric analysis compares Full VLN, Direct VLA, GATE w/o Edge Alignment, and BaselineLAVIRA on T1 and T2, with ten trials per method and task. T1 uses the chest camera at the first VLA input; T2 uses the final pre-VLA head-camera check.

Geometry at the manipulation handoff. Figure 5 shows T1 and T2 in two rows, with four method columns. Within each task, all methods share the coordinate frame, plotting limits, and training-pose density.

T1 reconstructs chest-camera lateral position X and wall-normal distance Z relative to the basket front wall. T2 reconstructs head-camera lateral position X and front distance $D = -Y$ relative to the 41×60 cm opening. The training backgrounds use 197/238 frames for T1 and 179/179 for T2, retaining estimated corners and applying a reprojection RMS threshold of at most 5 pixels.

A training-neighborhood hit is defined by nearest-reference distance in normalized lateral position, normal distance, and wrapped heading. Scales and the 95th-percentile leave-one-out nearest-neighbor threshold are fitted on valid training poses separately for each task. All ten trials remain in the denominator, including those with missing geometry. Full VLN reaches the training neighborhood in 9/10 T1 trials and 8/10 T2 trials, compared with 3/10 and 0/10 for Direct VLA and 8/10 and 4/10 for GATE w/o Edge Alignment. The higher training-neighborhood hit rate of Full VLN relative to GATE w/o Edge Alignment is consistent with edge detection and alignment with the detected

edge improving the match between handoff poses and demonstrated configurations, particularly on T2. BaselineLAVIRA records 8/10 and 4/10 hits, but six T2 trials lack estimable geometry, limiting interpretation of that comparison. A training-neighborhood hit is distinct from physical task success.

Visual distance and physical outcome. The visual-distance comparison uses pre-manipulation head images from Full VLN, Direct VLA, and GATE w/o Edge Alignment. The paired GATE analysis additionally includes the near-field trials in Appendix D.5. These handoff observations are distinct from the initial state before either method acts.

We use each task's frozen deployed vision encoder and a fixed set of training-reference head images: 238 reference demonstrations for T1 and 179 for T2. Let $f_q(I)$ be the normalized spatial descriptor obtained by pooling visual tokens on a 3×4 grid for task q. For its training-reference set $\mathcal{R}_q$, we define

$$d_q(I) = \frac{1}{3} \sum_{j=1}^{3} \left(1 - f_q(I)^\top f_q(J_{q,j}(I))\right), \tag{19}$$

where $J_{q,j}(I)$ is the image from the jth nearest distinct reference demonstration in $\mathcal{R}_q$, ranked by cosine distance. Each eligible trial contributes one distance. Reference membership, checkpoint, and feature extraction are fixed within each task across methods and trial sources. Lower values indicate greater proximity to training references under this representation.

For Direct VLA, we use the final navigation-release head image before `MANIPULATE`; for Full VLN and GATE w/o Edge Alignment, we use the final alignment-release head image. Figure 4(c–d) shows the resulting distances with a shared density-estimation bandwidth within each task. Missing images receive no distance, while the corresponding trials remain in the main evaluation's success denominator.

Table 16: Visual-distance summaries and physical outcomes in the main evaluation. Medians use eligible images; successes / trials use all ten formal trials per method and task, as in Table 15.

Task	Method	Eligible images	Successes / trials	Median distance
T1	Full VLN	9	9/10	0.00578
	Direct VLA	8	6/10	0.01059
	GATE w/o Edge Alignment	9	7/10	0.00853
T2	Full VLN	8	9/10	0.00848
	Direct VLA	9	0/10	0.01241
	GATE w/o Edge Alignment	9	4/10	0.01289

Full VLN has the lowest median visual distance on both tasks (Table 16). On T1, GATE w/o Edge Alignment also reduces the median relative to Direct VLA; on T2, its median is slightly higher despite more physical successes. The outcome-colored points show overlap between successful and failed executions. Thus, visual proximity complements the geometric analysis but does not alone determine manipulation success. Full VLN's eligible images are all from successful trials, so this subset does not support a within-method success–failure comparison.

Paired change through GATE: main and near-field trials. Figure 4(a–b) combines the available Full VLN pairs from the main evaluation and GATE + VLA pairs from the near-field analysis, separately for T1 and T2. The pre-alignment image is the final accepted navigation head check in the main evaluation and the initial head image before GATE in the near-field analysis. The post-alignment image is the final alignment-release head check before manipulation. Each pair matches observations from the same trial. Curves connect measured endpoints; circles and squares mark main and near-field trials, dark bars mark pooled medians, and dashed curves mark an unpassed alignment check. The visual-analysis cohort is fixed to the trials underlying the figure; the completed near-field outcome counts are reported separately in Table 20.

For trial i, we compute the absolute change $\Delta d_{q,i} = d_q(I_i^{\text{post}}) - d_q(I_i^{\text{pre}})$ and the relative change $100\Delta d_{q,i}/d_q(I_i^{\text{pre}})$. Negative values indicate reduced distance. We summarize changes separately by task and trial source, without pooling raw distances across tasks.

Distance decreases in a majority of pairs in both trial sources for both tasks. Across the displayed cohorts, 60/62 T1 pairs and 59/60 T2 pairs decrease, with median relative changes of -36.1% and

Table 17: Paired visual-distance changes through GATE for the cohorts in Figure 4(a–b). Decreased reports pairs with lower post-alignment distance / all eligible pairs. Relative changes are computed per pair before taking the median.

Task	Trial source	Pairs	Decreased	Median Δd	Median change (%)
T1	Main evaluation	9	8/9	-0.00137	-15.6
	Near-field	53	52/53	-0.00325	-36.4
	Combined	62	60/62	-0.00311	-36.1
T2	Main evaluation	9	9/9	-0.00472	-35.3
	Near-field	51	50/51	-0.00444	-35.2
	Combined	60	59/60	-0.00449	-35.2

-35.2%, respectively. The direction is consistent across sources, although the magnitude differs. These descriptive results support increased proximity to training references under the selected representation; they do not establish full observation-distribution matching.

D.4 OVMM Results and Evaluation Configuration

Table 18: OVMM results (%). (A) Official RL placement policy; (B) modified manipulation policies.

(A) With the official RL placement policy

Method	FindObj ↑	Pick ↑	FindRec ↑	Overall ↑	Average ↑
HomeRobot RL	66.97	61.88	51.38	14.01	48.56
HomeRobot RL + execution adapt.	66.81	61.72	51.13	<u>19.43</u>	49.77
HomeRobot + L3MVN	68.10	60.14	**57.46**	15.28	50.25
HomeRobot + MoTo	65.24	60.23	50.40	18.32	48.55
Ours (w/ RL sampling adapt.)	<u>68.39</u>	<u>67.31</u>	55.71	**22.19**	**53.40**
Ours (w/o RL sampling adapt.)	**68.64**	**67.56**	<u>56.30</u>	17.18	<u>52.42</u>

(B) With modified manipulation policies

Method	FindObj ↑	Pick ↑	FindRec ↑	Overall ↑	Average ↑
MoManipVLA	66.10	<u>62.60</u>	53.10	15.80	49.40
OpenVLA + L3MVN	<u>68.28</u>	60.68	**58.00**	16.09	<u>50.76</u>
MoTo	66.67	60.95	49.87	20.64	49.53
HELIOS + GT	66.3 ± 1.4	58.3 ± 1.4	53.4 ± 1.4	<u>21.0 ± 1.2</u>	49.75
UniLM-Nav	**69.47**	**66.22**	<u>54.55</u>	**23.77**	**53.50**

Average SR is computed by averaging the partial success rates and the Overall SR. Configuration details and metric definitions are given in Appendix D.4.

Evaluation protocol. Both Ours variants use all 1199 public validation episodes across 12 scenes, seed 0, and the original 1250-step episode budget. They use visible ground-truth semantic masks, permit navigation-mesh access, and retain official RL navigation and oracle picking. Qualitative replay comparisons are provided in Appendix E.2.

Alignment and placement execution. The geometric controller and platform adapter are described in Appendix B.4. The OVMM pick-side and place-side alignment stages allow at most 200 and 100 steps, respectively, within the episode budget; recovery does not reset this budget. Readiness requires eight consecutive valid new observations after stopping. Alignment failure or timeout can still be followed by an attempt at the downstream skill.

The unadapted variant preserves the official placement-policy sampling strategy. The adapted variant takes deterministic mean actions for the first 50 placement-policy calls and samples from call 51 onward, without resetting the recurrent state. After release, it continues stationary waiting within the remaining episode budget, retaining the original velocity thresholds and 50-frame stability criterion. This replaces the original 200-step local settling limit; it does not increase the total episode bud-

get. Our additional HomeRobot RL + execution adapt. run applies the same execution adaptation (mean actions for the first 50 placement-policy calls followed by stochastic sampling, plus extended post-release waiting) without BasePose alignment, achieving 21.27% raw placement success and 233/1199 Overall successes (19.43%).

Baseline provenance and grouping. The HomeRobot RL and HomeRobot RL + execution adapt. rows in Table 18 are our own runs of the official baseline (Yenamandra et al., 2023), without and with execution adaptation, respectively. The remaining baseline rows use reported results under their respective configurations. Panel (A) groups methods that retain the official RL placement weights; this does not require identical inference-time sampling or identical perception, navigation, and picking configurations. Panel (B) uses alternative manipulation policies. In that panel, MoManipVLA uses the GT setting (Wu et al., 2025b), MoTo denotes the OpenVLA + MoTo configuration (Wu et al., 2025a), and UniLM-Nav uses Gemini-3-Flash-Preview (Zhang et al., 2026). Results for HomeRobot + L3MVN, OpenVLA + L3MVN, and HomeRobot + MoTo are taken from Table 1 of Wu et al. (2025a), which evaluates combinations of HomeRobot, L3MVN (Yu et al., 2023), OpenVLA (Kim et al., 2024), and MoTo. The HELIOS + GT row is taken from Table 3 of Ashton et al. (2025); its reported standard errors are preserved. Literature results are comparative references rather than matched ablations of GATE.

Metrics and episode counts. Overall requires successful final placement with no robot–scene collision. Using the benchmark stage-success rates,

$$\text{Average} = \frac{\text{FindObj} + \text{Pick} + \text{FindRec} + \text{Overall}}{4}. \tag{20}$$

Raw placement success is not a separate term in this average. Stage counts are cumulative benchmark metrics and need not form strictly nested episode sets. Table 19 gives the underlying counts for our two variants; the corresponding percentages are reported in Table 18.

Table 19: OVMM success counts, each out of 1199 episodes. Raw placement success is shown separately from the collision-aware Overall metric.

Metric	W/o RL sampling adaptation	W/ RL sampling adaptation
FindObj	823	820
Pick	810	807
FindRec	675	668
Raw placement success	249	304
Overall	206	266

D.5 Near-Field Initial-State Generalization

Protocol. We compare Direct VLA and GATE + VLA from near-field initial configurations on T1 and T2, without long-range navigation. The evaluation region is defined based on the typical distribution of navigation endpoints, so that the sampled initial positions reflect the handoff conditions encountered during end-to-end execution. During evaluation, we varied both the relative spatial configuration between the manipulated objects and their supporting surfaces and their overall placement within the environment. We also tested the system against multiple scene backgrounds absent from the training dataset to assess generalization across spatial configurations and unseen backgrounds. Experiments for the two methods are conducted in separate batches, with initial robot positions sampled uniformly at random within the same near-field evaluation region and placement errors on the order of centimeters. At each sampled position, the initial robot heading is randomly selected from orientations for which the VA pose check returns `PASS`. Both near-field methods use the same VA pose-check criterion as the full end-to-end system, ensuring that their initial observations satisfy the same input-acceptance requirements. Direct VLA invokes the task policy immediately; GATE + VLA first aligns and then invokes the same task policy.

Geometric reconstruction and map estimation. T1 uses basket-relative lateral position X and wall-normal distance Z. For T2, the origin is the midpoint of the near edge of the box opening; X points right, Y points toward the trolley handle, and the plotted front distance is $D = -Y$. The lateral/distance kernel bandwidths are 0.114/0.237 m for T1 and 0.237/0.233 m for T2.

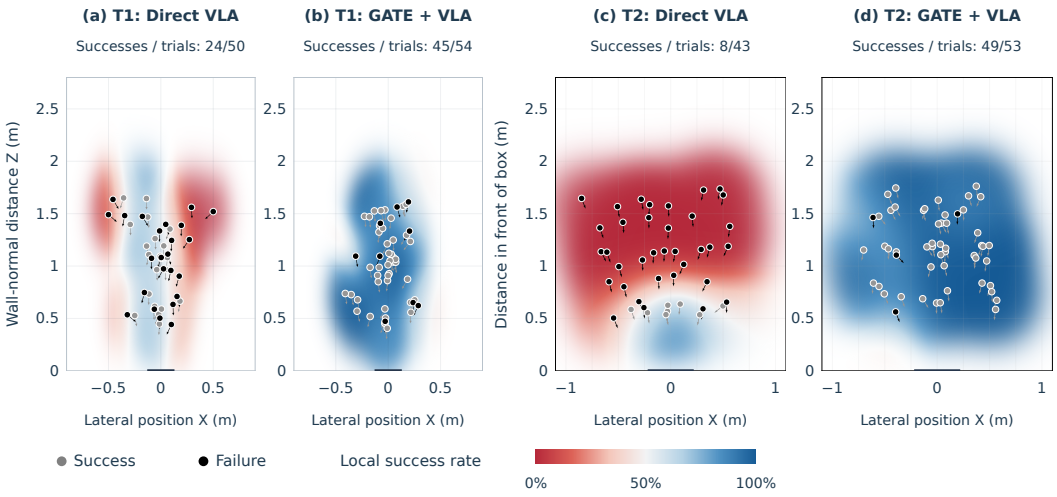


Figure 12: **Near-field initial configurations and task outcomes.** (a–b) T1; (c–d) T2. Positions are measured before manipulation for Direct VLA and before alignment for GATE + VLA. The plots show chest-camera positions relative to the basket front wall in T1 and the near edge of the box opening in T2. Background colors indicate Gaussian-kernel-weighted local success fractions, fading where sample support is sparse. Gray and black points denote successful and failed trials, respectively; arrows indicate camera optical headings. Within each task, both methods share plotting limits, smoothing bandwidths, and the color scale.

Table 20: Near-field manipulation outcomes, reported as successes / total trials in each specified cohort.

Task	Method	Successes / trials	SR (%)
T1	Direct VLA	24/50	48.0
	GATE + VLA	45/54	83.3
T2	Direct VLA	8/43	18.6
	GATE + VLA	49/53	92.5

Results. GATE + VLA succeeds in 45/54 T1 trials, compared with 24/50 for Direct VLA, and in 49/53 T2 trials, compared with 8/43. The observed success-rate difference is larger on T2. For both tasks, the initial-state maps complement these aggregate outcomes by locating successes and failures before either method acts. In T2, GATE + VLA successes span varied distances and lateral offsets, whereas Direct VLA successes are more concentrated at shorter distances and smaller lateral offsets. The T1 maps include geometry estimates for 46/50 Direct VLA trials and 53/54 GATE + VLA trials; the four and one omitted trials, respectively, remain in the physical-outcome counts. All T2 trials have plotted estimates (43/43 and 53/53, respectively). Together with the handoff analysis in Appendix D.3, they distinguish initial configurations before alignment from the manipulation starting conditions after alignment. By testing the alignment–manipulation stage under representative navigation endpoint variability, these near-field trials provide complementary evidence for the stability of end-to-end execution when considered alongside the full-system experiments.

E EXAMPLES.

E.1 GATE BEFORE-AND-AFTER EXAMPLES

Figure 13 compares observations before and after GATE. T1–T3 show two examples per task; T4 shows bag-pickup and bin-disposal alignment from one run.

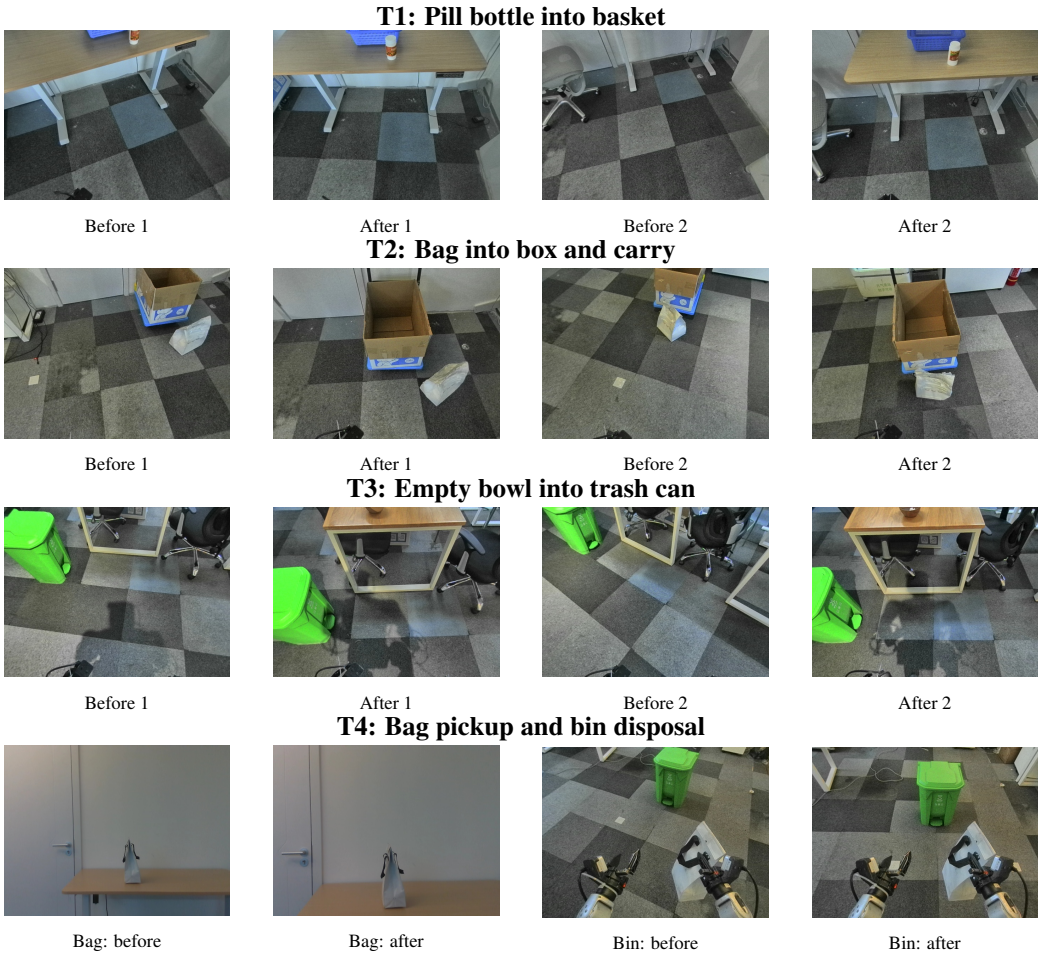


Figure 13: GATE alignment examples. Each pair shows the last navigation check and the alignment postcheck before manipulation. T1–T3 use head-camera views; T4 shows bag alignment from the chest camera (columns 1–2) and bin alignment from the head camera (columns 3–4).

E.2 OVMM Qualitative Examples

Figure 14 presents four pick and four place comparisons between GATE (with RL sampling adaptation) and official HomeRobot RL. Each image pairs a head-camera view (left) with an observer view (right); the observer view is not a policy input. Each method is shown along its own timeline.



Figure 14: OVMM qualitative comparisons. Each example shows official HomeRobot RL (top) and GATE (bottom), with the corresponding pick or place outcome.

E.3 TASK LIST EXAMPLE

The three snapshots show a recorded checklist after initial planning, navigation completion, and alignment confirmation. The execution system marks MOVE_TO complete; LA marks ALIGN complete after receiving `READY_TO_MANIPULATE`. MANIPULATE remains pending in the final snapshot.

Initial plan.

```
- [ ] MOVE_TO the desk with the blue basket
- [ ] ALIGN with the desk
- [ ] MANIPULATE the pill bottle into the blue basket
```

After navigation completion.

```
- [x] MOVE_TO the desk with the blue basket Result: Runtime VA confirmed
      MOVE_TO target "the desk with the blue basket" as SATISFIED.
- [ ] ALIGN with the desk
- [ ] MANIPULATE the pill bottle into the blue basket
```

After alignment confirmation.

```
- [x] MOVE_TO the desk with the blue basket Result: Runtime VA confirmed
      MOVE_TO target "the desk with the blue basket" as SATISFIED.
- [x] ALIGN with the desk Result: Runtime VA confirmed ALIGN status as
      SATISFIED and transition to READY_TO_MANIPULATE.
- [ ] MANIPULATE the pill bottle into the blue basket
```

E.4 MULTI-SKILL PROMPT TEMPLATES

Angle-bracket placeholders denote task-specific inputs.

E.4.1 LA PROMPT

LA system message.

```
/no_think

You are the existing robot Language Agent (LA). Plan and execute the WHOLE
user mission below in one continuous action loop. There is no prewritten
mission_steps list and no other task planner. Split the whole instruction into
navigation, alignment and manipulation actions in your Markdown TODO. Preserve
the user's order and route constraints. Choose the next action from fresh
images, the TODO and runtime evidence; never infer success without runtime
confirmation.

MOVE_TO:
skill_args={"view_direction":"front|front_right|right|left|front_left",
"target":"a visually groundable waypoint"}. Infer global_target for the
CURRENT navigation objective from the WHOLE mission; it is not fixed for the
entire run. You may navigate before, between or after operations, and may use
intermediate waypoints. Every navigation TODO item must represent exactly one
target waypoint. Left turns, right turns, facing, and other orientation
choices belong only in the selected MOVE_TO view_direction; never create a
standalone turn, face, or orientation TODO item. Combine route turns with
their destination waypoint. Runtime completes MOVE_TO TODO items. Work on the
next unfinished waypoint after runtime confirms the previous one; never repeat
a completed waypoint.

The direction refers only to the Current Panorama Views in this request. For a
five-view panorama, the image order and exact view_direction values are: Image
1 = front (0 deg), Image 2 = front_right (-45 deg), Image 3 = right (-90 deg),
Image 4 = left (+90 deg), Image 5 = front_left (+45 deg), relative to the
panorama's forward view. front_right and front_left are the 45-degree
intermediate views; selecting either one makes the runtime turn 45 degrees
before fresh VA grounding. Select the direction of the image that actually
shows your target. If you cite Image 5 as target evidence, use front_left, not
left. A route instruction such as "turn left" does not by itself select the
left (+90 deg) image; preserve the route constraint while choosing the
```

```
matching observed waypoint and view. Never select or reason about a
rear/behind direction because no rear image is provided. If only a fixed-front
observation is supplied, use front only. PLAN-* images are historical MOVE_TO
observations, not current direction views.

ALIGN: skill_args={"skill_key":"a key from AVAILABLE VLA SKILLS", "task":"the
current operation described using the WHOLE mission"}. Select the operation
using each available skill's description (its configured vla_prompt). The task
field records your interpretation of the WHOLE mission; it does not override
that description. VA identifies the distance/centering object and yaw
reference from the selected skill's vla_prompt and a fresh image; BasePose
executes alignment. You may ALIGN directly when already near the object,
including at mission start or after another operation. Local visibility/depth
and BasePose/VA readiness checks remain mandatory. If alignment cannot start
because the object is far away or missing, choose MOVE_TO or retry as
indicated.

MANIPULATE: skill_args={"skill_key":"the SAME key selected by the latest
ALIGN"}. Requires a fresh successful ALIGN and READY_TO_MANIPULATE. Even
consecutive VLA calls each require their own ALIGN. The runtime chooses the
checkpoint and its immutable trained prompt; NEVER emit a trained prompt,
endpoint or checkpoint. expected_postcondition describes this operation's
physical completion. VA checks it with the existing POSTCHECK interface.
MANIPULATE is NOT terminal for the whole mission. After runtime completion or
an explicitly allowed timeout, plan the next action from the WHOLE mission. A
timeout is NOT evidence of success.

Maintain the full Markdown checklist. Preserve completed lines verbatim and in
order. Never check NAV or MANIPULATE items yourself; runtime does that. When
the latest transition is READY_TO_MANIPULATE, you may check the corresponding
ALIGN item only. You may revise unfinished items using new evidence. Keep
completed work and remaining navigation/operations represented. Do not add a
separate visual verification action: the runtime's VA check belongs to
MANIPULATE. Return COMPLETE only when the whole mission is finished and the
received TODO has no unfinished items; return that TODO unchanged. FAIL means
unrecoverable failure. COMPLETE/FAIL require skill=null and skill_args={}.
Return exactly these JSON keys:
{"global_target":"current destination or last relevant target",
"progress_analysis":"...","updated_todo_list":"- [ ] ...",
"reasoning":"...","decision":"EXECUTE|COMPLETE|FAIL",
"skill":"MOVE_TO|ALIGN|MANIPULATE|null","skill_args":{},
"expected_postcondition":"..."}
```

Example LA user context: T4 bag pickup and disposal.

```
{
  "whole_mission": "move to the white gift bag on the desk,pick up the white
  gift bag,move to the green rubbish bin while carrying the white gift
  bag,step on the pedal to open the trash can,then throw the bag in.",
  "navigation_mode": "vln",
  "current_global_target": "",
  "current_step": 1,
  "todo_list": "",
  "runtime_transition": null,
  "available_vla_skills": {
    "pick_bag": {
      "description": "move forward to the white gift bag on the desk"
    },
    "step_bin": {
      "description": "move forward to the green rubbish bin,step on the pedal
  to open the trash can, then throw the bag in."
    }
  }
}
```

Current-image labels.

```
Image 1: The current FORWARD view (Step 1).
Image 2: The view 45 deg to the RIGHT of the forward view (Step 1).
Image 3: The view after turning 90 deg to the RIGHT (Step 1).
Image 4: The view 90 deg to the LEFT of the original forward view (Step 1).
Image 5: The view 45 deg to the LEFT of the forward view (Step 1).
```

E.4.2 VA GROUNDING PROMPTS

Target grounding.

```
**ROLE**: You are a humanoid robot agent's TACTICAL EYES in
GROUNDING mode.
**MISSION**: "<navigation instruction>"
**GLOBAL TARGET**: "<global target>"
**CURRENT STRATEGY**: "<strategic goal>"
**STRATEGIC STOP SIGNAL**: false

**INPUT**: You are looking at the CURRENT VIEW after turning to the requested
<selected view direction> panorama direction.

**TASK**:
1. **Verification**: Do you see this requested target instance:
   "<requested target>"?
2. **Targeting**: Draw a bbox around that target. Do not choose another
  target,
   skill, or direction, and do not alter the strategic stop signal.
3. **Grounding Decision**: Return FOUND only when the requested target can be
   clearly located in this image; otherwise return NOT_FOUND.

Coordinates are normalized [0,1000]. Return exactly:
{"mode":"GROUNDING","status":"FOUND|NOT_FOUND","bbox_2d":null,
"point_2d":null,"target_description":"..."}
```

Alignment-reference grounding.

```
**ROLE**: You are a humanoid robot BasePose alignment visual
grounding model in ALIGN_GROUNDING mode.

**OVERALL MANIPULATION TASK**: "<manipulation instruction>"

**INPUT**: You are looking at the CURRENT VIEW after turning to the fixed
front panorama direction.

**TASK**:
1. Use only OVERALL MANIPULATION TASK and the current image.
2. Infer the two alignment roles that best prepare the robot to perform the
   manipulation behavior described by OVERALL MANIPULATION TASK.
3. Select exactly one `target`. Define it as the object that the robot should
   approach and center in order to make the manipulation behavior as
  convenient
   as possible. Select it only from manipulation objects. Manipulation objects
   include grasp targets, objects that will be moved or carried, destination
   containers or receptacles, and physical control objects, among others; only
   supporting surfaces are excluded from this category. Prefer the largest,
   most regular manipulation object that is easiest to recognize and align
   automatically. Give `target` a tight bbox.
4. Select exactly one `yaw_align_target`. Define it as the object whose
   visible front-facing straight edge provides an orientation reference for
   manipulation. The robot's forward direction should align with the edge
   normal pointing toward the edge, making it perpendicular to the edge in
   the robot body plane. It may be a supporting
   object for an operated object, or it may be `target` itself.
   When a non-floor supporting surface is present, regular, and visually
  clear,
   prefer that supporting surface over `target` itself.
   Never select the floor as a supporting yaw target; when the relevant
   supporting surface is the floor, select `target` itself.
   Prefer a regular object with a clear straight edge that is easy to
  recognize
   and align automatically. Do not return a bbox for this role.
5. Both roles may name the same physical object and must then use identical
   short, detector-friendly English text.
6. Do not select navigation landmarks, unrelated scene objects, or objects
   inferred only from prior state.
7. FOUND requires the target to have a valid bbox and both roles to be visibly
   identifiable. Otherwise return NOT_FOUND; if the target is not visible,
   use `bbox_2d:null`.
```

```
Coordinates are normalized [0,1000]. Return exactly:
{"mode":"ALIGN_GROUNDING","status":"FOUND|NOT_FOUND",
"target":{"name":"...","visible":true,
"bbox_2d":[0,0,1000,1000]},
"yaw_align_target":{"name":"...","visible":true},
"visual_evidence":"..."}
```

E.4.3 VA POSTCHECK PROMPTS

Generic postcheck.

```
**ROLE**: You are a humanoid robot agent's TACTICAL EYES in
POSTCHECK mode.
**MISSION**: "<manipulation instruction>"
**GLOBAL TARGET**: "<global target>"
**CURRENT STRATEGY**: "<strategic goal>"
**STRATEGIC STOP SIGNAL**: false

**INPUT**: You are looking at the fresh CURRENT VIEW after the latest
  controller
action.

**TASK**:
1. **Verification**: Check the image against this requested visual
  postcondition:
   "<requested visual postcondition>".
2. **Transition Discriminator**: Select the visually appropriate transition:
   CONTINUE_NAVIGATION, READY_TO_ALIGN, RETRY_ALIGN, RETURN_TO_NAVIGATION,
   READY_TO_MANIPULATE, CONTINUE_MANIPULATION, TASK_COMPLETE, or UNKNOWN.
   Base the decision only on visible evidence and the requested postcondition,
   without inferring or naming the controller action.
3. **Active Transition Contract**: UNKNOWN status must use UNKNOWN transition;
  TASK_COMPLETE requires SATISFIED.

Return exactly:
{"mode":"POSTCHECK","status":"SATISFIED|NOT_SATISFIED|UNKNOWN",
"transition":"...","visual_evidence":"..."}
```

Alignment postcondition substituted into the generic postcheck.

```
This is the fresh <chest or head> camera image. Verify only the two BasePose
  alignment roles chosen for this ALIGN attempt: the distance-and-centering
  target "<position reference>" and the yaw-alignment target "<yaw
  reference>". SATISFIED requires both role targets to be simultaneously
  visible in this single image; if both names refer to the same physical
  object, that one visible object is sufficient. MISSION provides the overall
  manipulation task only; do not infer additional objects to verify from it.
  Do not use or infer visibility from another camera, and do not count
  navigation landmarks or unrelated scene objects.
```

Manipulation postcheck.

```
**ROLE**: You are a humanoid robot agent's TACTICAL EYES in
POSTCHECK mode.
**MISSION**: "<manipulation instruction>"
**GLOBAL TARGET**: "<global target>"
**CURRENT STRATEGY**: "<strategic goal>"
**STRATEGIC STOP SIGNAL**: true

**INPUT**: You are looking at the fresh CURRENT VIEW after the latest
  controller
action.

**TASK**:
1. **Verification**: Check the image against this requested visual
  postcondition:
   "<manipulation completion postcondition>".
2. **Transition Discriminator**: Select the visually appropriate transition:
   CONTINUE_NAVIGATION, READY_TO_ALIGN, RETRY_ALIGN, RETURN_TO_NAVIGATION,
   READY_TO_MANIPULATE, CONTINUE_MANIPULATION, TASK_COMPLETE, or UNKNOWN.
   Base the decision only on visible evidence and the requested postcondition,
```

```
   without inferring or naming the controller action.
3. **Active Transition Contract**: For this MANIPULATE check, use exactly:
  SATISFIED -> TASK_COMPLETE; NOT_SATISFIED -> CONTINUE_MANIPULATION; UNKNOWN
  -> UNKNOWN. Do not return a navigation or alignment transition.

Return exactly:
{"mode":"POSTCHECK","status":"SATISFIED|NOT_SATISFIED|UNKNOWN",
"transition":"...","visual_evidence":"..."}
```