NAV4MANIP: FROM NAVIGATION TO DEXTEROUS MOBILE MANIPULATION THROUGH CO-EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Dexterous mobile manipulation is essential for robots to perform useful tasks in complex, unstructured environments. However, existing systems face two challenges. First, navigation remains too coarse for downstream manipulation, which often requires a precise base position and approach orientation rather than merely reaching a meter-scale goal region. Second, navigation and manipulation are commonly trained in isolation, lacking an interactive feedback loop that enables the two capabilities to co-evolve. These limitations prevent current systems from reliably solving practical, long-horizon dexterous mobile-manipulation tasks. To address them, we introduce NAV4MANIP, a bidirectionally coupled co-evolution system for long-horizon dexterous mobile manipulation. It comprises two complementary architectures: NAV4MANIP-N learns long-horizon navigation and manipulation-aware stopping, while NAV4MANIP-M post-trains a world-action model (WAM) to provide the downstream dexterous-manipulation interface. We train NAV4MANIP-N in three stages: multi-task co-tuning learns navigation behavior and manipulation-aware affordance prediction; rejection sampling fine-tuning (RFT) rejects low-quality policy-generated paths and fine-tunes the model on retained manipulation-successful trajectories; and reinforcement learning from AI feedback (RLAIF) uses execution-grounded counterfactual preferences to further improve navigation decisions. To support this framework, we establish the NAV-N-MANIP benchmark and construct Nav4Manip-Pairs from benchmark rollouts, providing large-scale paired navigation–manipulation trajectories and rollout-derived affordance supervision. Extensive experiments on established vision-language navigation (VLN) benchmarks and NAV-N-MANIP demonstrate state-of-the-art performance, while real-world evaluations show that NAV4MANIP completes diverse dexterous mobile-manipulation tasks. Project website: `https://nav4manip-anonymous.github.io/Nav4Manip`.

1 INTRODUCTION

Dexterous mobile manipulation integrates navigation and manipulation: navigation brings the robot within reach of a target workspace, while manipulation enables dexterous physical interaction with the environment. Recent VLN policies map egocentric video and language instructions to navigation actions or waypoints (Zhang et al., 2024b;a; 2026a; Xue et al., 2026), whereas manipulation policies predominantly learn tabletop skills such as reaching, grasping, and object relocation (Brohan et al., 2023; Chi et al., 2023; Team et al., 2024; Ye et al., 2026; Yuan et al., 2026).

However, existing VLN methods are insufficient as upstream policies for downstream manipulation. First, manipulation policies have limited tolerance to variations in the robot's base pose: workspace geometry, kinematic reachability, visibility, and collision constraints make success highly sensitive to the terminal position and orientation. By contrast, VLN benchmarks typically declare success anywhere within 3 m of a reference goal (Anderson et al., 2018; Krantz et al., 2020). An endpoint that satisfies VLN metrics may therefore leave the target unreachable, occluded, or visible only from an unsuitable approach direction. Second, existing mobile-manipulation systems commonly model locomotion and manipulation either with a single whole-body policy over room-scale task environments (Fu et al., 2025; Black et al., 2025) or as two separately trained, open-loop modules (Yenamandra et al., 2023; Liu et al., 2024). The former does not directly address long-horizon, language-guided navigation, while the latter prevents manipulation outcomes from informing and

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.

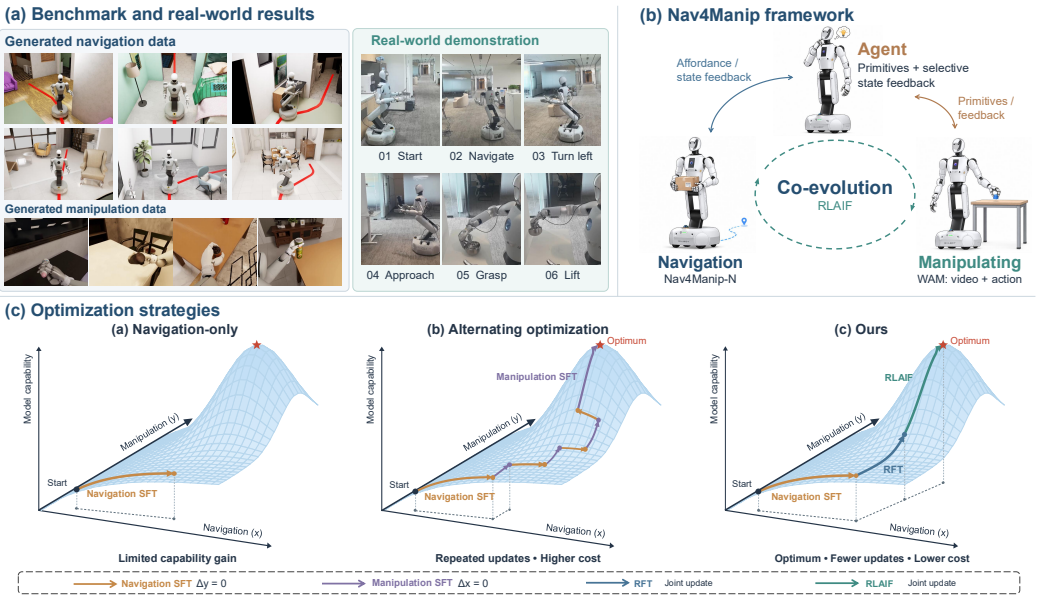

Figure 1: **Overview of NAV4MANIP and its distinction from prior mobile-manipulation paradigms.** (a) The NAV-N-MANIP benchmark evaluates long-horizon mobile manipulation, and its rollouts produce Nav4Manip-Pairs, which contains paired navigation–manipulation trajectories with rollout-derived affordance supervision. See Appendix E for more results. (b) NAV4MANIP enables navigation and manipulation to co-evolve by coupling NAV4MANIP-N with the modular downstream manipulation model NAV4MANIP-M. Manipulation outcomes provide feedback for improving the navigator. (c) Existing paradigms optimize this interface only partially: navigation-only supervised fine-tuning (SFT) does not optimize manipulation-valid handoffs, whereas alternating optimization repeatedly collects manipulation data at changing navigation endpoints. Demonstrations collected at one endpoint do not readily transfer to another because base pose and robot–object geometry change. In contrast, our co-evolution strategy combines multi-task co-tuning, verifier-guided RFT, and RLAIF from execution-grounded counterfactual preferences to improve end-to-end capability. Appendix A provides a more detailed comparison of the three paradigms.

correcting the navigator. Moreover, these systems focus on gripper-based manipulation rather than dexterous multi-finger interaction. More broadly, separately applying supervised fine-tuning to navigation and manipulation data provides no mechanism for the two capabilities to co-evolve toward a shared end-to-end objective.

We address these limitations by introducing NAV4MANIP, a bidirectionally coupled co-evolution system with two complementary architectures: NAV4MANIP-N learns long-horizon navigation and manipulation-aware stopping, while NAV4MANIP-M post-trains a WAM to provide the downstream dexterous-manipulation interface. As illustrated in Figure 1, navigation-only optimization yields limited joint capability, while alternating optimization repeatedly trains both modules. Our staged co-evolution strategy improves end-to-end performance while avoiding repeated manipulation-policy training, achieving higher task success with fewer module-training stages under matched per-stage sample counts (Table 3). NAV4MANIP-N jointly predicts navigation actions and manipulation-aware endpoint affordances. Multi-task co-tuning first establishes broad navigation competence and manipulation-aware handoff prediction. Verifier-guided RFT then exposes NAV4MANIP-N to trajectories induced by its own policy and retains only paths whose terminal poses pass navigation checks and repeated downstream manipulation verification. Finally, RLAIF uses an execution agent to interpret navigation–manipulation outcomes and construct evidence-supported counterfactual preference pairs for further optimizing NAV4MANIP-N. The resulting knowledge is distilled into the navigation model, enabling manipulation feedback to improve future navigation.

To support training and evaluation, we establish NAV-N-MANIP, a benchmark that evaluates both long-horizon instruction-following navigation and supports executing downstream manipulation. Rollouts on NAV-N-MANIP yield Nav4Manip-Pairs, a 233-hour paired navigation–manipulation

dataset. For each candidate endpoint, Nav4Manip-Pairs associates a collision-free navigation path with repeated manipulation outcomes and an empirical affordance score, providing supervision for navigation actions, endpoint prediction, ranking, and manipulation-valid handoffs. In summary, we make the following contributions:

- We introduce NAV4MANIP, a dexterous mobile-manipulation co-evolution system comprising three-stage NAV4MANIP-N and WAM-post-training NAV4MANIP-M.
- We establish the NAV-N-MANIP benchmark for long-horizon dexterous mobile manipulation and construct Nav4Manip-Pairs, a 233-hour dataset of paired navigation–manipulation trajectories, candidate endpoints, and rollout-derived affordance scores.
- We conduct extensive experiments on established VLN tasks and NAV-N-MANIP, together with real-robot evaluations, demonstrating strong navigation performance and reliable end-to-end dexterous mobile manipulation across diverse tasks.

2 RELATED WORK

VISION-LANGUAGE NAVIGATION DATASETS. Built on Matterport3D (Chang et al., 2017), R2R, R4R, and RxR provide increasingly long routes and multilingual grounding (Anderson et al., 2018; Jain et al., 2019; Ku et al., 2020). VLN-CE enables continuous navigation (Krantz et al., 2020), while REVERIE and OpenUAV extend evaluation to object grounding and aerial navigation (Qi et al., 2020; Wang et al., 2025b). ScaleVLN scales synthetic supervision (Wang et al., 2023), and RoomTour3D and Sekai leverage exploration videos (Han et al., 2025; Li et al., 2026b). VLN-PE, VLNVerse, and InternData-N1 further broaden physical realism, embodiment coverage, and scene diversity (Wang et al., 2025a; Lin et al., 2025; Wei et al., 2026b). Although these efforts substantially expand navigation supervision and some consider downstream embodied tasks, most VLN benchmarks do not explicitly evaluate whether terminal poses support manipulation. Nav4Manip-Pairs addresses this gap by pairing navigation paths and candidate endpoints with manipulation trajectories and rollout-derived success rates.

VISION-LANGUAGE NAVIGATION ALGORITHMS. VLN methods have progressed from recurrent policies and data augmentation (Anderson et al., 2018; Fried et al., 2018; Tan et al., 2019) to navigation pre-training and history modeling (Hao et al., 2020; Hong et al., 2020; Guhur et al., 2021; Chen et al., 2021), followed by explicit spatial memory and planning (Chen et al., 2022; An et al., 2023; 2024). Recent foundation-model approaches extend these capabilities along several complementary directions. For reasoning and agentic planning, NavGPT, MapGPT, and InstructNav prompt large models to interpret instructions and construct navigation plans (Zhou et al., 2024; Chen et al., 2024; Long et al., 2024), while JanusVLN improves long-horizon reasoning through compact semantic and spatial memories (Zeng et al., 2026). For continuous and embodied control, NaVid and StreamVLN operate over streaming observations, NaVILA connects navigation to legged locomotion, and DualVLN separates high-level grounding from real-time local control (Zhang et al., 2024b; Wei et al., 2026c; Cheng et al., 2025; Wei et al., 2026a). For multi-task and cross-embodiment generalization, NaviLLM, Uni-NaVid, and NavFoM combine navigation with broader embodied supervision, while OmniVLA, ABot-N0/N1, InternVLA-N1, and Qwen-RobotNav further expand task and embodiment coverage (Zheng et al., 2024; Zhang et al., 2024a; 2026a; Hirose et al., 2026; Chu et al., 2026; Gong et al., 2026; Wei et al., 2026b; Zhang et al., 2026b). Complementing these efforts, NAV4MANIP-N learns a manipulation-valid position–yaw acceptance region and refines its approach using execution feedback.

MOBILE MANIPULATION. HomeRobot and OK-Robot integrate navigation, grasping, and placement for open-world household tasks (Yenamandra et al., 2023; Liu et al., 2024), while Mobile ALOHA and $\pi_{0.5}$ learn whole-body policies for longer-horizon mobile manipulation (Fu et al., 2025; Black et al., 2025). Humanoid systems such as EgoHumanoid, WholeBodyVLA, Ψ_0, and ω-0 further unify locomotion and manipulation control (Shi et al., 2026; Jiang et al., 2026; Wei et al., 2026d; Li et al., 2026a). Focusing on the navigation–manipulation interface, MoMa-Kitchen and BaSeNet investigate manipulation-aware base placement (Zhang et al., 2025; Naik et al., 2024). However, they do not address how long-horizon, language-conditioned navigation and dexterous manipulation can be coupled through execution feedback. NAV4MANIP addresses this gap through staged co-evolution:

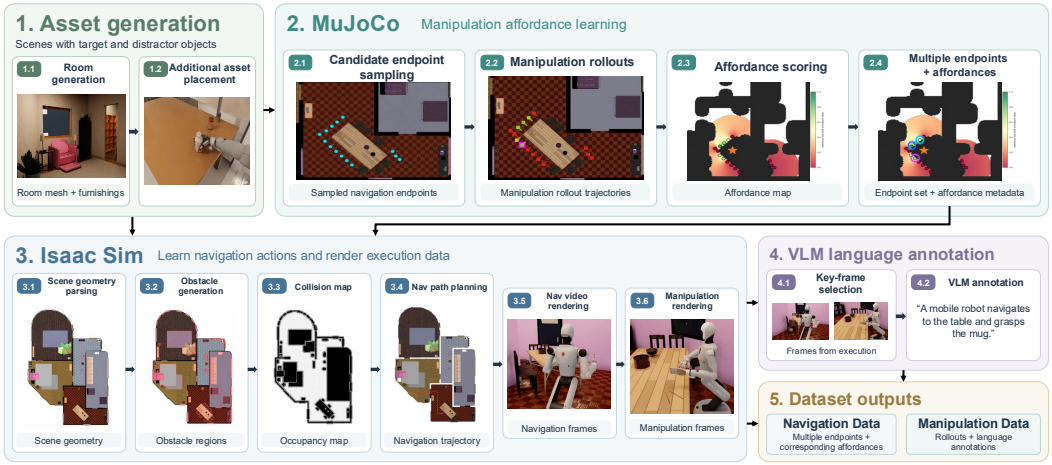

Figure 2: **Nav4Manip-Pairs data-generation pipeline.** The pipeline constructs paired navigation trajectories, manipulation-validated endpoints, and aligned vision–language annotations.

paired navigation–manipulation training aligns the two modules, while downstream execution outcomes progressively refine navigation and stopping, improving the handoff from navigation over long distances to successful dexterous manipulation.

3 NAV4MANIP

3.1 NAV4MANIP-PAIRS CONSTRUCTION AND REPRESENTATION

Figure 2 summarizes our automated data-generation pipeline. We combine 221 EmbodiedGen v2 houses (Wang et al., 2026b) with 176 scenes from the VLNVerse training set (Lin et al., 2025), insert task objects, and standardize the scene configurations across MuJoCo and Isaac Sim. In MuJoCo, candidate base poses are proposed from the target-object location and surrounding support surface, then evaluated through repeated manipulation rollouts with object-pose perturbations to estimate endpoint affordances. Each selected navigation-reachable endpoint is paired with a collision-validated navigation path. The aligned navigation and manipulation trajectories are replayed in Isaac Sim to render synchronized observations, after which a VLM generates task instructions and vision–language annotations. The resulting records connect navigation trajectories with downstream manipulation outcomes. Full generation and validation details are provided in Appendix D.

For each task, candidate endpoints around the support surface are evaluated through repeated manipulation rollouts with object-pose perturbations. We represent a paired Nav4Manip-Pairs instance compactly as

$$\begin{aligned} \mathcal{D}_i &= (l_i, \mathcal{N}_i, \mathcal{M}_i)\,, \\ \mathcal{M}_i &= \left(q_i, \{\tau_{i,k}^m\}_{k=1}^{K_i}\right), \end{aligned} \tag{1}$$

where l_i is the task instruction, $\mathcal{N}_i$ collects all navigation-related fields, and $\mathcal{M}_i$ is the corresponding manipulation record. The latter contains the empirical endpoint-affordance score q_i and, for each selected feasible endpoint, the K_i successful manipulation rollout traces $\{\tau_{i,k}^m\}_{k=1}^{K_i}$ collected at that endpoint. The navigation record includes the initial and terminal base configurations, collision-validated path, rendered observations, and action labels. We retain both successful and unsuccessful candidates as discrete endpoint proposals for affordance prediction and ranking, while selected feasible candidates are paired with validated navigation trajectories. The exact contents of $\mathcal{N}_i$ and $\mathcal{M}_i$, the rollout outcome labels, the definition of q_i, and the complete sampling, physical-validation, and pairing procedures are provided in Appendix D.

We use both parts of each paired record for training: the navigation record $\mathcal{N}_i$ and the endpoint-affordance score q_i in $\mathcal{M}_i$ jointly supervise navigation and endpoint affordances, while the successful manipulation traces in $\mathcal{M}_i$, their aligned observations, and task instructions supervise NAV4MANIP-M. After this paired-data training, NAV4MANIP-M is held fixed while its execution outcomes guide

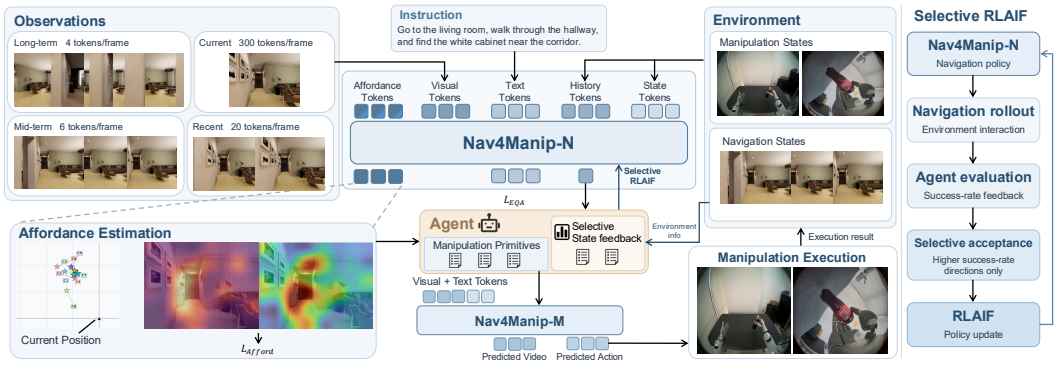

Figure 3: **Overview of NAV4MANIP.** NAV4MANIP-N jointly predicts navigation actions and manipulation-aware endpoint affordances. The bird's-eye-view (BEV) map shows the robot and endpoint–yaw candidates in ground-plane coordinates. The two affordance heatmaps visualize rollout success over endpoint–yaw configurations and discrete object-position perturbations, respectively, indicating where the robot should stop and how robust the endpoint is to target displacement. Co-tuning, verifier-guided RFT, and counterfactual RLAIF progressively improve the policy, whose terminal pose is handed directly to NAV4MANIP-M at inference.

navigation refinement. This staged coupling constitutes our co-evolution procedure; Appendix G gives the executor details.

3.2 MANIPULATION-AWARE NAVIGATION PIPELINE

As shown in Figure 3, at navigation step t, NAV4MANIP-N receives learnable affordance queries, multi-scale visual tokens, instruction tokens, task-relevant history, and the current robot and environment state. The history summarizes temporally sampled observations and actions and, during counterfactual RLAIF data collection, may additionally contain selected navigation–manipulation feedback. We use h_t as compact notation for this complete policy context and let a_t^n denote the selected navigation action. With a multimodal backbone initialized from Qwen3-VL (Bai et al., 2025), the NAV4MANIP-N navigation policy and the NAV4MANIP-M manipulation policy are written as

$$a_t^n \sim \pi_\theta(\cdot \mid h_t), \qquad a_t^m \sim \pi_m(\cdot \mid h_t, p_{agent}). \tag{2}$$

Here, π_θ denotes the navigation-action distribution used consistently in co-tuning, RFT, and RLAIF. The NAV4MANIP-M policy π_m predicts the manipulation action a_t^m conditioned on the policy context h_t and the manipulation primitive p_{agent} selected by an agent. The navigation model also predicts grounded EQA answers, a dense ego-centered affordance map, and image-grounded endpoint proposals, which guide the robot toward manipulation-ready endpoints. Appendix B.1, Appendix B.2, and Appendix G.2 provide more implementation details.

3.3 MULTI-TASK CO-TUNING

Table 1 summarizes the composition of our multi-task co-tuning mixture. Our multi-task co-tuning mixture contains 4.9M samples (625.08 hours) from RxR (Ku et al., 2020), HM3D ObjectNav (Yadav et al., 2023), R2R (Anderson et al., 2018), our rendered EQA data, and Nav4Manip-Pairs. Public navigation datasets provide broad action supervision, the rendered EQA data improve grounded visual reasoning, and Nav4Manip-Pairs jointly supervise navigation behavior and manipulation-valid handoffs.

We optimize the co-tuning objective

$$\mathcal{L}_{\text{co}} = \mathcal{L}_{\text{nav}} + \lambda_{\text{Afford}}\mathcal{L}_{\text{Afford}} + \lambda_{\text{EQA}}\mathcal{L}_{\text{EQA}}, \tag{3}$$

where $\mathcal{L}_{\text{nav}}$ is the token-level navigation-action loss, $\mathcal{L}_{\text{Afford}}$ supervises the dense heatmap and discrete endpoint predictions with rollout-derived outcomes, and $\mathcal{L}_{\text{EQA}}$ trains grounded answer generation. The affordance objective includes dense-map prediction, endpoint regression and scoring, ranking, and consistency between the BEV and image-grounded predictions. Joint co-tuning preserves

Table 1: Composition of the multi-task co-tuning mixture. For Nav4Manip-Pairs, "action + affordance" denotes joint supervision of navigation behavior and manipulation-valid endpoints. Policy-generated RFT records are collected separately after co-tuning.

Source	Duration (h)	Samples	Episodes	Supervision
RxR	174.67	1.9M	6,668	action tokens
HM3D ObjectNav	115.66	416K	6,603	action tokens
R2R	58.39	631K	3,603	action tokens
Rendered EQA (ours)	43.26	156K	8,976	answer text
Nav4Manip-Pairs (ours)	233.11	1.8M	17,927	action + affordance
Total	625.08	4.9M	43,777	

broad navigation competence while initializing the policy to reason about downstream manipulation feasibility.

3.4 Verifier-Guided Rejection Sampling Fine-Tuning

RFT resamples navigation continuations from states induced by Nav4Manip-N's own decisions and retries manipulation, using the co-tuned prior to improve manipulation-valid handoffs. For each task initialization $\mathcal{E}_i$, we execute $\pi_{\theta_{\text{co}}}$ in closed loop and sample intermediate states along the resulting trajectory. From each selected state, Nav4Manip-N rolls out a new continuation toward a candidate endpoint. The original prefix and resampled continuation form the RFT path P_i^{RFT}, which terminates at b_i^{RFT}. Repeated manipulation trials at this endpoint yield its empirical affordance q_i^{RFT}. The resulting record $\mathcal{D}_i^{\text{RFT}}$ follows the same format as $\mathcal{D}_i$, with trajectory-dependent fields regenerated by Nav4Manip-N.

The verifier discards unreachable endpoints, collision-invalid paths, failed handoffs, and samples below the manipulation-success threshold. The remaining samples are ranked by

$$s_i^{\text{RFT}} = q_i^{\text{RFT}} - \lambda_{\text{path}} \ell(P_i^{\text{RFT}}), \tag{4}$$

where q_i^{RFT} measures manipulation feasibility, $\ell(P_i^{\text{RFT}})$ is the path length, and λ_{path} balances affordance against path efficiency. High-scoring records form the retained set $\mathcal{D}_{\text{RFT}}^{+}$.

After expanding the retained trajectories into step-level context–action pairs, RFT performs supervised fine-tuning on the retained navigation actions:

$$\mathcal{L}_{\text{RFT}} = -\mathbb{E}_{(h,a^n)\sim\mathcal{D}_{\text{RFT}}^{+}} \left[\log \pi_\theta(a^n \mid h)\right]. \tag{5}$$

Here, (h, a^n) denotes a model-context and navigation-action pair from an accepted policy-generated trajectory. Initialized from $\pi_{\theta_{\text{co}}}$, the policy π_θ minimizes the expected token-level negative log-likelihood over the retained pairs. Thus, manipulation outcomes determine which policy-generated trajectories serve as SFT targets, while rejected rollouts contribute no gradient.

3.5 Agentic Counterfactual RLAIF from Execution Feedback

In the final training stage, we perform RLAIF, following the broader paradigm of using AI-generated feedback to iteratively improve embodied policies (Guo et al., 2026), by combining agentic counterfactual refinement with execution-based validation. Given a factual navigation–manipulation rollout, an execution agent $\mathcal{C}$ uses the execution feedback to propose a bounded navigation correction, such as moving closer, adjusting the approach yaw, or reapproaching around an obstruction. We then reset the simulator to the same task initialization and evaluate the refined decision in closed loop with the frozen Nav4Manip-M $\mathcal{W}$. We repeat this collection-and-update cycle for K_{iter} rounds, where K_{iter} is the RLAIF-round hyperparameter.

For task $(\mathcal{E}_i, l_i)$, let x_i denote the factual context, including the instruction, navigation decision, execution outcome, and updated environment state. The agent proposes the counterfactual refinement

$$y_i^{+} = \mathcal{C}(x_i), \tag{6}$$

where y_i^{+} is the refined navigation decision. Let r_i^{-} and r_i^{+} be the manipulation scores before and after refinement. We retain only execution-validated improvements:

$$\Delta r_i = r_i^{+} - r_i^{-}, \qquad \mathcal{D}_{\text{AI}} = \left\{(x_i, y_i^{+}) \mid \Delta r_i \geq \tau_\Delta\right\}. \tag{7}$$

Algorithm 1 Execution-validated agentic counterfactual RLAIF

Require: navigation policy π_θ, agent $\mathcal{C}$, frozen NAV4MANIP-M $\mathcal{W}$, tasks $\{(\mathcal{E}_i, l_i)\}$, threshold τ_Δ, RLAIF rounds K_{iter}

for $k = 1, \ldots, K_{\text{iter}}$ **do**
  $\mathcal{B} \leftarrow \varnothing$
  for each sampled task $(\mathcal{E}_i, l_i)$ **do**
    $(x_i, r_i^-) \leftarrow \textsc{ClosedLoopEvaluate}(\pi_\theta, \mathcal{W}, \mathcal{E}_i, l_i)$
    $y_i^+ \leftarrow \textsc{AgenticRefine}(\mathcal{C}, x_i)$
    Reset $\mathcal{E}_i$ to the same task initialization
    $r_i^+ \leftarrow \textsc{EvaluateRefinement}(\pi_\theta, \mathcal{W}, \mathcal{E}_i, l_i, x_i, y_i^+)$
    if $r_i^+ - r_i^- \geq \tau_\Delta$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x_i, y_i^+)\}$
    end if
  end for
  Fine-tune π_θ on $\mathcal{B}$ using $\mathcal{L}_{\text{RLAIF}}$
end for
return π_{θ^*}

We then fine-tune NAV4MANIP-N on the retained refinements using standard supervised learning:

$$\mathcal{L}_{\text{RLAIF}} = -\mathbb{E}_{(x,y^+)\sim\mathcal{D}_{\text{AI}}} \left[\log \pi_\theta(y^+ \mid x)\right], \tag{8}$$

where π_θ is initialized from the RFT navigator. This objective distills AI-generated, execution-validated refinements into NAV4MANIP-N, so neither the agent nor counterfactual evaluation is required at inference time. Algorithm 1 summarizes the complete procedure. Appendix B.3 and Appendix B.4 provide the agent loop, prompts, and training settings.

4 EXPERIMENTS

4.1 SETUP

We evaluate closed-loop continuous navigation on the val-unseen splits of R2R and RxR and on the official test split of the fine- and coarse-grained VLNVerse tasks. We further evaluate closed-loop navigation and manipulation on our Nav-N-Manip benchmark. At each terminal pose in Nav-N-Manip, we evaluate manipulation over 16 independently reset trials with object-pose perturbations. SR@Manip reports average single-attempt manipulation success, while SR@E2E measures joint navigation-to-grasp success, requiring arrival within 0.6 m of the reference goal and successful grasping; navigation failures count as failures. Following R2R conventions, we report success rate (SR), oracle success rate (OSR), success weighted by path length (SPL), navigation error (NE), and trajectory length (TL). Navigation SR@r measures the percentage of episodes whose final agent position is within r meters of the reference goal, using the benchmark's prescribed distance metric. Unless otherwise noted, R2R and RxR use the standard 3 m success radius; for our benchmark, we additionally report results at radii of 1.5 m, 1 m, and 0.6 m. NAV4MANIP-N operates on monocular RGB observations, executing one discrete action before receiving a new observation and replanning.

4.2 MAIN RESULTS

Figure 4 presents a unified comparison on R2R and RxR val-unseen and on the official test split of the fine- and coarse-grained VLNVerse tasks. We report the two primary metrics shared by all settings: success rate within 3 m (SR@3) and success weighted by path length at the same threshold (SPL@3). Methods are additionally marked as single-view (S) or multi-view (M), allowing NAV4MANIP-N to be compared under its monocular single-view protocol while retaining broader multi-view reference systems. Appendix Tables 6 and 7 report the complete benchmark-specific metrics, including OSR, NE, and trajectory length where available. Among single-view methods, NAV4MANIP-N achieves the best SR@3 and SPL@3 on both R2R (60.03/56.79) and RxR (62.06/52.60). It also matches the multi-view NavFoM closely: on R2R, its SPL@3 is higher (56.79 vs. 55.30), while on RxR its SR@3 and SPL@3 are within 2.34 and 3.60 points, respectively. On the VLNVerse test split, NAV4MANIP-N further surpasses the strongest multi-view baseline, Qwen-RobotNav, by 12.33/13.66 points in SR@3/SPL@3 on the fine-grained task and 16.58/16.95 points on the coarse-grained task.

We additionally evaluate whether navigation endpoints support downstream grasping in simulation. Table 2 compares instruction-following and ObjectNav task formulations. For this comparison, all

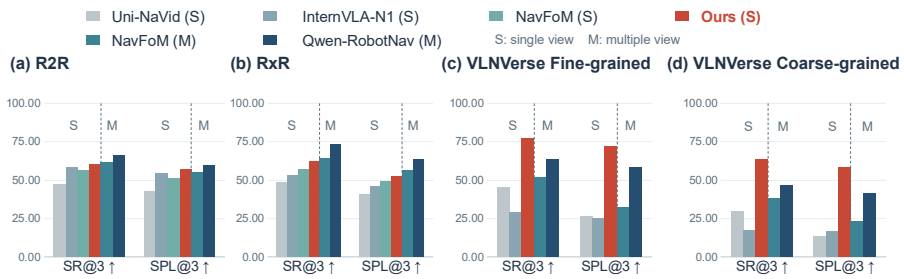

Figure 4: Closed-loop navigation comparison on (a) R2R val-unseen, (b) RxR val-unseen, and the VLNVerse test split for (c) fine-grained and (d) coarse-grained navigation. Each panel reports SR@3 and SPL@3 (higher is better). S and M denote single-view and multi-view visual inputs, respectively. Complete numerical results are provided in Appendix C.1.

Table 2: Simulation navigation-to-grasp results under instruction following and ObjectNav. All external navigation baselines use the same downstream manipulation interface as Nav4Manip for grasp execution and evaluation. All metrics follow the evaluation protocol in Section 4.1. Best/second-best results per setting are bolded/underlined.

Model	Task	SR@3.0 ↑	SR@1.5 ↑	SR@1.0 ↑	SR@0.6 ↑	NE ↓	SR@Manip ↑	SR@E2E ↑
OmniVLA	instruction	0.8	0	0	0	7.29	0	0
InternVLA-N1	instruction	3.5	0.4	0.4	0	6.47	0	0
NaVid	instruction	<u>35.7</u>	<u>14.7</u>	8.1	4.7	<u>4.33</u>	0.78	0.39
Uni-NaVid	instruction	30.2	14.0	<u>11.2</u>	<u>5.0</u>	5.07	<u>1.94</u>	<u>1.94</u>
NAV4MANIP	instruction	**90.3**	**85.3**	**81.8**	**71.3**	**1.12**	**61.17**	**56.31**
OmniVLA	ObjectNav	0.8	0	0	0	7.26	0	0
InternVLA-N1	ObjectNav	2.7	0.4	0	0	6.97	0	0
NaVid	ObjectNav	<u>37.2</u>	<u>18.2</u>	9.7	4.7	<u>4.37</u>	0.78	0
Uni-NaVid	ObjectNav	28.7	12.4	<u>10.9</u>	<u>5.0</u>	4.48	<u>3.88</u>	<u>2.71</u>
NAV4MANIP	ObjectNav	**88.8**	**83.0**	**79.1**	**71.7**	**1.18**	**52.78**	**48.15**

models are fine-tuned for one epoch on the NAV-N-MANIP training split and evaluated on its test split. Under instruction following, NAV4MANIP achieves 71.3% SR@0.6 and 56.31% end-to-end success; under ObjectNav, it achieves 71.7% SR@0.6 and 48.15% end-to-end success. Appendix Table 13 further compares our full system with general-purpose agents on ten NAV-N-MANIP episodes. GPT-6 Astra reaches all ten goals within 1.5 m, nine within 1.0 m, and six within 0.6 m, yielding 5/10 manipulation successes with 445 s navigation latency. Our system succeeds in 8/10 episodes at every radius, completes manipulation in 7/10, and reduces navigation latency to 14 s.

Figure 5 demonstrates the complete system on real-world dexterous mobile-manipulation tasks. In Figure 5(a), the robot navigates to a shelf, refines its base pose, grasps and transports the target object, and delivers it to the recipient. Figure 5(b) shows dynamic obstacle handling: the robot pauses when a person enters its path and resumes after the path clears. Figure 5(c) illustrates agentic counterfactual refinement, where the agent compares corrective actions using task feedback, adjusts the base pose, and re-observes the scene before manipulation.

4.3 ABLATION OF THE TRAINING PIPELINE

Table 3 evaluates the training pipeline under a shared architecture and evaluation protocol. The baseline is trained on RxR, HM3D ObjectNav, and R2R. Multi-task co-tuning (MTC) additionally incorporates our generated data, comprising Rendered EQA and Nav4Manip-Pairs; verifier-guided RFT then learns from manipulation-verified policy rollouts. MTC provides the largest initial gain, raising SR@0.6 from 10.7 to 52.7 and SR@E2E from 4.07 to 31.01. RFT further improves precise stopping: relative to MTC, SR@1.0 increases from 69.8 to 73.8, SR@0.6 from 52.7 to 60.9, and NE decreases from 1.81 to 1.76. These gains show that filtering policy-generated trajectories using downstream execution outcomes primarily improves the fine-grained endpoint accuracy required for manipulation.

Starting from the same RFT checkpoint, ACR performs counterfactual refinement at inference time without parameter updates, whereas RLAIF distills execution-validated refinements into the final NAV4MANIP policy. Both substantially improve the navigation–manipulation handoff: relative to

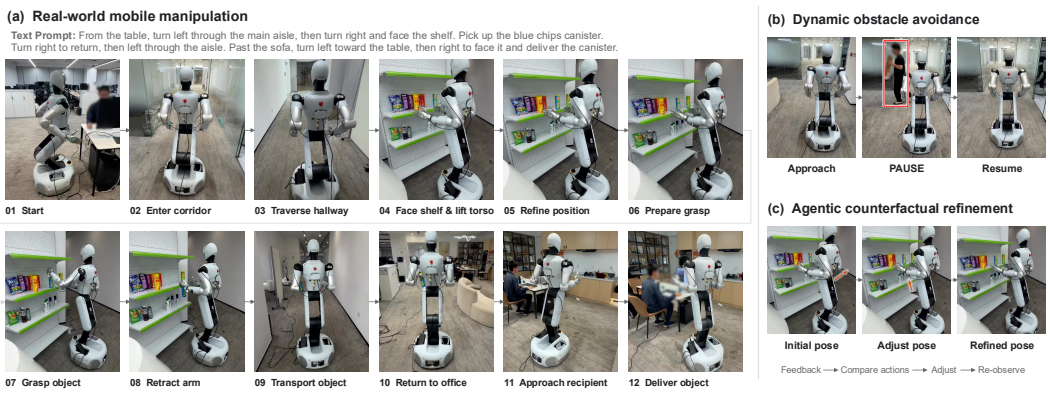

Figure 5: **Real-world navigation and dexterous mobile manipulation.** (a) Long-horizon navigation and object delivery. (b) Dynamic obstacle avoidance. (c) Agentic counterfactual base-pose refinement. See Appendices F and H for deployment details and demonstration tasks.

Table 3: Training-pipeline ablation on NAV-N-MANIP. The baseline uses RxR, HM3D ObjectNav, and R2R; Multi-Task Co-Tuning (MTC) adds Rendered EQA and Nav4Manip-Pairs. Rejection-Sampling Fine-Tuning (RFT) builds on MTC; Agentic Counterfactual Refinement (ACR; inference time, no training) and two-iteration RLAIF build on RFT. Alternating optimization performs two navigation–manipulation rounds with the same number of training samples per stage as RLAIF. Best/second-best results are bolded/underlined.

Model variant	SR@3.0 ↑	SR@1.5 ↑	SR@1.0 ↑	SR@0.6 ↑	NE ↓	SR@Manip ↑	SR@E2E ↑
Baseline	56.2	28.7	19.6	10.7	3.54	6.01	4.07
+ MTC	80.2	77.1	69.8	52.7	1.81	39.34	31.01
+ RFT	80.8	76.7	73.8	60.9	1.76	38.18	31.79
+ ACR (IT)	<u>82.4</u>	<u>78.5</u>	<u>76.6</u>	64.3	<u>1.55</u>	<u>53.88</u>	<u>47.10</u>
+ RLAIF (NAV4MANIP)	**89.5**	**84.1**	**80.4**	**71.7**	**1.15**	**56.87**	**52.13**
Alternating Optimization	80.6	78.3	76.4	<u>68.4</u>	1.61	41.47	34.30

RFT, ACR raises SR@Manip from 38.18 to 53.88 and SR@E2E from 31.79 to 47.10. With two training iterations, RLAIF achieves the best result across all navigation and manipulation metrics, including 71.7 SR@0.6, 1.15 m NE, 56.87 SR@Manip, and 52.13 SR@E2E. With matched per-stage training sample counts, two rounds of alternating navigation–manipulation optimization require four module-training stages but achieve only 34.30% SR@E2E, compared with 52.13% for two-round RLAIF with a fixed manipulation policy. Compared with ACR, it improves SR@0.6 by 7.4 points, reduces NE by 0.40 m, and raises SR@Manip and SR@E2E by 2.99 and 5.03 points, respectively. RLAIF therefore provides stronger navigation and handoff quality while eliminating the inference-time refinement agent: Appendix Table 14 shows that total inference time decreases from 50.33 s to 13.80 s, a 72.6% reduction. Additional navigation diagnostics are provided in Appendix C.

5 CONCLUSION

We introduced NAV4MANIP, a bidirectionally coupled dexterous mobile-manipulation system. Our automated data-generation pipeline pairs long-horizon navigation trajectories with rollout-derived manipulation affordances. Multi-task co-tuning, verifier-guided RFT, and RLAIF from execution-grounded counterfactual preferences jointly replace coarse, one-shot endpoint prediction with manipulation-informed navigation learning. Experiments on our NAV-N-MANIP benchmark demonstrate that NAV4MANIP enables navigation and manipulation to co-evolve through their mutual feedback, improving the system's ability to solve real-world dexterous mobile-manipulation tasks.

Limitations. Our evaluation uses a single Galbot G3 robot with Sharpa hands, and our navigation data cover only indoor environments. Future work should assess transfer across manipulation policies, robot embodiments, and outdoor settings. Our bidirectional co-evolution framework retains separate navigation and manipulation models; extending it to a unified model with a shared action space for long-horizon mobile manipulation is a promising direction.

AI USE STATEMENT

Generative AI tools were used to annotate our generated dataset, provide feedback for RLAIF, serve as agent baselines during evaluation, and assist with language polishing. The authors reviewed the AI-assisted content and take full responsibility for the final manuscript.

ETHICS STATEMENT

This work primarily uses generated simulation environments and does not rely on personal or human-subject data. Learned navigation and manipulation components may fail under distribution shift and should be deployed with collision checking, hardware limits, emergency stops, and appropriate human supervision. Any data release will document model-generated annotations, known limitations, and applicable asset licenses.

REPRODUCIBILITY STATEMENT

The main paper specifies the task, training objectives, and refinement procedure. Appendices B and D provide model, training, data-generation, and evaluation details; Appendix C.1 reports complete benchmark results.

REFERENCES

Adobe. Mixamo. https://www.mixamo.com/. Accessed: 2026-09-24.

Dong An, Yuankai Qi, Yangguang Li, Yan Huang, Liang Wang, Tieniu Tan, and Jing Shao. Bevbert: Multimodal map pre-training for language-guided navigation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5130–5145, 2024.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko S"underhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 3674–3683. IEEE, 2018.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee Wong. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9796–9810, 2024.

Jiayi Chen, Yubin Ke, Lin Peng, and He Wang. Dexonomy: Synthesizing all dexterous grasp types in a grasp taxonomy. In *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2504.18829.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847, 2021.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16516–16526. IEEE, 2022.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged Robot Vision-Language-Action Model for Navigation. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.018.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026.

Zedong Chu, Shichao Xie, Xiaolong Wu, Yanfen Shen, Minghua Luo, Zhengbo Wang, Fei Liu, Xiaoxu Leng, Junjun Hu, Mingyang Yin, et al. Abot-n0: Technical report on the vla foundation model for versatile embodied navigation. *arXiv preprint arXiv:2602.11598*, 2026.

Zehua Fan, Wenqi Lyu, Wenxuan Song, Linge Zhao, Yifei Yang, Xi Wang, Junjie He, Lida Huang, Haiyan Liu, Bingchuan Sun, et al. Prospect: Unified streaming vision-language navigation via semantic–spatial fusion and latent predictive representation. *arXiv preprint arXiv:2603.03739*, 2026.

Yusen Feng, Bingchen Han, Jiangran Lyu, Kai Liu, Yixin Zheng, Yuxuan Wan, Weiheng Liu, Sun Han, Ruiqin Li, Yulong Zhang, Fangfu Liu, Xuesong Shi, Libin Liu, Yizhou Wang, Zhizheng Zhang, and He Wang. WAM-TTT: Steering world-action models by watching human play at test time. *arXiv preprint arXiv:2607.06988*, 2026.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation using low-cost whole-body teleoperation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4066–4083, 2025.

Ruiyan Gong, Yingnan Guo, Junjun Hu, Jintao Kong, Xiaoxu Leng, Tianlun Li, Weize Li, Fei Liu, Zhicheng Liu, Jia Lu, et al. Abot-n1: Toward a general visual language navigation foundation model. *arXiv preprint arXiv:2607.10383*, 2026.

Raktim Gautam Goswami, Amir Bar, David Fan, Tsung-Yen Yang, Gaoyue Zhou, Prashanth Krishnamurthy, Michael Rabbat, Farshad Khorrami, and Yann LeCun. World models for learning dexterous hand-object interactions from human videos. *arXiv preprint arXiv:2512.13644*, 2025.

Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1614–1623. IEEE, 2021.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026.

Mingfei Han, Liang Ma, Kamila Zhumakhanova, Ekaterina Radionova, Jingyi Zhang, Xiaojun Chang, Xiaodan Liang, and Ivan Laptev. Roomtour3d: Geometry-aware video-instruction tuning for embodied navigation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27586–27596. IEEE, 2025.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13134–13143. IEEE, 2020.

Hesai Technology. Hesai showcases kosmo spatial intelligence platform and robotic lidar at waic 2026, underscoring its physical ai vision, July 2026. Accessed: 2026-09-24.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Omnivla: An omni-modal vision-language-action model for robot navigation. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. *arXiv preprint arXiv:2011.13922*, 2020.

Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1862–1872, 2019.

Haoran Jiang, Jin Chen, Qingwen Bu, Li Chen, Modi Shi, Yanjie Zhang, Delong Li, Chuanzhe Suo, Hongyang Li, et al. Wholebodyvla: Towards unified latent vla for whole-body loco-manipulation control. In *International Conference on Learning Representations*, volume 2026, pp. 157438–157461, 2026.

Glenn Jocher, Jing Qiu, Mengyu Liu, Shuai Lyu, Fatih Cagatay Akyon, and Muhammet Esat Kalfaoglu. Ultralytics YOLO26: Unified real-time end-to-end vision models. *arXiv preprint arXiv:2606.03748*, 2026.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pp. 104–120. Springer, 2020.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4392–4412, 2020.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

Zhe Li, Zhenzhe Zhang, Yangyang Wei, Wenjie Zhang, Xichen Yuan, Peiyuan Zhi, Gen Li, Xinying Guo, Fengjie Gao, Jianfei Yang, et al. ω-0: A latent predictive world action model for concurrent humanoid loco-manipulation. *arXiv preprint arXiv:2608.06375*, 2026a.

Zhen Li, Chuanhao Li, Xiaofeng Mao, Shaoheng Lin, Ming Li, Shitian Zhao, Zhaopan Xu, Xinyue Li, Yukang Feng, Jianwen Sun, et al. Sekai: A video dataset towards world exploration. *Advances in Neural Information Processing Systems*, 38, 2026b.

Guoxin Lian, Shuo Wang, Yucheng Wang, Yongcai Wang, Maiyue Chen, Kaihui Wang, Bo Zhang, Zhizhong Su, Deying Li, and Zhaoxin Fan. Mapdream: Task-driven map learning for vision-language navigation. *arXiv preprint arXiv:2602.00222*, 2026.

Sihao Lin, Zerui Li, Xunyi Zhao, Gengze Zhou, Liuyi Wang, Rong Wei, Rui Tang, Juncheng Li, Hanqing Wang, Jiangmiao Pang, et al. Vlnverse: A benchmark for vision-language navigation with versatile, embodied, realistic simulation and evaluation. *arXiv preprint arXiv:2512.19021*, 2025.

Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. LDA-1B: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

Lakshadeep Naik, Sinan Kalkan, Sune L. Sørensen, Mikkel B. Kjærgaard, and Norbert Krüger. Basenet: a learning-based mobile manipulator base pose sequence planning for pickup tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3666–3673. IEEE, 2024.

Dantong Niu, Zhuoyang Liu, Zekai Wang, Boning Shao, Zhao-Heng Yin, Anirudh Pai, Yuvan Sharma, Stefano Saravalle, Ruijie Zheng, Jing Wang, Ryan Punamiya, Mengda Xu, Yuqi Xie, Yunfan Jiang, Letian Fu, Konstantinos Kallidromitis, Matteo Gioia, Junyi Zhang, Jiaxin Ge, Haiwen Feng, Fabio Galasso, Wei Zhan, David M. Chan, Yutong Bai, Roei Herzig, Jiahui Lei, Li Fei-Fei, Ken Goldberg, Jitendra Malik, Pieter Abbeel, Yuke Zhu, Danfei Xu, Linxi Fan, and Trevor Darrell. T-Rex: Tactile-reactive dexterous manipulation. *arXiv preprint arXiv:2606.17055*, 2026.

NVIDIA. Isaac Sim. Software, version 6.0.1, 2026. URL `https://github.com/isaac-sim/IsaacSim`.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9982–9991, 2020.

Modi Shi, Shijia Peng, Jin Chen, Haoran Jiang, Tianyu Li, Di Huang, Ping Luo, Hongyang Li, and Li Chen. Egohumanoid: Unlocking in-the-wild loco-manipulation with robot-free egocentric demonstration. *arXiv preprint arXiv:2602.10106*, 2026.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2610–2621, 2019.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Liuyi Wang, Xinyuan Xia, Hui Zhao, Hanqing Wang, Tai Wang, Yilun Chen, Chengju Liu, Qijun Chen, and Jiangmiao Pang. Rethinking the embodied gap in vision-and-language navigation: A holistic study of physical and visual disparities. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9455–9465. IEEE, 2025a.

Runze Wang, Yuqian Fu, Yu Li, Tao Lin, Tianwen Qian, Mohamed Elhoseiny, Bo Zhao, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Afford-vla: Action-aligned visual planning via internalized affordance. *arXiv preprint arXiv:2605.24203*, 2026a.

Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Hongsheng Li, Yue Liao, Si Liu, et al. Towards realistic uav vision-language navigation: Platform, benchmark, and methodology. In *International Conference on Learning Representations*, volume 2025, pp. 7292–7310, 2025b.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6622–6631. IEEE, 2019.

Xinjie Wang, Liu Liu, Taojun Ding, Andrew Choi, Chaodong Huang, Mengao Zhao, Ziang Li, Jackson Jiang, Chunlei Yu, Shengxiang Liu, et al. Embodiedgen v2: An agentic, simulation-ready 3d world engine for embodied ai. *arXiv preprint arXiv:2607.07459*, 2026b.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13753–13762, 2024.

Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986. IEEE, 2023.

Meng Wei, Chenyang Wan, Peng Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pp. 12380–12396, 2026a.

Meng Wei, Chenyang Wan, Peng Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pp. 12380–12396, 2026b.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, Xihui Liu, and Jiangmiao Pang. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. In *IEEE International Conference on Robotics and Automation*, 2026c.

Songlin Wei, Hongyi Jing, Boqian Li, Zhenyu Zhao, Jiageng Mao, Zhenhao Ni, Sicheng He, Sheng Zang, Xiawei Liu, Kaidi Kang, Jie Liu, Weiduo Yuan, Marco Pavone, Di Huang, and Yue Wang. ψ-0: An open foundation model towards universal humanoid loco-manipulation. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026d. doi: 10.15607/RSS.2026.XXII.021.

Xinda Xue, Junjun Hu, Minghua Luo, Shichao Xie, Jintao Chen, Zixun Xie, Quan Kuichen, Wei Guo, Zedong Chu, Mu Xu, et al. Omninav: A unified framework for prospective exploration and visual-language navigation. In *International Conference on Learning Representations*, volume 2026, pp. 106216–106231, 2026.

Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, et al. Habitat-matterport 3d semantics dataset. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4927–4936. IEEE, 2023.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Jim Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin S. Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, Zsolt Kira, Manolis Savva, Angel X. Chang, Devendra Singh Chaplot, Dhruv Batra, Roozbeh Mottaghi, Yonatan Bisk, and Chris Paxton. Homerobot: Open-vocabulary mobile manipulation. In *Conference on Robot Learning*, 2023.

Kun-Yang Yu, Yingzhe Li, Hongyu Xu, Shi-Yu Tian, Zhi Zhou, Yang Chen, Ming Yang, Sheng Wang, Qing Yu, Lan-Zhe Guo, et al. Lookstep: Efficient vision-language navigation with linguistic foresight and event driven memory. *arXiv preprint arXiv:2609.02350*, 2026a.

Qize Yu, Jiadi You, Yuran Wang, Jiaqi Liang, Bowen Ping, Yang Tian, Yue Chen, Minghong Cai, Zeying Gong, Ruihai Wu, et al. Affordancevla: A vision-language-action model empowering action generation through affordance-aware understanding. *arXiv preprint arXiv:2606.06155*, 2026b.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, and Xing Wei. Janusvln: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pp. 33001–33026, 2026.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024a.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*, 2024b.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, et al. Embodied navigation foundation model. In *International Conference on Learning Representations*, volume 2026, pp. 127293–127322, 2026a.

Jiazhao Zhang, Gengze Zhou, Hale Yin, Yiyang Huang, Zixing Lei, Qihang Peng, Haoqi Yuan, Jie Zhang, Xudong Guo, Xiaoyue Chen, et al. Qwen-robotnav technical report: A scalable navigation model designed for an agentic navigation system. *arXiv preprint arXiv:2606.18112*, 2026b.

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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6315–6326. IEEE, 2025.

Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist model for embodied navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13624–13634. IEEE, 2024.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. EgoScale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 7641–7649, 2024.

A Dexterous Mobile-Manipulation Task Definition

A dexterous mobile-manipulation task is defined by a task instance s, a language instruction l, a navigation policy π_θ, and a downstream manipulation policy π_m. In our system, π_m is instantiated by Nav4Manip-M. For episode i, the navigator generates a trajectory $\tau_i^n = (o_{1:T_i}^n, a_{1:T_i}^n)$ that terminates at the base pose $b_i = (x_i, y_i, \theta_i)$, from which the manipulation policy attempts the requested interaction. Unlike standard VLN, merely reaching a meter-scale goal region is insufficient: dexterous mobile manipulation requires a sufficiently precise terminal pose, with a base position and approach yaw that allow π_m to complete the manipulation task without collisions. A successful dexterous mobile-manipulation system must therefore combine long-horizon navigation with precise terminal alignment at the navigation–manipulation interface.

Figure 6 provides a unified view of three optimization strategies at the navigation–manipulation interface. The navigation dimension x encompasses the complete navigation problem: long-horizon instruction following, obstacle avoidance, goal progress, and selection of the terminal base position and approach yaw. The manipulation dimension y represents the arm and finger poses required to physically interact with the target object. The scene-level examples ground this distinction in a bowl-picking task. Navigation-only training can reach endpoint A, which is close to the bowl under a conventional navigation metric, without recognizing that the chairs block a collision-free arm approach. Optimizing only along x therefore cannot use manipulation outcomes to distinguish a geometrically valid endpoint from one that supports the required arm–hand interaction, limiting end-to-end capability.

A second strategy is coordinate-wise optimization, illustrated in the middle panels of Figure 6, which alternates navigation updates along x with manipulation updates along y. These panels expose two practical inefficiencies. First, manipulation rollouts at an unsuitable endpoint such as A repeatedly encounter the chair obstruction, making useful demonstrations difficult and costly to obtain. Second, after navigation moves the terminal pose from A to B, actions collected at A assume a different base pose, approach direction, and robot–object geometry. Reusing them at B creates an action mismatch, while correcting it requires another round of rollout collection and manipulation retraining. Our controlled comparison in Table 3 evaluates two navigation–manipulation rounds, comprising four module-training stages. With matched per-stage training sample counts, this baseline achieves lower end-to-end success than two-round RLAIF, which avoids repeated manipulation-policy training. Appendix C.3 provides the experimental details.

Our staged co-evolution strategy, shown in the rightmost panel of Figure 6, follows a shorter coupled path. Multi-task co-tuning establishes long-horizon navigation and manipulation-aware endpoint reasoning, allowing the navigator to favor endpoint B on the unobstructed side. The aligned manipulation traces in Nav4Manip-Pairs also train Nav4Manip-M on the corresponding handoff configurations before feedback-based navigation refinement. Verifier-guided RFT then uses downstream manipulation outcomes to improve policy-generated navigation trajectories, and agentic counterfactual RLAIF distills execution-validated corrections into the navigator. Together, these stages improve the navigation–manipulation interface and enable a smooth transition from a reachable approach to a successful grasp while preserving separate policies. Unlike alternating optimization, our approach does not repeatedly retrain the two modules; unlike monolithic policies, it requires no shared base-and-arm action space. The feedback agent is used only during training; inference requires only the trained navigator and Nav4Manip-M.

B System Architecture

B.1 Multi-Scale Visual Encoding

Following the multi-scale visual-history encoding strategy of Uni-NaVid (Zhang et al., 2024a), Nav4Manip-N preserves both fine-grained evidence at the current step and long-horizon context by organizing its visual observations into four temporal tiers: the current frame and recent-, middle-, and long-range history. The current frame is retained at the highest spatial resolution, whereas progressively older observations are represented with progressively fewer visual tokens. This hierarchy allocates most of the visual-token budget to information that is immediately relevant for control while maintaining a compact record of earlier scene layouts, landmarks, and interactions. With a

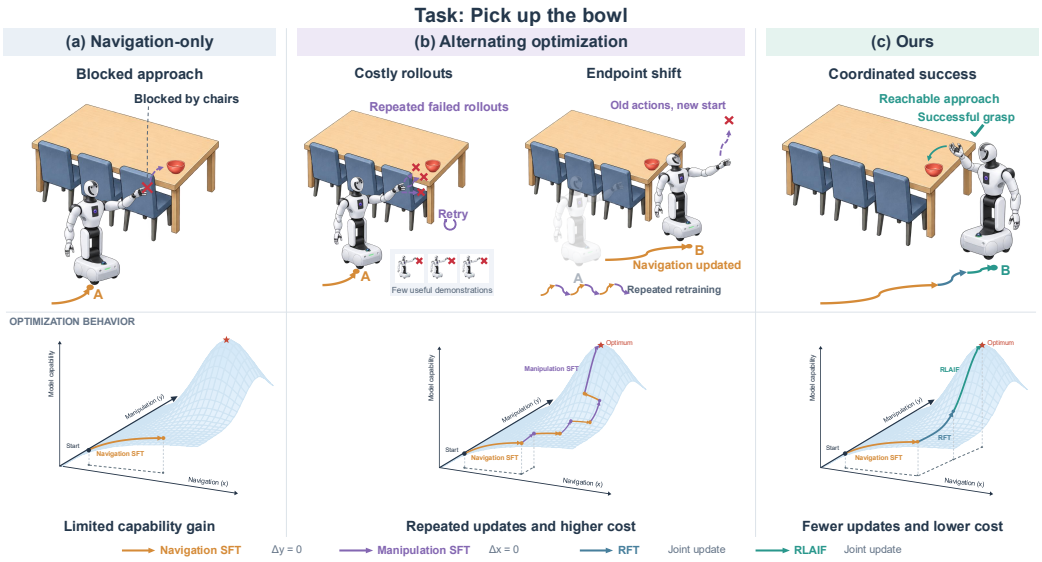

Figure 6: **Comparison of optimization strategies at the navigation–manipulation interface.** Navigation-only optimization stops at a chair-blocked endpoint. Alternating optimization wastes manipulation rollouts there and suffers action mismatch after navigation moves the endpoint from A to B. Our staged co-evolution instead selects a reachable approach and smoothly completes the grasp with fewer updates.

history stride of four, the default recent-, middle-, and long-range capacities are 8, 16, and 64 frames, respectively. During training, we randomize the recent and middle capacities over $\{4, 6, 8, 12\}$ and $\{8, 12, 16, 24\}$, while keeping the long-range capacity at 64, thereby exposing the model to histories of varying duration.

All tiers are encoded by the shared Qwen3-VL vision encoder and multimodal spatial merger (Bai et al., 2025); no additional learned projector is introduced for history frames. The vision tokenizer uses 16×16 patches with a spatial merge size of two. Thus, the current 640×480 observation produces a 20×15 grid containing 300 merged visual tokens. For an earlier observation o_i assigned to tier $r \in \{\text{recent}, \text{mid}, \text{long}\}$, we apply adaptive average pooling to the merged feature grid,

$$v_i^{(r)} = \text{Pool}_r(\text{Enc}_{\text{vis}}(o_i)) \,,$$

yielding grids of $5\times4 = 20$, $3\times2 = 6$, and $2\times2 = 4$ tokens for the recent, middle, and long tiers, respectively. Letting $v_t^{(0)}$ denote the current-frame tokens, we concatenate the tiers from the oldest temporal scale to the current observation:

$$V_t = \left[\{v_i^{(\text{long})}\};\ \{v_i^{(\text{mid})}\};\ \{v_i^{(\text{recent})}\};\ v_t^{(0)}\right]. \tag{9}$$

The default visual-token budget is 1,024. During training, we sample an integer budget between 384 and 1,024 for each batch. If the packed sequence exceeds this budget, we discard the oldest long-range frames first, followed by middle- and then recent-range frames, while always preserving the current observation. Validation and deployment use the deterministic 1,024-token budget.

Temporal order is represented by textual timestamps, such as `<3.5 seconds>`, placed adjacent to each image block. The visual M-RoPE temporal index remains zero because the observations are provided as individual images. When odometry is available, relative motion $\Delta p = (\Delta x, \Delta y, \Delta\psi)$ is normalized by $(0.25\text{ m}, 0.25\text{ m}, 15^\circ)$, encoded using sine and cosine functions at frequencies $\{1, 2, 4, 8\}$, and projected by a $24 \to 256 \to 4096$ MLP whose output is added to every token of the corresponding frame. During training, we perturb each relative-motion estimate with zero-mean Gaussian noise whose standard deviations are $(0.02\text{ m}, 0.02\text{ m}, 0.017\text{ rad})$ for $(\Delta x, \Delta y, \Delta\psi)$, respectively. When odometry is unavailable, we replace the motion encoding with a learned `no_odom` embedding. Recent and middle tiers additionally include serialized action history; the entire action-history side channel is dropped with probability 0.1 during training.

Table 4: Backbone configuration of NAV4MANIP-N.

Component	Configuration
Vision Encoder	
Layers / attention heads	27 / 16
Hidden / intermediate dimensions	1,152 / 4,304
Output dimension	4,096
Patch / spatial merge / temporal patch	16 / 2 / 2
Deep-stack layers	8, 16, 24
Language Model	
Layers	36
Hidden / intermediate dimensions	4,096 / 12,288
Attention / KV heads	32 / 8
Vocabulary size	151,936
Maximum context length	262,144
Positional Encoding	
M-RoPE sections	$[24, 20, 20]$
RoPE base	5,000,000

For a uniform interface across simulation, offline datasets, and real-robot deployment, we exchange observations as JPEG images. Each image is decoded in BGR format, center-cropped to the expected 4:3 view, resized using area interpolation, and converted to RGB before visual encoding. The current, recent-, middle-, and long-range tiers use resolutions of 640×480, 320×224, 160×128, and 96×96, respectively. To reduce sensitivity to a particular camera setup, the training data incorporate variations in camera height and pitch, illumination, and horizontal field of view. Together with the history and token-budget randomization described above, these variations enable NAV4MANIP-N to tolerate moderate changes in viewpoint, lighting conditions, and camera intrinsics in real-world applications.

B.2 DETAILED NAV4MANIP-N ARCHITECTURE

We initialize NAV4MANIP-N from Qwen3-VL-8B-Instruct (Bai et al., 2025). The backbone comprises a vision transformer, a multimodal merger, and an autoregressive language model, with the configuration summarized in Table 4. Although the vision backbone has a temporal patch size of two, we supply the observation history as individually encoded images. During training, we jointly optimize the vision encoder, multimodal merger, and language model using full-parameter fine-tuning.

Learned queries and prediction heads. Inspired by the use of learnable queries to elicit task-specific visual and affordance representations in BLIP-2, AffordanceVLA, and Afford-VLA (Li et al., 2023; Yu et al., 2026b; Wang et al., 2026a), we adapt the pretrained backbone by adding ten learned query tokens: one dense-affordance query, eight endpoint-candidate queries, and one action query. Each query has a learnable embedding that is initialized as a model parameter and optimized end to end with the backbone and prediction heads. The queries are registered as additional tokenizer entries, inserted immediately after the assistant prefix in map–candidate–action order, and excluded from the language-model token loss. Their contextualized hidden states are passed to the corresponding prediction heads.

The hidden state of the dense-affordance query is decoded by a $4096\to4096\to8192$ GELU MLP and reshaped into the logit map $\hat{M}_t \in \mathbb{R}^{32\times32\times8}$. This ego-centered grid covers $x, y \in [-3.2, 3.2]$ m at a resolution of 0.2 m per cell, with the robot origin located at cell $(16, 16)$.

All eight candidate queries share a $4096 \to 512 \to 512 \to 7$ MLP. Candidate k predicts $[\hat{x}, \hat{y}, \hat{\theta}, \hat{u}, \hat{v}, z^v, z^s]$. We map $(\hat{x}, \hat{y})$ through $3.2\tanh(\cdot)$, $\hat{\theta}$ through $\pi\tanh(\cdot)$, and the normalized image coordinates $(\hat{u}, \hat{v})$ through a sigmoid; z^v and z^s are the visibility and affordance-score logits, respectively. Candidate slots retain the ordering provided by Nav4Manip-Pairs, so no Hungarian or nearest-neighbor matching is required. In addition, a goal/waypoint head operates on the fused multimodal state to predict two three-dimensional waypoints.

Multi-task co-tuning losses. For completeness, we expand all terms in the co-tuning objective from Eq. 3:

$$\mathcal{L}_{\text{co}} = \mathcal{L}_{\text{nav}} + \lambda_{\text{Afford}}\mathcal{L}_{\text{Afford}} + \lambda_{\text{EQA}}\mathcal{L}_{\text{EQA}}. \tag{10}$$

The navigation term is the autoregressive negative log-likelihood of the target action-token sequence $a^n = (a_1^n, \ldots, a_{T_a}^n)$ under the complete multimodal context h:

$$\mathcal{L}_{\text{nav}} = -\mathbb{E}_{(h,a^n)}\left[\sum_{s=1}^{T_a} \log \pi_\theta(a_s^n \mid h, a_{<s}^n)\right]. \tag{11}$$

Only target action tokens contribute to this loss; prompt, observation, and learned-query positions are masked. The grounded EQA term has the same token-level form. For an answer sequence $e = (e_1, \ldots, e_{T_e})$,

$$\mathcal{L}_{\text{EQA}} = -\mathbb{E}_{(h,e)}\left[\sum_{s=1}^{T_e} \log p_\theta(e_s \mid h, e_{<s})\right], \tag{12}$$

where the loss mask selects answer tokens and excludes the preceding visual and textual context. Dataset-specific masks activate $\mathcal{L}_{\text{nav}}$, $\mathcal{L}_{\text{EQA}}$, and $\mathcal{L}_{\text{Afford}}$ only when the corresponding supervision is available.

The affordance objective supervises the dense map and discrete endpoint candidates with rollout-derived targets:

$$\mathcal{L}_{\text{Afford}} = \mathcal{L}_{\text{dense}} + \mathcal{L}_{\text{cand}} + 0.5\mathcal{L}_{\text{rank}} + 0.2\mathcal{L}_{\text{cons}}. \tag{13}$$

Let n index samples in a mini-batch, $k \in \{1, \ldots, 8\}$ index candidate slots, and g index the $32 \times 32 \times 8$ position–yaw entries of the dense map. For entry (n, g), z_{ng} is the predicted logit, $a_{ng} \in [0, 1]$ is the affordance target, and $m_{ng} \in \{0, 1\}$ indicates whether the entry is labeled. With σ denoting the sigmoid function, the masked dense-map loss is

$$\ell_{\text{BCE}}(z, y) = -y \log \sigma(z) - (1 - y) \log(1 - \sigma(z)), \tag{14}$$

$$\mathcal{L}_{\text{dense}} = \frac{\sum_{n,g} m_{ng}\ell_{\text{BCE}}(z_{ng}, a_{ng})}{\max(1, \sum_{n,g} m_{ng})}. \tag{15}$$

This binary-cross-entropy loss supports soft targets, while entries with $m_{ng} = 0$ are ignored.

For candidate (n, k), let $\mathbf{p}_{nk} = (x, y)$, θ_{nk}, $\mathbf{q}_{nk} = (u, v)$, $v_{nk} \in \{0, 1\}$, and $s_{nk} \in [0, 1]$ be the target position, yaw, normalized image coordinates, visibility, and affordance score. Their predictions are denoted by hats, while z_{nk}^v and z_{nk}^s are the visibility and score logits from the candidate head. The mask $c_{nk} \in \{0, 1\}$ indicates whether the candidate slot is valid. The candidate loss is

$$\mathcal{L}_{\text{cand}} = \frac{1}{\max(1, \sum_{n,k} c_{nk})} \sum_{n,k} c_{nk}\Big[\|\hat{\mathbf{p}}_{nk} - \mathbf{p}_{nk}\|_1 + d_{\mathbb{S}^1}(\hat{\theta}_{nk}, \theta_{nk}) + v_{nk}\|\hat{\mathbf{q}}_{nk} - \mathbf{q}_{nk}\|_1 + \ell_{\text{BCE}}(z_{nk}^v, v_{nk}) + \ell_{\text{BCE}}(z_{nk}^s, s_{nk})\Big]. \tag{16}$$

Thus, unlabeled map cells and invalid candidate slots do not contribute to the loss, and image-coordinate regression is applied only to visible candidates. Yaw is supervised by the shortest angular distance $d_{\mathbb{S}^1}(\hat{\theta}, \theta) = \left|\text{atan2}\Big(\sin(\hat{\theta} - \theta), \cos(\hat{\theta} - \theta)\Big)\right|$.

For ranking, $\mathcal{P}$ contains valid candidate pairs (n, i, j) from the same sample whose target scores satisfy $s_{ni} > s_{nj} + 10^{-4}$. We apply the margin loss directly to the pre-sigmoid score logits:

$$\mathcal{L}_{\text{rank}} = \frac{\sum_{(n,i,j)\in\mathcal{P}} \max(0, 0.1 - z_{ni}^s + z_{nj}^s)}{\max(1, |\mathcal{P}|)}. \tag{17}$$

For consistency, $\mathcal{B}_{\text{cam}}$ denotes the mini-batch samples with camera parameters, $b \in \{1, \ldots, 8\}$ indexes yaw bins, Q maps a predicted ground position to its discrete BEV cell, and Π_n projects that position to normalized image coordinates for sample n and clips them to $[0, 1]^2$. The consistency loss is

$$\mathcal{L}_{\text{cons}} = \frac{1}{\max(1, \sum_{n\in\mathcal{B}_{\text{cam}},k} c_{nk})} \sum_{n\in\mathcal{B}_{\text{cam}},k} c_{nk}\Big[-\log \max\Big(10^{-4}, \max_b \sigma\big(z_{n,Q(\hat{\mathbf{p}}_{nk}),b}\big)\Big) + \|\hat{\mathbf{q}}_{nk} - \Pi_n(\hat{\mathbf{p}}_{nk})\|_1\Big]. \tag{18}$$

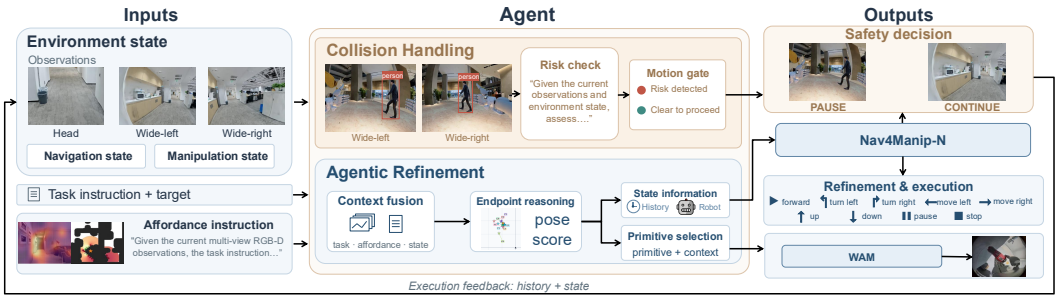

Figure 7: **Agent-driven counterfactual refinement in NAV4MANIP.** Environment observations and navigation–manipulation state, together with the task instruction and endpoint affordances, enter the coordinating agent. Agentic Refinement fuses this context to reason over endpoint poses and scores, update task-relevant history and robot state, and propose a bounded refinement or accept the endpoint. In parallel, Collision Handling converts detector evidence into a `PAUSE`/`CONTINUE` safety decision that gates motion execution. Safe refinements are executed through NAV4MANIP-N until the endpoint is accepted; the agent then selects a manipulation primitive and supplies its visual–textual context to NAV4MANIP-M. The resulting execution outcome is written back as history and updated state, closing the feedback loop for the next decision.

This term encourages a candidate to lie on a high-response map cell while agreeing with its geometric image projection. Following the implementation, it is applied to every valid candidate in camera-calibrated samples and is not masked by target visibility.

Action decoding. The action query conditions autoregressive generation of an action sequence. The complete action interface contains nine commands: `forward`, `turn_left`, `turn_right`, `move_left`, `move_right`, `up`, `down`, `pause`, and `stop`. During the initial multi-task co-tuning and RFT stages, explicit action supervision and constrained decoding cover the seven-action subset excluding `up` and `down`.

Agentic counterfactual RLAIF extends the action interface with `up` and `down` to unify it with the agentic counterfactual refinement interface and support real-world adaptation to workspaces at different heights. The agent proposes bounded corrections, of which only those that improve downstream manipulation under execution-based verification are retained as training targets. The resulting model supports the complete nine-action interface. However, `up` and `down` are disabled in all reported simulation evaluations, including the training-pipeline ablations. RLAIF is therefore evaluated using the same seven-action subset as multi-task co-tuning and RFT. Its gains in these simulation ablations do not arise from executing additional height-adjustment actions at evaluation time.

Prompt format. For the Qwen3-VL-based NAV4MANIP-N, we use the following fixed system prompt: "You are a wheeled indoor navigation robot. Follow the instruction using the observation history and current image." The user message contains the task instruction and requests actions from the complete nine-action set described above. Each observation is preceded by a textual timestamp; for observations in the recent and middle tiers, the actions executed after that observation are appended as additional context. The ten learned queries are inserted immediately after the assistant prefix, and the outer conversation markers follow the Qwen3-VL chat template.

B.3 AGENTIC NAVIGATION–MANIPULATION LOOP

Agent Loop and Prompt Interface. Figure 7 details how the agent $\mathcal{C}$ produces the counterfactual refinement $y_i^+ = \mathcal{C}(x_i)$ used in Section 3.5. For task $(\mathcal{E}_i, l_i)$, the factual context x_i contains the task instruction, factual navigation decision, execution outcome, and updated environment state. Within local refinement cycle t, the agent receives nearby manipulation-affordance predictions

$$\mathcal{A}_{i,t} = \{(b_{i,t,k}, \hat{q}_{i,t,k})\}_{k=1}^{N_{i,t}}, \qquad b_{i,t,k} = (p^x_{i,t,k}, p^y_{i,t,k}, \theta_{i,t,k}). \tag{19}$$

Each pair contains a candidate base pose and its predicted manipulation-success score. We denote the current base pose by $b_{i,t}$ and its predicted score by $\hat{q}_{i,t} = \hat{q}(b_{i,t})$. The rollout-derived empirical affordance target q predicted by $\hat{q}$ is defined in Appendix D.2, Eq. 34. The current-pose score supports the manipulation-readiness decision, while $\mathcal{A}_{i,t}$ provides nearby alternatives for local endpoint reasoning. Hats identify online predictions and distinguish them from the rollout-derived empirical scores used to compute r_i^- and r_i^+.

The local environment state is

$$S_{i,t} = (O_{i,t}, s_{i,t}^{\text{nav}}, s_{i,t}^{\text{manip}}, H_{i,t}), \qquad O_{i,t} = I_{i,t}^{H}, \tag{20}$$

where $O_{i,t}$ contains only the current head-camera observation; $s_{i,t}^{\text{nav}}$ is the navigation state; $s_{i,t}^{\text{manip}}$ summarizes the arm, dexterous hand, and manipulation state; and $H_{i,t}$ contains the relevant factual interaction history. An external YOLO26n detector (Jocher et al., 2026) processes the head-camera image and returns structured detections

$$D_{i,t} = g_{\text{det}}(I_{i,t}^{H}) = \{(c_{i,t,j}, B_{i,t,j}, \rho_{i,t,j})\}_{j=1}^{M_{i,t}}, \tag{21}$$

where $c_{i,t,j}$, $B_{i,t,j}$, and $\rho_{i,t,j}$ denote the class, bounding box, and confidence of detection j, respectively. The complete navigation action space is formatted over multiple lines for readability:

$$\begin{aligned}\mathcal{U} = \{&\texttt{forward}, \texttt{turn_left}, \texttt{turn_right},\\ &\texttt{move_left}, \texttt{move_right}, \texttt{up},\\ &\texttt{down}, \texttt{pause}, \texttt{stop}\}.\end{aligned} \tag{22}$$

Given these inputs, Agentic Refinement proposes a local correction and, once the endpoint is accepted, provides the manipulation primitive and visual–textual context required by NAV4MANIP-M. Collision Handling supplies the YOLO26n results to the agent, which checks whether the proposed motion is safe to execute.

Agentic Refinement. Agentic Refinement implements the agent $\mathcal{C}$ from Section 3.5. At local cycle t, it integrates the factual context, task instruction, affordance predictions, environment state, and interaction history to select one bounded correction,

$$\tilde{u}_{i,t} = f_{\text{ref}}(l_i, \mathcal{A}_{i,t}, S_{i,t}), \qquad \tilde{u}_{i,t} \in \mathcal{U}_{\text{ref}}, \tag{23}$$

where f_{ref} denotes the inference function of the refinement agent $\mathcal{C}$, instantiated by the fixed prompt below. It maps the task instruction, candidate-pose affordances, and current environment state to one bounded refinement action. The refinement action space is

$$\mathcal{U}_{\text{ref}} = \mathcal{U} \setminus \{\texttt{pause}\}. \tag{24}$$

Thus, `pause` remains in the complete action space but is reserved for the safety override; Agentic Refinement proposes $\tilde{u}_{i,t}$ from the other eight actions.

Collision Handling. Collision Handling does not ask the agent to detect objects directly from pixels. Instead, the agent receives the YOLO26n detections $D_{i,t}$ and analyzes their classes, locations, and confidence together with the proposed refinement action and current state:

$$c_{i,t}^{\text{safe}} = f_{\text{col}}(D_{i,t}, \tilde{u}_{i,t}, S_{i,t}), \qquad c_{i,t}^{\text{safe}} \in \{\texttt{PAUSE}, \texttt{CONTINUE}\}. \tag{25}$$

The safety decision produces the action sent to the robot controller,

$$u_{i,t} = \begin{cases} \texttt{pause}, & c_{i,t}^{\text{safe}} = \texttt{PAUSE},\\ \tilde{u}_{i,t}, & c_{i,t}^{\text{safe}} = \texttt{CONTINUE}. \end{cases} \tag{26}$$

This detector-assisted mechanism is an independent physical-motion gate: it does not alter the manipulation-valid handoff criterion or modify the outputs of NAV4MANIP-N.

In each local cycle, the robot executes at most one bounded correction before acquiring a fresh state,

$$S_{i,t} \xrightarrow{f_{\text{ref}}} \tilde{u}_{i,t} \xrightarrow{f_{\text{col}}(D_{i,t})} u_{i,t} \xrightarrow{\text{execute}} S_{i,t+1}. \tag{27}$$

The agent then reevaluates the current-pose manipulation affordance from the updated observations and robot state. This receding-horizon procedure conditions every correction on factual execution

feedback instead of generating an open-loop motion sequence. The existing `stop` action accepts the current endpoint and triggers the NAV4MANIP-M handoff, so no additional acceptance token is required. The executed corrections and final accepted endpoint together define the refined navigation decision $y_i^+ = \mathcal{C}(x_i)$.

At each local cycle, the prompt receives the following structured view of x_i:

$$(l_i, \mathcal{A}_{i,t}, s_{i,t}^{\text{nav}}, s_{i,t}^{\text{manip}}, O_{i,t}, D_{i,t}, H_{i,t}) \tag{28}$$

and returns one action from $\mathcal{U}_{\text{ref}}$. Collision Handling may replace this proposal with `pause`; a replaced proposal is not recorded as an executed refinement target. Otherwise, the action is executed and the newly observed state is fed into the next cycle.

Once the endpoint is accepted, the agent combines the navigator representation, current affordance, EQA answer, and environment state to select a manipulation primitive and construct the visual–textual context for NAV4MANIP-M. During counterfactual RLAIF collection, the simulator is then reset to the same task initialization $\mathcal{E}_i$, and y_i^+ is evaluated with the frozen NAV4MANIP-M $\mathcal{W}$ to obtain r_i^+. The factual rollout supplies r_i^-. The pair (x_i, y_i^+) is retained only when $r_i^+ - r_i^- \geq \tau_\Delta$, exactly matching Eq. 7. Thus, bounded endpoint correction is the inner loop of $\mathcal{C}$, while same-initialization replay and execution-based filtering form the outer RLAIF loop.

Agent Prompt. The following fixed prompt is used by the code agent to perform counterfactual endpoint refinement during RLAIF data collection. It instructs the agent to interpret execution evidence, propose one bounded navigation correction at a time, prepare the NAV4MANIP-M handoff after endpoint acceptance, and return the resulting state to the refinement loop. We use GPT Astra with low reasoning effort as the code-agent backend.

```
You are the Agent for Nav4Manip, an embodied dexterous mobile-
    manipulation system. Your task is to coordinate navigation,
    endpoint refinement, and downstream manipulation using the task
    instruction, manipulation affordance, current head-camera
    observation, robot state, and interaction feedback.

## CURRENT TASK
{task_instruction}

## INPUT
- **Visual Observation**: One current RGB image from the head camera
- **Object Detections**: YOLO26n class labels, bounding boxes, and
    confidence scores for the current head-camera image
- **Affordance Information**: Current-pose manipulation affordance
    and nearby candidate base poses represented as `(x, y, yaw,
    affordance_score)`
- **Navigation State**: Current navigation status and navigation-
    related state
- **Manipulation State**: Current arm, dexterous hand, object-
    interaction, and manipulation-phase state
- **Navigator Context**: Current task-relevant outputs from the
    navigation model
- **Execution History**: Selected navigation and manipulation history
     from previous interaction cycles
- **Execution State**: Latest manipulation outcome and updated
    environment state, when available

## CORE RESPONSIBILITIES
1. **COLLISION HANDLING**: Analyze the YOLO26n detection results
    together with the proposed action and robot state; when they
    indicate a collision or person-safety risk, replace the proposed
    refinement action with `pause`
2. **ASSESS MANIPULATION READINESS**: Determine whether the current
    base configuration provides a suitable handoff for downstream
```

```
    manipulation using the task instruction, affordance, head-camera
    observation, and robot state
3. **REFINE ENDPOINT**: If local adjustment is required, select
    exactly one refinement action according to the current
    observation and state
4. **PREPARE NAV4MANIP-M HANDOFF**: When the endpoint is accepted,
    select a manipulation primitive and construct concise visual-
    textual context for Nav4Manip-M
5. **SELECT EXECUTION FEEDBACK**: After execution, retain navigation-
    relevant history and updated state, especially changes in
    visibility, reachability, object state, robot state, and approach
     feasibility
6. **CLOSED-LOOP UPDATE**: Make each refinement decision from the
    latest observation and state; after an executed correction, wait
    for fresh observations before making the next decision

## OUTPUT FORMAT

### Endpoint Refinement
Return exactly one action from `forward`, `turn_left`, `turn_right`, `
    move_left`, `move_right`, `up`, `down`, or `stop`. Use `stop` to
    accept the current endpoint and trigger the Nav4Manip-M handoff.
    The independent collision handler may replace the selected action
     with `pause` before execution.

### Nav4Manip-M Handoff
{
  "primitive": "<one manipulation primitive from the provided
    primitive set>",
  "context": "<concise task-relevant visual-textual context for
    Nav4Manip-M>"
}

### Navigation Feedback
{
  "history": "<task-relevant execution history>",
  "state": "<updated environment and robot state relevant to
    navigation>"
}

## DECISION RULES
1. **TASK CONDITIONING**: Judge endpoint quality according to
    downstream manipulation executability rather than target
    proximity alone
2. **AFFORDANCE AWARENESS**: Use the current-pose and nearby
    candidate affordance predictions when evaluating the handoff
3. **MINIMAL CORRECTION**: Prefer the smallest bounded correction
    sufficient to improve the current endpoint
4. **FRESH OBSERVATION**: Reassess the endpoint after every physical
    correction using newly observed state
5. **EVIDENCE BASED**: Do not infer visibility, reachability, object
    motion, or task success without support from the provided inputs
6. **MODULE SEPARATION**: Do not replace the navigation policy or
    generate low-level manipulation actions; navigation is handled by
     the navigator and manipulation actions are produced by Nav4Manip-
    M
7. **SAFETY SEPARATION**: Collision and person-safety gating is
    handled by an independent safety mechanism
```

Table 5: Training and rollout configuration of NAV4MANIP-N.

Setting	Value
Model and runtime	
Backbone	Qwen3-VL-8B-Instruct
Trainable components	vision encoder, merger, language model, queries, prediction heads
Initialization	10 special-token embeddings via tokenizer resize; heads via PyTorch defaults
Attention / memory	SDPA; gradient checkpointing
Distributed training	replicated full model; 512 MiB chunked all-reduce
Sequence length	dynamic padding; 262,144-token context ceiling
Visual input	head-camera RGB only (all stages)
Co-tuning	
Hardware	32× RTX PRO 5000 Blackwell GPUs (72 GB each)
Precision / optimizer	bfloat16 / Adafactor
Batch size	1 per GPU ×32 GPUs ×32 accumulation steps = 1,024
Learning rate	1.3×10^{-5} (backbone); 1.0×10^{-4} (prediction heads)
Schedule	160-step warmup; cosine decay to $0.1\eta_{\max}$
Regularization	weight decay = 0; gradient clipping = 1
Training duration	2 epochs
Data / input	4.9M samples (Table 1); head-camera RGB only
Task loss weights	$\lambda_{\text{Afford}} = \lambda_{\text{EQA}} = 1$
Affordance loss weights	dense / candidate / ranking / consistency: $1/1/0.5/0.2$
RFT	
Rollout collection	500 tasks; 8 intermediate states per trajectory
Manipulation verification	16 trials per endpoint
Path penalty	$\lambda_{\text{path}} = 0.10$; denominator floor: 1 m; clipping range: $[0, 1]$
Optimization	batch size 32; learning rates 5×10^{-6} / 1.0×10^{-4}; 1 epoch
RLAIF	
Collection	$K_{\text{iter}} = 2$; 500 tasks per round; GPT Astra agent (low reasoning effort)
Manipulation verification	16 matched trials per endpoint; $\tau_\Delta = 0.25$
Optimization	batch size 32; learning rates 5×10^{-6} / 1.0×10^{-4}; 1 epoch per round

B.4 IMPLEMENTATION DETAILS

We initialize NAV4MANIP-N from Qwen3-VL-8B-Instruct and jointly optimize the vision encoder, multimodal merger, language model, learned queries, and prediction heads. Training uses bfloat16 precision, SDPA attention, gradient checkpointing, and a replicated full model with 512 MiB chunked all-reduce. The ten special-token embeddings are created through the standard tokenizer-resize path, and the dense and candidate heads use the default initialization of their PyTorch linear layers. Samples are dynamically padded within each batch; the 262,144-token backbone limit is a context ceiling rather than a fixed training length.

Co-tuning uses the 4.9M-sample mixture in Table 1. We set $\lambda_{\text{Afford}} = \lambda_{\text{EQA}} = 1$ in Eq. 3; within $\mathcal{L}_{\text{Afford}}$, the dense, candidate, ranking, and consistency weights are $(1, 1, 0.5, 0.2)$. All stages use only monocular RGB observations from the head camera. Table 5 summarizes the remaining settings.

For RFT, we instantiate the path term in Eq. 4 as normalized clipped excess length,

$$\bar{\ell}_i = \text{clip}\left(\frac{L_i - L_i^*}{\max(L_i^*, 1\,\text{m})}, 0, 1\right), \tag{29}$$

so $\lambda_{\text{path}} = 0.10$ contributes at most 0.10 to the ranking score. For both RFT and RLAIF, manipulation scores are empirical success rates over 16 executions. RLAIF evaluates the factual and refined endpoints using matched perturbation seeds and retains a pair when $r_i^+ - r_i^- \geq 0.25$.

C ADDITIONAL EXPERIMENTS

This section supplements the main evaluation with complete navigation results and more detailed analyses of the proposed system. We first report benchmark-wide comparisons, then isolate the effects

Table 6: Closed-loop navigation on R2R and RxR val-unseen. SR, OSR, and SPL are percentages; NE is in meters. Best, second-best, and third-best results per benchmark are bolded, underlined, and double-underlined, respectively.

		R2R				RxR		
Model	Views	SR@3 ↑	SPL@3 ↑	OSR@3 ↑	NE ↓	SR@3 ↑	SPL@3 ↑	NE ↓
NaVid	single	37.40	35.90	49.10	5.47	34.50	23.80	8.41
Uni-NaVid	single	47.00	42.70	53.30	5.58	48.70	40.90	6.24
LookStep	single	49.70	45.30	55.90	5.34	46.90	39.90	6.89
NaVILA	single	54.00	49.00	62.50	5.22	49.30	44.00	6.77
StreamVLN	single	56.90	51.90	64.20	4.98	52.90	46.00	6.22
InternVLA-N1	single	58.20	54.00	63.30	4.83	53.50	46.10	5.91
PROSPECT	single	58.90	54.00	<u>65.20</u>	4.92	54.60	46.20	5.70
MapDream	single	59.80	54.40	64.40	<u>4.59</u>	59.40	49.20	4.96
NavFoM	single	56.20	51.20	64.90	5.01	57.40	49.40	5.51
NavFoM	multi	<u>61.70</u>	<u>55.30</u>	<u>72.10</u>	<u>4.61</u>	<u>64.40</u>	<u>56.20</u>	<u>4.74</u>
Qwen-RobotNav	multi	**65.70**	**59.60**	**72.70**	**4.36**	**73.40**	**63.50**	**4.16**
NAV4MANIP-N	single	<u><u>60.03</u></u>	<u><u>56.79</u></u>	64.11	4.80	<u><u>62.06</u></u>	<u><u>52.60</u></u>	<u><u>4.76</u></u>

of navigation-model architecture and co-tuning data, and finally examine the agentic refinement and RLAIF pipeline through system-level accuracy, latency, and iteration studies.

C.1 FULL NAVIGATION RESULTS

Tables 6 and 7 report the complete numerical results corresponding to Figure 4, covering R2R (Anderson et al., 2018) and RxR (Ku et al., 2020) val-unseen as well as the fine- and coarse-grained tasks on the official VLNVerse test split (Lin et al., 2025). Success Rate (SR@3) is the fraction of episodes whose final position lies within 3 m of the goal, while Oracle Success Rate (OSR@3) counts an episode if any visited position enters that radius. Success weighted by Path Length (SPL@3) combines SR@3 with a penalty for paths longer than the shortest-path reference. Navigation Error (NE) is the final shortest-path distance to the goal in meters, and Trajectory Length (TL) is the executed path length in meters. Higher values are better for SR, OSR, and SPL, whereas lower values are better for NE; TL is reported as a diagnostic without a universal preferred direction. For metrics with an indicated direction, the best and second-best results within each benchmark or task are shown in bold and underlined, respectively; Table 6 additionally double-underlines third-best results. The R2R and RxR comparisons include NaVid (Zhang et al., 2024b), Uni-NaVid (Zhang et al., 2024a), LookStep (Yu et al., 2026a), NaVILA (Cheng et al., 2025), StreamVLN (Wei et al., 2026c), InternVLA-N1 (Wei et al., 2026b), PROSPECT (Fan et al., 2026), MapDream (Lian et al., 2026), NavFoM (Zhang et al., 2026a), and Qwen-RobotNav (Zhang et al., 2026b). For VLNVerse, we compare against Uni-NaVid, InternNav-N1, NavFoM, and Qwen-RobotNav. We also include the test-split Seq2Seq, CMA, HNR, RDP, and GAMA baselines reported in the original VLNVerse study (Sutskever et al., 2014; Wang et al., 2019; 2024; 2025a; Lin et al., 2025). We retain each method's stated single- or multi-view input configuration, as indicated in the tables, and evaluate NAV4MANIP-N under the single-view setting used throughout the paper.

On the VLNVerse test split, NAV4MANIP-N achieves 76.08/71.59 SR@3/SPL@3 on the fine-grained task and 63.17/58.49 on the coarse-grained task. Relative to the strongest prior method in Table 7, Qwen-RobotNav, these results improve SR@3/SPL@3 by 12.33/13.66 points for fine-grained navigation and 16.58/16.95 points for coarse-grained navigation. Among the baselines originally reported with VLNVerse, GAMA gives the strongest fine-grained test result (37.72/33.85), while GAMA and RDP provide the two strongest coarse-grained SR/SPL trade-offs.

C.2 VLN MODEL ABLATIONS

We evaluate three groups of factors affecting the navigation model: architectural components, the co-tuning data composition, and additional fine-tuning with real-to-sim scenes. For the backbone comparison, we replace Qwen3-VL-8B with the smaller Qwen3-VL-4B model (Bai et al., 2025).

Table 7: Closed-loop navigation on the official VLNVerse test split. SR, OSR, and SPL are percentages; NE and TL are in meters. Dashes denote metrics not reported for the test split. Best/second-best results per task are bolded/underlined; TL is not ranked.

		Fine-grained test					Coarse-grained test				
Model	Views	SR@3 ↑	SPL@3 ↑	OSR@3 ↑	NE ↓	TL	SR@3 ↑	SPL@3 ↑	OSR@3 ↑	NE ↓	TL
Seq2Seq	single	30.19	28.15	–	4.79	5.63	27.17	25.53	–	5.00	6.32
CMA	single	28.83	25.54	–	5.24	8.53	31.85	28.67	–	4.72	7.53
HNR	single	32.24	29.03	–	4.93	7.98	34.59	30.45	–	4.62	7.95
RDP	single	36.38	32.29	–	4.60	8.18	36.60	32.21	–	4.65	8.35
GAMA	single	37.72	33.85	–	4.52	5.53	37.20	31.02	–	4.23	3.58
Uni-NaVid	single	45.74	26.91	67.02	4.11	12.35	29.68	13.47	42.70	5.05	10.33
InternNav-N1	single	28.95	25.00	38.69	5.68	9.23	17.51	16.54	23.32	5.71	4.00
NavFoM	multi	51.59	32.40	69.68	4.13	8.58	38.02	23.15	45.93	4.93	6.52
Qwen-RobotNav	multi	63.75	57.93	72.81	3.00	8.16	46.59	41.54	55.97	4.08	7.68
NAV4MANIP-N	single	**76.08**	**71.59**	**82.94**	**2.20**	8.29	**63.17**	**58.49**	**73.66**	**3.05**	8.14

Table 8: Architecture ablations on NAV-N-MANIP. SR@3 and SR@0.6 are navigation success rates at the indicated radii, in percent; NE is in meters. SR@Manip is the single-attempt manipulation success rate, while SR@E2E measures joint navigation and single-attempt manipulation success. Both follow Section 4.1 and are reported as percentages. The full model uses two RLAIF iterations. Best/second-best results are bolded/underlined.

Model variant	SR@3 ↑	SR@0.6 ↑	NE ↓	SR@Manip ↑	SR@E2E ↑
w/o multi-scale history	71.5	39.9	2.41	24.03	18.41
w/o affordance queries and losses	85.5	67.4	1.43	41.28	31.78
Alternative backbone: Qwen3-VL-4B	82.0	68.2	1.61	41.84	28.57
Ours	**89.5**	**71.7**	**1.15**	**56.87**	**52.13**

Architectural Components. Table 8 shows that removing multi-scale history lowers SR@0.6 from 71.7% to 39.9% and SR@E2E from 52.13% to 18.41%, consistent with temporal context supporting precise arrival and task completion. The smaller Qwen3-VL-4B backbone remains competitive in precise stopping but trails the full model in end-to-end success (28.57% vs. 52.13%). Removing affordance queries and losses lowers SR@3 from 89.5% to 85.5%, SR@0.6 to 67.4%, and SR@E2E to 31.78%. These results associate affordance supervision with improvements in both navigation accuracy and downstream success.

Table 9: Ablation of co-tuning data sources for the Nav4Manip-N navigation module. The architecture is kept fixed, and the comparison focuses on the effects of training-data composition on Nav4Manip-N. Each ablated variant removes one source from the full training mixture; sampling ratios are not separately varied. VLNVerse results use the official test split. Success metrics are percentages; NE is in meters. Best/second-best results per task are bolded/underlined.

	R2R			VLNVerse (fine test)			VLNVerse (coarse test)			NAV-N-MANIP	
Model variant	SR@3 ↑	SPL@3 ↑	NE ↓	SR@3 ↑	SPL@3 ↑	NE ↓	SR@3 ↑	SPL@3 ↑	NE ↓	SR@Manip ↑	SR@E2E ↑
w/o Nav4Manip-Pairs	49.8	47.1	5.69	71.7	70.6	2.52	58.1	56.3	3.32	7.56	2.89
w/o Matterport 3D data	17.4	16.3	9.11	69.4	67.5	2.66	57.7	56.0	3.31	37.40	27.33
w/o EQA	49.8	47.3	5.89	70.7	69.3	2.57	59.6	57.4	3.26	36.43	26.55
NAV4MANIP-N	**60.0**	**56.8**	**4.80**	**76.1**	**71.6**	**2.20**	**63.2**	**58.5**	**3.05**	**38.18**	**31.79**

Training Data Sources. Table 9 shows different effects across tasks. Removing Matterport 3D data most strongly affects R2R navigation: SR@3 falls from 60.0% to 17.4%, while NE rises from 4.80 m to 9.11 m. Removing Nav4Manip-Pairs most strongly affects the manipulation handoff, lowering SR@Manip/SR@E2E from 38.18%/31.79% to 7.56%/2.89%. Removing EQA also lowers performance, including SR@E2E to 26.55%. Thus, component effects depend on evaluation task: Matterport removal most affects R2R, while Nav4Manip-Pairs removal is most consequential for manipulation and end-to-end success. The leave-one-source-out results do not identify an optimal mixture ratio.

Table 10: Effect of fine-tuning on real-to-sim environments.

Setting	SR@3 ↑	SPL@3 ↑	OSR@3 ↑	SR@Manip ↑
w/o real-to-sim fine-tuning	<u>19.17</u>	<u>19.14</u>	<u>30.83</u>	<u>0.83</u>
with real-to-sim fine-tuning	**72.50**	**72.10**	**82.50**	**55.83**

Real-to-Sim Fine-Tuning. Table 10 evaluates the effect of additional fine-tuning on real-to-sim environments. SR@3 rises from 19.17% to 72.50%, and SR@Manip from 0.83% to 55.83%, accompanied by improvements in SPL@3 and OSR@3. These gains show that adaptation to the evaluated environments improves both navigation reliability and the suitability of the resulting endpoints for manipulation.

C.2.1 Diagnostic Analyses

The following diagnostics examine navigation failures before the manipulation handoff using the diagnostic-study checkpoint.

Table 11: Navigation diagnostics on R2R val-unseen. SR@3 and SPL@3 are reported in percent. Dashes denote unreported metrics; θ is the magnitude of the required initial rotation. Best/second-best results within each comparison group are bolded/underlined.

Setting	SR@3 ↑	SPL@3 ↑
Instruction		
Matched	**60.03**	–
Mismatched	<u>19.36</u>	–
Initial rotation		
$0^\circ \leq \theta < 30^\circ$	**73.00**	–
$30^\circ \leq \theta < 90^\circ$	<u>58.39</u>	–
$\theta \geq 90^\circ$	24.11	–
History length		
88 frames	**60.03**	–
5 frames	<u>49.16</u>	–
Stopping rule		
$p_{\text{stop}} \geq 0.01$	<u>60.03</u>	**56.79**
Argmax	**60.20**	<u>54.11</u>

Vision and language. Table 11 shows that instruction mismatch reduces SR@3 from 60.03% to 19.36% over 1,839 episodes. In a 200-episode intervention subset, removing or altering route content lowers SR by 36.8–51.5 points; changing the target name lowers it by 29.8 points, compared with 13.9 points when the target sentence is deleted. Behavior diverges earlier for empty or mismatched instructions than for target-sentence deletion, suggesting that route language affects decisions early while target semantics matter nearer termination. Across all five tested models, action logits are more sensitive to visual than language perturbations. Together, these results suggest complementary roles for language in route specification and vision in local action selection.

Endpoint and initial orientation. Table 11 shows lower SR@3 as the required initial turn grows, falling from 73.00% below 30° to 24.11% at or above 90°. A separate data audit found that 38.7% of action samples came from generators whose initial camera is nearly always route-aligned, a possible contributor to a `forward`-first bias. The STOP threshold and argmax rules have similar SR@3 (60.03% and 60.20%), while the threshold yields higher SPL@3 (56.79% vs. 54.11%). On a separate frozen 8B checkpoint, tightening the endpoint radius from 3.0 m to 0.6 m reduces R2R/RxR SR from 5.98%/7.71% to 0.49%/0.38%. These diagnostics point to distinct orientation and termination challenges; success at a coarse radius does not establish a manipulation-ready endpoint.

History. Table 11 shows that reducing history from 88 to 5 frames lowers SR@3 from 60.03% to 49.16%. Extending history beyond 88 frames yields no measurable improvement in this diagnostic.

The results support retaining temporal context up to the tested horizon, with diminishing returns beyond it.

Failure accumulation and recovery. Among failed trajectories, 25.6% deviate by more than 3 m from the reference corridor and 20.0% terminate before reaching the goal; none exhausts the step budget. The nearly constant per-meter survival probability of 0.917–0.939 is consistent with gradual error accumulation. This pattern motivates recovery trajectories from off-demonstration states, navigation and manipulation verification, and execution-conditioned endpoint refinement. Table 3 evaluates the associated training stages and their end-to-end effect.

C.3 System Ablation Study

Alternating optimization. The alternating-optimization baseline in Table 3 performs two rounds of sequential navigation and manipulation training. Each round first optimizes the navigation policy and then the manipulation policy, yielding the sequence $N_1 \rightarrow M_1 \rightarrow N_2 \rightarrow M_2$ and four module-training stages in total. Each stage uses the same number of training samples as an RLAIF training stage. We evaluate the navigation and manipulation policies obtained after the second round. In comparison, two-round RLAIF performs two navigation-policy training stages while keeping the manipulation policy fixed. Alternating optimization achieves 68.4% SR@0.6, 41.47% SR@Manip, and 34.30% SR@E2E, compared with 71.7%, 56.87%, and 52.13%, respectively, for RLAIF. Thus, RLAIF achieves higher task success with fewer module-training stages under matched per-stage sample counts.

Agent-loop comparison in simulation. Table 12 compares four refinement-agent backends over the same ten simulated episodes while holding the surrounding Nav4Manip loop fixed. GPT-6 Astra achieves the highest SR@0.6 (7/10) and the lowest mean navigation latency (61 s per episode). GPT-5.6 Terra instead gives the lowest NE (2.22 m) and the highest manipulation success (4/10), while Terra and Sol tie for the second-highest SR@0.6 (6/10). Thus, no backend dominates every metric: Astra provides the best precise-navigation success and latency, whereas Terra produces the most manipulation-ready endpoints in this small-scale comparison. Here the GPT model serves only as the counterfactual-refinement component; this setting therefore differs from the direct-control baseline in Table 13.

Table 12: Refinement-agent comparison within the fixed Nav4Manip loop in simulation, using ten episodes per backend. NE in meters, and mean navigation latency per episode in seconds. Best/second-best results are bolded/underlined.

Agent	SR@0.6 ↑	NE ↓	SR@Manip ↑	Mean nav. latency (s) ↓
GPT-5.6 Luna	5/10	2.39	2/10	78
GPT-5.6 Terra	<u>6/10</u>	**2.22**	**4/10**	<u>70</u>
GPT-5.6 Sol	<u>6/10</u>	2.29	<u>3/10</u>	73
GPT-6 Astra	**7/10**	<u>2.25</u>	2/10	**61**

Direct navigation agents. We evaluate GPT-6 Astra, GPT-5.6 Sol, GPT-5.6 Terra, and GPT-5.6 Luna as direct closed-loop navigation agents on ten Nav-N-Manip episodes (Table 13). All four models receive the same natural-language instruction, head-camera RGB observations, collision feedback, action space, and action budget; only the model backend changes. At each decision, a model generates one to six atomic navigation actions. These execute sequentially, and an updated observation is returned after the sequence. No global maps, agent poses, goal coordinates, reference paths, affordance annotations, or online evaluation metrics are exposed. The prompt box under *Shared prompt for direct navigation* provides the complete prompt used by all four GPT baselines, including the observation interface, action definitions, action budget, stopping rules, and required output format. This action-sequence interface applies to the GPT baselines; Nav4Manip-N uses the single-action replanning interface described in the main text.

Table 13: Direct-agent comparison on ten NAV-N-MANIP episodes.

Method	SR@3.0 ↑	SR@1.5 ↑	SR@1.0 ↑	SR@0.6 ↑	NE ↓	SR@Manip ↑	Mean nav. latency (s) ↓
GPT-5.6 Luna	1/10	0/10	0/10	0/10	7.021	0/10	<u>123</u>
GPT-5.6 Terra	6/10	4/10	0/10	0/10	3.011	0/10	848
GPT-5.6 Sol	7/10	5/10	1/10	0/10	2.281	0/10	614
GPT-6 Astra	**10/10**	**10/10**	**9/10**	<u>6/10</u>	**0.573**	<u>5/10</u>	445
NAV4MANIP	<u>8/10</u>	<u>8/10</u>	<u>8/10</u>	**8/10**	<u>0.972</u>	**7/10**	**14**

GPT-6 Astra reaches all ten goals within 1.5 m, nine within 1.0 m, and six within 0.6 m, yielding five manipulation successes. It also has a lower mean NE than our system (0.573 versus 0.972 m). Our system reaches eight goals at every tested radius and records seven manipulation successes, with mean navigation latency of 14 s per episode compared with 123–848 s for the direct GPT agents. These episode-level latencies reflect both backend response time and the number of closed-loop model-query rounds: less effective agents may be polled more times before stopping, so the comparison should not be interpreted as measuring per-query model latency alone.

Inference-time refinement versus RLAIF. Table 14 complements the accuracy comparison in Table 3. ACR invokes the counterfactual refinement agent at inference time, whereas RLAIF learns from execution-validated corrections during training. RLAIF reduces mean navigation latency per episode from 50.33 s to 13.80 s, a 72.6% reduction, while improving end-to-end success from 47.10% to 52.13%. Its SR@0.6 also improves from 64.3% to 71.7%. These results support distilling refinement behavior into the navigation policy when navigation latency matters.

Table 14: Mean navigation latency per episode for ACR and RLAIF, averaged over the evaluation episodes and reported in seconds. Corresponding accuracy is reported in Table 3. Best/second-best results are bolded/underlined.

Method	Mean nav. latency (s) ↓
ACR	<u>50.33</u>
RLAIF	**13.80**

Number of RLAIF rounds. Table 15 shows that a second RLAIF round improves every reported metric. Increasing K_{iter} from 1 to 2 raises SR@0.6 from 70.0% to 71.7% and reduces NE from 1.43 m to 1.15 m. The same update increases SR@Manip by 4.35 points and SR@E2E by 4.84 points, indicating that the additional round improves both endpoint precision and the downstream navigation–manipulation handoff.

Table 15: RLAIF iteration ablation on NAV-N-MANIP.

K_{iter}	SR@3.0 ↑	SR@1.5 ↑	SR@1.0 ↑	SR@0.6 ↑	NE ↓	SR@Manip ↑	SR@E2E ↑
1	<u>84.1</u>	<u>79.7</u>	<u>78.1</u>	<u>70.0</u>	<u>1.43</u>	<u>52.52</u>	<u>47.29</u>
2	**89.5**	**84.1**	**80.4**	**71.7**	**1.15**	**56.87**	**52.13**

Shared prompt for direct navigation.

```
You are an embodied visual-navigation agent operating in an unseen
    indoor environment. Your task is to complete the entire
    navigation episode by following the given natural-language
    instruction.

At each interaction step, you are provided with:
1. A natural-language navigation instruction.
2. The latest egocentric RGB observation from the robot's head camera.

3. The remaining action budget.
```

```
4. Feedback indicating whether the immediately preceding forward
    action was blocked by a collision.

You must navigate using only the instruction, the current head-camera
     RGB observation, and the available interaction history. You must
     not assume access to a global map, the robot's coordinates, the
    target coordinates, a reference trajectory, an affordance map,
    distance-to-goal measurements, simulator metadata, or success
    metrics.

The available atomic actions are:
- FWD: move forward by 0.25 m.
- TL: rotate left by 15 degrees.
- TR: rotate right by 15 degrees.
- STOP: terminate the navigation episode and submit the current pose
    as the final navigation result.

For each decision, output between one and six atomic actions. The
    actions are executed sequentially without receiving a new
    observation between them. Use shorter action sequences when the
    scene is ambiguous, when approaching an obstacle, or when precise
     localization is required. A forward action blocked by a
    collision still consumes one action from the budget.

Follow the landmarks, spatial relations, and directional constraints
    in the navigation instruction in the specified order.
    Continuously compare the current visual observation with the
    instruction, maintain an internal estimate of navigation progress,
     and revise the route when the observed scene is inconsistent
    with the expected route. If a collision occurs, reorient and
    select a visibly traversable alternative rather than repeatedly
    attempting the same movement.

Issue STOP only when you judge that the instructed destination has
    been reached and that the robot is positioned appropriately for
    the subsequent manipulation task. STOP is irreversible and, if
    included in an action sequence, must be the final action. Do not
    continue exploring after the destination has been confidently
    reached.

The episode has a maximum budget of 500 atomic actions. Complete the
    navigation autonomously and do not request privileged
    environmental information.

Return only a comma-separated sequence of valid action tokens, for
    example:
FWD,FWD,TL
or:
TR,FWD,STOP
```

D NAV4MANIP-PAIRS DATA GENERATION PIPELINE

This section details the construction of each synchronized Nav4Manip-Pairs record. Section D.1 defines the paired-data representation and artifact conventions. Section D.2 estimates candidate-pose affordances through physics-validated manipulation rollouts, and Section D.3 constructs collision-audited navigation trajectories to the selected position and approach yaw. Section D.4 aligns navigation and manipulation observations through cross-simulator replay and domain randomization. Sections D.5 and D.6 describe language annotation, validation, and batch-level quality control. Together, these stages provide supervision for navigation actions, endpoint-affordance prediction, and manipulation-aware handoff evaluation.

D.1 NAV4MANIP-PAIRS DATA ARCHITECTURE

For task initialization $\mathcal{E}_i$ and a fixed offline manipulation verifier π_v, we define the utility of terminal base pose b and the corresponding manipulation-ready goal region as

$$\begin{aligned} q(b;\mathcal{E}_i) &= \Pr[\pi_v \text{ succeeds} \mid b, \mathcal{E}_i], \\ G_m(\mathcal{E}_i;\eta_{\text{aff}}) &= \{b \in \mathcal{B}_{\text{free}} : q(b;\mathcal{E}_i) \geq \eta_{\text{aff}}\}. \end{aligned} \tag{30}$$

Here, $\mathcal{B}_{\text{free}}$ is the collision-free base-pose space and η_{aff} is the affordance threshold. We call an endpoint manipulation-ready when it lies in $G_m(\mathcal{E}_i;\eta_{\text{aff}})$; the subsequent physical rollout determines whether an individual manipulation attempt succeeds. Thus, endpoint quality depends on downstream manipulation feasibility rather than proximity alone.

For candidate endpoint j in task i, let $b_{i,j} = (p^x_{i,j}, p^y_{i,j}, \theta_{i,j})$ be its navigation-valid base pose. We group its navigation and manipulation fields as

$$\begin{aligned} \mathcal{N}_{i,j} &= (\mathcal{E}_i, b_{i,j}, P_{i,j}), \\ \mathcal{M}_{i,j} &= \left(q_{i,j}, \{\tau^m_{i,j,k}\}_{k=1}^{K_{i,j}}\right), \\ \mathcal{R}_{i,j} &= (\mathcal{N}_{i,j}, \mathcal{M}_{i,j}). \end{aligned} \tag{31}$$

Here, $\mathcal{N}_{i,j}$ is the candidate navigation record and $\mathcal{M}_{i,j}$ is its manipulation record. The empirical affordance $q_{i,j}$ is defined from repeated manipulation outcomes by Eq. 34 in Section D.2. Here, $K_{i,j}$ is the number of stored full trajectories. Initial candidate evaluation saves outcome summaries and references to evaluation shards, but no full trajectories. After selecting endpoint j^*_i, a separate recording run saves successful manipulation trajectories. Accordingly, an accepted rendered episode satisfies $K_{i,j^*_i} \geq 1$ and $K_{i,j} = 0$ for $j \neq j^*_i$. Other positive and negative candidates remain available through their scores, outcome summaries, and navigation records.

Episode assembly copies the NPZ file containing the successful manipulation trajectories recorded at endpoint j^*_i. The renderer selects one trace τ^m_{i,j^*_i,k^*} using its stored trajectory index and offset range and records this choice in the rendering metadata. Let $\tau^n_i = (o^n_{i,1:T_i}, a^n_{i,1:T_i})$ denote the rendered navigation trajectory. The expanded selection and assembly relation is

$$\begin{aligned} \{\mathcal{R}_{i,j}\}_j &\xrightarrow{\text{select } j^*_i} \mathcal{R}_{i,j^*_i} \xrightarrow{\text{assemble with } l_i,\tau^n_i} \mathcal{D}_i, \\ \mathcal{N}_i &= (\mathcal{E}_i, b_{i,j^*_i}, P_{i,j^*_i}, \tau^n_i), \\ \mathcal{M}_i &= \left(q_{i,j^*_i}, \{\tau^m_{i,j^*_i,k}\}_{k=1}^{K_{i,j^*_i}}\right), \\ \mathcal{D}_i &= (l_i, \mathcal{N}_i, \mathcal{M}_i). \end{aligned} \tag{32}$$

When $\mathcal{D}_i$ is expanded into step-level training examples, we use the policy context notation from Section 3.2. Suppressing the episode index on h_t for clarity, its contents are

$$h_t = (L_i, Q, V_{i,t}, A^{\text{hist}}_{i,t}). \tag{33}$$

Here, L_i denotes the tokens of instruction l_i, Q the learned query tokens, $V_{i,t}$ the multi-scale visual tokens constructed from the navigation observations in τ^n_i, and $A^{\text{hist}}_{i,t}$ the executed navigation-action history serialized as described in AppendixB.1. NAV4MANIP-N uses these same input fields during RLAIF training and deployment.

Thus, $\mathcal{D}_i$ is one paired training instance whose navigation part $\mathcal{N}_i$ and manipulation part $\mathcal{M}_i$ are assembled from the selected record $\mathcal{R}_{i,j^*_i}$; the remaining candidates provide affordance-ranking supervision through their stored summaries. This matches Eq. 1, with $q_i \equiv q_{i,j^*_i}$ and $\tau^m_{i,k} \equiv \tau^m_{i,j^*_i,k}$ in the compact notation. We reserve $\mathcal{A}_{i,t}$ for the online affordance-candidate set in Eq. 19. Failed evaluations retain outcome and failure metadata, but their full trajectories are not saved.

Each episode stores the selected demonstration and candidate metadata in the artifact groups summarized in Table 16. A shared endpoint ID links affordance, navigation, grasp, rendering, and language records. Candidate summaries contain scores, attempt and success counts, feasibility, and references to detailed evaluation shards; navigation observations and action labels are stored in the rendering and VLN sidecars. A fixed planar transform maps EmbodiedGen scene coordinates to the robot manipulation frame. Table 17 summarizes the coordinate and trajectory conventions.

Table 16: Nav4Manip-Pairs episode artifacts. Paths are relative to `episode/`; optional files depend on the generation configuration.

Group	File	Contents
Metadata	`episode_meta.json`	Scene and endpoint IDs, frame transforms, and cross-artifact references
`affordance/`	`affordance_map.json`	Candidate scores and evaluation counts
	`eval/*/handoff_meta.json`	Endpoint-specific handoff parameters and shard references
	`object_rates.json`	Discrete object-position success rates
`grasp/`	`sceneinfo_00.json`	Scene, robot, object, and manipulation configuration
	`shard_00.json`	Rollout outcomes and failure metadata
	`shard_00_traj.npz`	Successful manipulation trajectories
`nav/`	`nav_paths.json`	Candidate paths, reachability, and path statistics
	`collision_audit.json`	Path and docking collision checks
	`nav_overlay.png`	Route visualization
	`paths/*.json`	Replayable endpoint-specific drive plans
`render/`	`camera_poses.json`	Frame-indexed camera poses and phase labels
	`episode.mp4`	Aligned navigation–manipulation observations
	`nav.mp4`	Navigation-only observations
`vln/`	`instruction_video.json`	Instruction and deterministic destination grounding
	`prompt_video.txt`	Saved annotation prompt
	`scene_map.txt`	Serialized scene-map context
	`dataset/*.jsonl`	Serialized training examples

Table 17: Coordinate and manipulation-trajectory conventions used in Nav4Manip-Pairs.

Quantity	Convention
Scene/grid	Meters; 0.05 m cells; rows index y and columns index x
Base and joints	Base/parking yaw and joint positions in radians
Manipulation yaw	Selected modified-Dexonomy yaw in degrees (Chen et al., 2025)
Object pose	$[x, y, z, q_w, q_x, q_y, q_z]$
Trajectory	50 Hz; traces are indexed by `offsets` in the NPZ archive
NPZ fields	`robot_qpos`, `robot_cmd`, `obj_pose`, `offsets`, `dt`, `n_arm`, `meta`

Scenes are partitioned before training to define the training and test splits of Nav-N-Manip. The generated-house split contains 182 training houses and 39 held-out test houses. For VLNVerse, we use 176 scenes from its official training set and partition them into 168 training scenes and 8 held-out test scenes for Nav-N-Manip. The held-out partitions from these two sources together constitute the Nav-N-Manip test split used in our simulation evaluation. Training and test scenes are disjoint within each source. This Nav-N-Manip partition is separate from the official VLNVerse test split used for the standalone navigation benchmark.

D.2 Manipulation Affordance Estimation

Candidate endpoints are generated around support surfaces accessible to the robot. Tables, desks, counters, nightstands, workbenches, and vanities with heights in $[0, 0.84]$ m are ranked by semantic name, occupancy, and usable area. The generator examines surfaces in this order until it finds a geometrically feasible, navigation-reachable dock. Along each rectangular surface edge, feasible intervals must accommodate the 0.779×0.769 m chassis, a 0.02 m rectangular margin, and a 0.35 m docking sweep. Sampling three evenly spaced positions per interval yields a geometry-dependent number of candidates.

Each endpoint is passed through a deterministic handoff to the offline verifier π_v, implemented by our modified Dexonomy manipulation generator. The handoff specifies the scene, robot, support surface, object and pose-sampling protocol, endpoint and approach yaw, scene-to-robot transform, and fixed execution parameters. The modified pipeline uses cuRobo for inverse kinematics and motion generation and MuJoCo for physical validation. Candidate evaluation returns evaluation cells, outcomes, attempt and success counts, successful yaws, and failure categories, with full-trajectory recording disabled.

For candidate endpoint j in task i, let $N_{i,j}^{\mathrm{eval}}$ be the number of recorded evaluation cells. Cell c specifies an initial object configuration (o_x, o_y, γ) and target position (f_x, f_y). The generator screens eight placement yaws, $\{0^\circ, 45^\circ, \ldots, 315^\circ\}$, using inverse kinematics and arm-configuration change, then retains a set $\widehat{\Phi}_{i,j,c}$ for planning and physical validation. Let $Y_{i,j,c,\phi}^{\mathrm{eval}} = 1$ when yaw ϕ completes the validated manipulation sequence, and zero otherwise. The empirical endpoint affordance is

$$\zeta_{i,j,c}^{\mathrm{eval}} = \mathbf{1}\left[\exists \phi \in \widehat{\Phi}_{i,j,c} : Y_{i,j,c,\phi}^{\mathrm{eval}} = 1\right],$$
$$q_{i,j} = \begin{cases} \dfrac{1}{N_{i,j}^{\mathrm{eval}}} \displaystyle\sum_{c=1}^{N_{i,j}^{\mathrm{eval}}} \zeta_{i,j,c}^{\mathrm{eval}}, & N_{i,j}^{\mathrm{eval}} > 0, \\ 0, & N_{i,j}^{\mathrm{eval}} = 0. \end{cases} \tag{34}$$

The described configuration retains one yaw per cell (`KEEP_TOP=1`), so $\zeta_{i,j,c}^{\mathrm{eval}}$ is the binary outcome of the selected yaw. The denominator counts recorded evaluation cells; samples rejected before cell creation are excluded. The explicit zero case applies when a completed evaluation contains no cells; for a nonempty evaluation, the observed success fraction may also be zero. Missing or invalid evaluation shards are excluded. The evaluation count $N_{i,j}^{\mathrm{eval}}$ is distinct from the number $K_{i,j}$ of stored trajectory traces in Eq. 31. The offline affordance $q_{i,j}$ estimates $q(b_{i,j}; \mathcal{E}_i)$ under this sampling procedure.

We use 4,096 seeded initial configurations, stratified into five bins per axis, together with a pool of 64 target positions. For each precise initialization, up to eight feasible targets are selected. Each dock has a budget of 24 precise initializations; filtering makes the realized number of evaluation cells variable. Phase-timing randomization is disabled. Evaluation cells, yaw outcomes, and physical-validation statistics are retained, alongside discrete 5×5 object-position rates in `object_rates.json` when generated. These statistics use no smoothing, interpolation, or learned calibration.

A successful trial must complete approach, lift, transport, placement, release, settling, and retreat. Validation rejects forbidden contacts above 1 N, requires 0.03 m support clearance before removing the low-clearance exemption, and maintains hand–object contact until release. Placement uses approximately 5 mm base clearance; object stability is checked after settling and again after retreat. Failures distinguish planning, forbidden contact, early release, settling, retreat, and retreat-induced displacement.

Among geometrically feasible and navigation-reachable candidates, selection ranks endpoints by decreasing $q_{i,j}$, decreasing successful-cell count, and stable endpoint ID. The selected dock j_i^* is rerun with successful-trajectory recording enabled (`SAVE_TRAJ=1`, `SAVE_FAILS=0`) until the requested number of successes is obtained or its initialization budget is exhausted. Episodes with no recorded success are rejected. The resulting trajectories span approach and grasp through placement, settling, and retreat. They supply the manipulation demonstrations, while the earlier candidate-evaluation records supply the affordance scores.

D.3 Navigation and Trajectory Rollout

Navigation must reach both the manipulation endpoint and its approach direction. The planner rasterizes floor, wall, and furniture meshes at 0.05 m resolution, marks non-floor geometry occupied, and dilates obstacles by the chassis circumradius, 0.548 m. Eight-neighbor A* plans to a pre-dock pose 0.35 m before the parking pose, with no corner cutting, unit axial cost, diagonal cost $\sqrt{2}$, and clearance preference weight 0.5. For grid node v, goal node v_g, and coordinate differences $d_x = |x_v - x_{v_g}|$ and $d_y = |y_v - y_{v_g}|$, the admissible heuristic is

$$h_{\mathrm{A}*}(v, v_g) = \max(d_x, d_y) + (\sqrt{2} - 1)\min(d_x, d_y). \tag{35}$$

The initial pose is sampled from the pre-dock pose's eight-connected free-space component and must lie 1.2–16 m from the pre-dock pose in straight-line distance. The sampler tests at most 500 candidates and retains a pool of up to eight collision-valid paths in the preferred 3–16 m path-length range. The pipeline randomly selects among paths whose absolute target-length error is within 0.75 m of the best error in the pool. If the pool is empty, it falls back to the valid path closest to the target length, subject to any configured hard minimum. Path smoothing uses an additional 0.06 m clearance

margin. Disconnected or collision-invalid endpoints remain explicit and are ineligible as rendered demonstrations.

The grid path is converted into a smoother route by collapsing collinear points, applying forward and reverse string-pulling, performing three collision-checked midpoint-relaxation rounds, and fitting circular fillets. Fillet radii are capped at 0.70 m, with tangent lengths limited to 45% of each adjacent segment; arcs below 0.5 m radius are omitted and retained arcs are sampled every 2.5°. An optional route-jitter stage inserts a via point between 35% and 65% of the route and replans both segments. Segment checks sample points at intervals of approximately 0.4 grid cells. The navigation audit tests each drive pose on the uninflated grid using a chassis footprint inset by 0.05 m to reduce rasterization artifacts and sampled at 0.08 m spacing. Docking is checked separately through swept chassis polygons against furniture using the separating-axis theorem. These checks preserve collision validity through path post-processing and docking.

Each validated candidate path $P_{i,j}$ is resampled at $\Delta t = 0.1$ s with nominal speed 0.6 m/s and turn rate 1.2 rad/s; heading changes of at least 10° between adjacent path segments become explicit pivots. The final 0.35 m docking segment runs at 0.25 m/s and ends at the position and yaw of $b_{i,j}$. No explicit acceleration or jerk limits are imposed. Acceptance requires validated navigation and docking motions, arrival at the pre-dock and parking poses, and navigation–manipulation endpoint agreement within 0.12 m.

D.4 Cross-Simulator Rendering and Domain Randomization

NVIDIA Isaac Sim (NVIDIA, 2026) replays the scene assets together with the deterministic handoff, selected manipulation trajectory, and navigation drive plan produced by the preceding stages. Episode metadata and frame-indexed camera poses align the navigation and manipulation segments and record the manipulation subsampling. This persistent alignment supports waypoint and action serialization without temporal interpolation.

Rendering produces 640×480, 10 Hz videos using path tracing with 32 samples per pixel and H.264/YUV420p encoding at CRF 18. The nominal navigation-camera profile has 90° horizontal field of view, 1.25 m height, -25° pitch, and a 0.42 m forward mount. Because domain randomization may alter the rendered head-camera profile, the applied intrinsics and extrinsics are stored for every episode. Wrist, stereo/right-camera, and third-person outputs are optional.

The rendering interface supports perturbations of camera pitch, height, field of view, forward and lateral mount positions, and principal point, with route- and episode-level controls. When enabled, head-camera height offsets and horizontal fields of view are sampled uniformly from $[-0.10, 0.10]$ m and $[58^\circ, 74^\circ]$, respectively. The navigation-video camera uses separate configurable ranges. The applied camera parameters are recorded for each episode.

D.5 Language Annotation

We use Opus 4.8 to generate the free-form language annotations. For each episode, the annotation interface combines visual evidence with structured context by sampling evenly spaced navigation frames and intermediate manipulation frames, attaching phase and landmark anchors from `camera_poses.json`, and serializing scene geometry, paths, landmarks, endpoints, and handoff metadata. Our configuration provides eight navigation frames and two manipulation frames, resizes images to a maximum dimension of 768 pixels, includes frame anchors, and omits the full scene map. Requests use a 180 s timeout, allow at most three attempts, and apply a backoff of $1.5(\text{attempt}+1)$ s.

Language and geometric supervision are stored separately: `instruction` contains free-form text, while `destination_grounding` is deterministically derived from the selected endpoint, object-relative quantities, affordance metadata, and trajectory references. Our versioned per-episode JSON schema, `vln-episode-1.0`, also records start pose, goal, path, frames, samples, actions, conventions, and warnings.

D.6 Quality Control and Batch Execution

An episode is published only after artifact, geometric, and visual checks pass. The driver requires nonempty `render/episode.mp4`, `render/nav.mp4`, and

`render/camera_poses.json`; validation additionally checks core navigation, grasp, language, and metadata records, cross-artifact references, endpoint alignment within 0.12 m, and an instruction of at least 20 characters. Missing, unreadable, or malformed trajectories cause rejection. Rendering QC verifies that hand geometry and materials are opaque and that the robot remains visible during manipulation. A separate RGB/depth occlusion diagnostic is optional. The default black-frame check samples up to 12 frames and rejects an output when the mean fraction of near-black pixels exceeds 0.25; a pixel is counted as near-black when all three RGB channels are below 12 on the 8-bit intensity scale.

Batch execution tracks scene checking, endpoint evaluation, aggregation, assembly, rendering, annotation, validation, and copying. Endpoint evaluations run as independent subprocesses with round-robin GPU assignment; aggregation waits for their shards. When resumed, the pipeline reuses existing handoffs, summaries, and shards, while stale handoffs or missing selected-dock artifacts trigger re-evaluation. The pipeline retries assembly and rendering once, clears failed renders before retrying, and quarantines episodes that fail final validation. Completion markers and batch manifests support interrupted runs.

E GALLERY

This section qualitatively complements the main experiments by visualizing the diversity and realism of generated environments, representative Nav4Manip-Pairs and rendered EQA examples, complete simulated and real-world navigation–manipulation executions, and the visual evidence underlying manipulation-affordance predictions.

E.1 DIVERSITY AND REALISM OF GENERATED SIMULATION ENVIRONMENTS

To promote diversity in data generation, we vary the scene source and layout, navigation trajectory, object configuration, and grasp strategy. We use 221 furnished house scenes from EmbodiedGen v2 (Wang et al., 2026b), which provide diverse indoor layouts, furniture arrangements, and navigation spaces. Within each scene, we further vary the robot's initial pose, target location, object placement, and navigation trajectory to produce diverse navigation–manipulation sequences. Each manipulation episode contains one target object and 0–4 distractors sampled from a production-ready curated subset of Nav4Manip-Pairs comprising 141 object assets. Grasp poses and trajectories are drawn from a library of 7,941 curated grasp records, further diversifying object interactions and manipulation behaviors. Scene appearance is inherited from the original scene assets and their referenced materials, while manipulation-object textures and materials are loaded from the corresponding source assets. Figure 8 presents representative environments across multiple room categories.

Figure 9 qualitatively compares simulation scenes generated by our method with those produced by a GPT-6 Astra baseline. Across the displayed examples, the baseline produces more repetitive layouts and appearances, whereas our method exhibits greater visual and structural diversity. The complete prompt used for the baseline follows the figure. Its target of 180 rooms specifies the controlled baseline batch and is distinct from the 221 EmbodiedGen v2 house scenes used in Nav4Manip-Pairs.

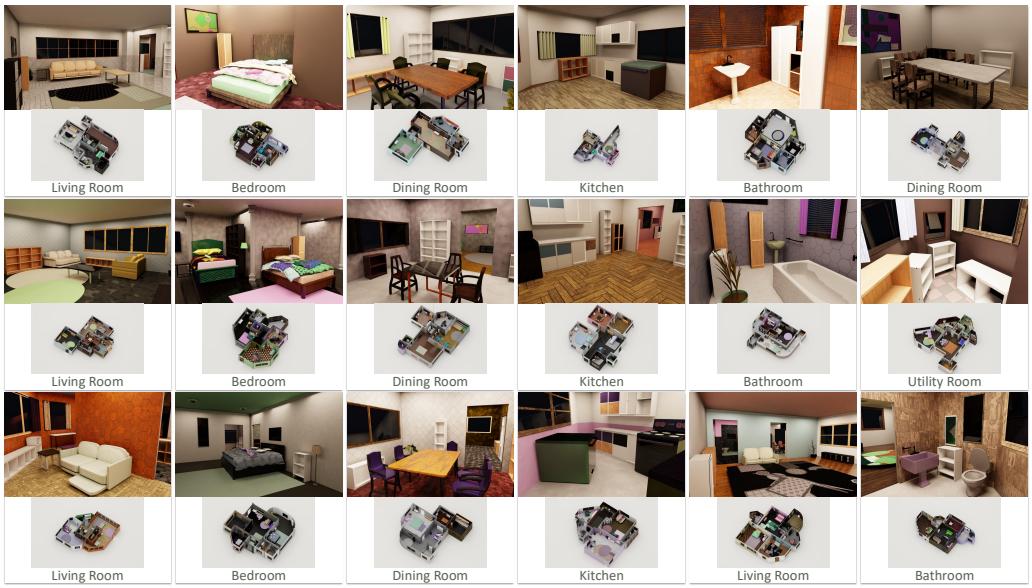

Figure 8: **Diversity of generated simulation environments.** Eighteen representative scenes generated by our pipeline are shown across multiple room categories. Each example pairs an interior rendering with its corresponding 3D scene layout. The gallery demonstrates diversity in room geometry, visual appearance, furniture arrangement, and object placement.



Figure 9: **Qualitative comparison of generated simulation scenes.** Our method (top row) is compared with a GPT-6 Astra baseline (bottom row). Both methods receive the same task information and access to the same permitted EmbodiedGen v2 repository assets, including available 3D geometry, textures, and materials. In these examples, the baseline produces more repetitive layouts and appearances, whereas our method yields greater visual and structural diversity.

Baseline agent prompt.

```
You are a 3D simulation-environment generation agent. Create
    realistic, fully furnished residential rooms that can support
    indoor robot navigation. Produce executable scene assets and
    renders of those assets, together with the information needed to
    reproduce them.

## CURRENT TASK
Generate 180 distinct rooms in a residential setting: 28 living rooms
    , 28 bedrooms, 26 dining rooms, 26 studies, 26 children's rooms,
    26 entrance halls, and 20 bathrooms. Each scene represents a room
     within a house. Include architectural elements and furniture or
```

```
    fixtures appropriate to its function. Prioritize visual realism,
    meaningful layout diversity, and usable navigation space.

## TASK INFORMATION AND AVAILABLE RESOURCES
1. Task requirements: You receive the same task-level information and
     access to the same permitted asset sources as the compared
    method. Generate furnished residential rooms for indoor
    navigation, with recognizable furniture and household fixtures
    and varied layouts. You may use permitted repository assets,
    create new geometry, or adapt existing geometry as needed to meet
     the room requirements.
2. Available asset sources: You have access to the same permitted
    EmbodiedGen v2 repository assets as the compared method,
    including available 3D geometry, collision representations,
    textures, and materials. Inspect the resources and use suitable
    wood, fabric, and marble materials where appropriate. You may
    execute procedural shaders and bake them into texture maps.
    Preserve asset licenses and records of transformations or
    renderer adaptations internally.
3. Unspecified design details: choose reasonable residential room
    dimensions, furniture selections, and arrangements. Record these
    decisions in the scene manifests so they can be reproduced.

## IMPLEMENTATION PARAMETERS FOR REPRODUCING THIS BATCH
Use MuJoCo 3.3.6 and MJCF scene descriptions. Model a circular
    navigation footprint with radius 0.30 m, an additional clearance
    margin of 0.10 m, and robot height 1.4 m. Use occupancy maps at
    0.05 m resolution and produce five navigation episodes per room.
    Render one overview and one interior view per room at 900 x 675
    pixels. These settings document choices made during the baseline
    implementation; they are not evidence of a common evaluation
    configuration across the compared methods.

## CORE RESPONSIBILITIES
1. Architectural plausibility: use residential dimensions and
    coherent room boundaries, entrances, and architectural details.
    Arrange windows and doors sensibly where appropriate, and keep
    entrances accessible.
2. Furniture and fixture quality: furnish each room with recognizable
    , detailed 3D objects, such as upholstered seating, tables, beds,
     shelving, cabinets, lamps, sinks, toilets, and showers as
    appropriate. Use permitted assets where suitable, and create or
    adapt geometry where needed. Ensure visible structural details,
    plausible scale, and proportions. Simple collision proxies may be
     used for physics, but the visible environment must contain
    furnished residential interiors rather than generic obstacle
    primitives.
3. Material appearance: apply supplied, generated, or asset-native
    textures and materials to appropriate surfaces, including
    furniture, floors, walls, and textiles. Use sensible texture
    scale, orientation, UV mapping, and color. Inspect for stretched
    textures, missing maps, and inconsistent materials.
4. Diversity: vary room dimensions, furniture arrangements, object
    selections, circulation patterns, and material combinations.
    Changing only a scene name, random seed, or color does not create
     a meaningfully different room. Check for duplicate layouts.
5. Navigation feasibility: respect the specified robot dimensions and
     clearance requirements. Check furniture placement, intersections,
     free-space connectivity, and start-to-goal reachability. Keep
    navigation paths consistent with the actual collision geometry,
    including collision representations of reused assets.
```

```
6. Iterative validation: load the generated scenes in the simulator,
    render representative views, inspect the results, and correct
    visible or geometric problems before completing the batch.

## OUTPUT FORMAT
Return a self-contained directory containing:
- Exactly 180 scene descriptions in MJCF format, with the geometry
    and material dependencies needed to load them.
- A manifest listing scene IDs, room types, dimensions, furniture,
    random seeds, asset sources, and any geometry or material
    adaptations.
- One overview and one interior render for every scene, produced from
     the delivered 3D assets, plus an HTML gallery for inspection.
- Five navigation episodes per room, with start and goal poses,
    reference paths, and occupancy maps: 900 episodes in total.
- Generation scripts, usage instructions, and a validation report.

## DECISION RULES
1. Follow the task requirements and the shared asset permissions.
    Record implementation assumptions and provenance for reused and
    newly generated assets.
2. Render the delivered 3D scene assets with documented camera,
    lighting, resolution, and material settings. Correct geometry or
    material defects in the scene rather than editing preview images.
3. Distinguish access to an asset source from actual use of
    particular geometry, collision, texture, or material files.
    Record the assets used and any adaptations in internal logs. Do
    not use the compared method's generated scene outputs as inputs.
```

E.2 Rendered Examples of Nav4Manip-Pairs and EQA

This subsection presents rendered examples of navigation, grasping, and embodied question answering (EQA). Nav4Manip-Pairs draws from 259 small-object assets and 221 houses; the generated manipulation episodes use the curated subset of 141 production-ready object assets described above. We also show navigation examples from R2R and RxR, both of which are based on Matterport3D; the full Matterport3D dataset contains 2,056 rooms across 90 building-scale scenes. Figure 10 illustrates variation in spatial layout, visual appearance, target object, support surface, viewpoint, and robot-arm configuration.

Figure 11 presents representative EQA examples covering object color, room-specific object color, counting, distance from the initial position, object existence, and logical reasoning. Each example contains two rendered observations and a corresponding question–answer pair, illustrating the visual contexts and question types represented in the dataset.

E.3 Simulated Dexterous Mobile Manipulation

Figure 12 presents six simulated dexterous mobile-manipulation examples in different indoor scenes. Each sequence begins with navigation toward a target object and proceeds through pre-grasp positioning, grasping, lifting and reorientation, placement, and release. The paired views provide complementary perspectives: the third-person view shows the robot's motion relative to its surroundings, while the first-person view captures hand–object interactions. Representative keyframes are selected to summarize the main stages of each sequence.

E.4 Real-World Dexterous Mobile Manipulation

We further demonstrate the complete system in a real-world dexterous mobile-manipulation scenario. As shown in Figure 13, the robot first navigates through the office environment to the target table and then performs the downstream manipulation sequence.

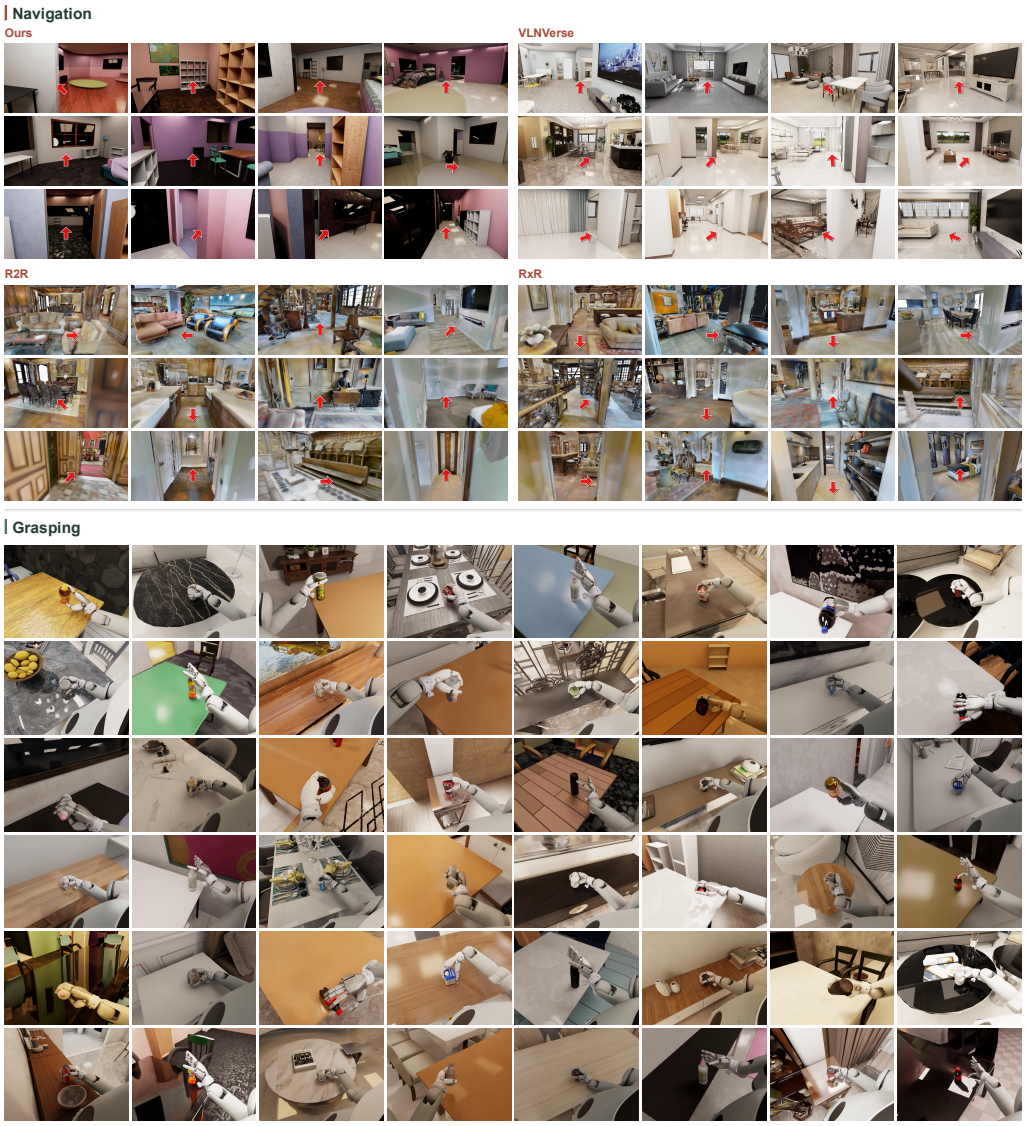

Figure 10: **Rendered navigation and grasping examples in Nav4Manip-Pairs.** The navigation gallery shows representative examples from our generated scenes, VLNVerse, R2R, and RxR, with red arrows indicating movement directions. The grasping gallery presents diverse target objects and support surfaces.

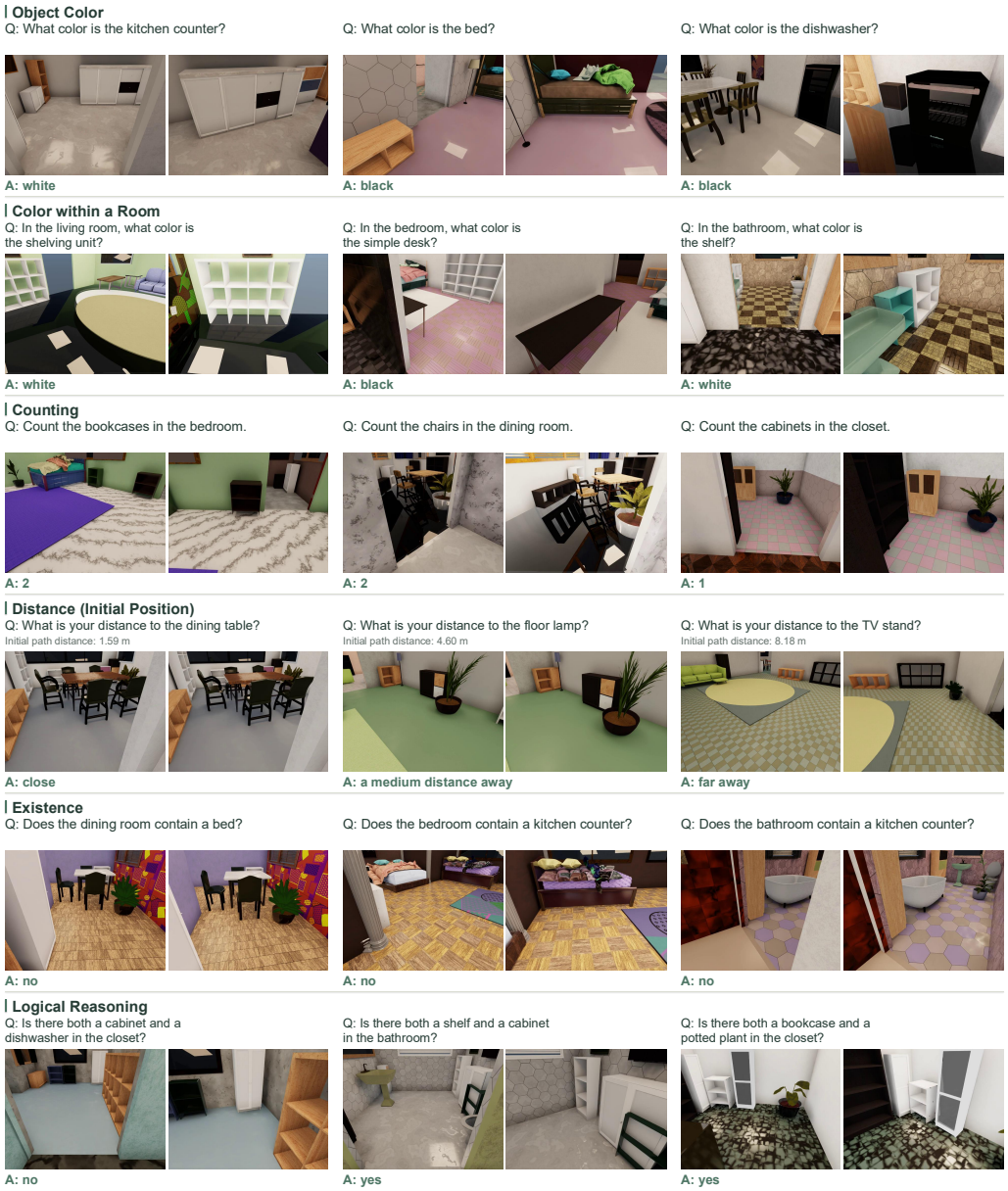

Figure 11: **Rendered examples of embodied question answering (EQA).** Eighteen examples cover six question categories: object color, room-specific object color, counting, distance from the initial position, object existence, and logical reasoning. Each row presents three examples, each with two rendered observations and a corresponding question–answer pair. The distance examples additionally report the initial path distance to the queried object.

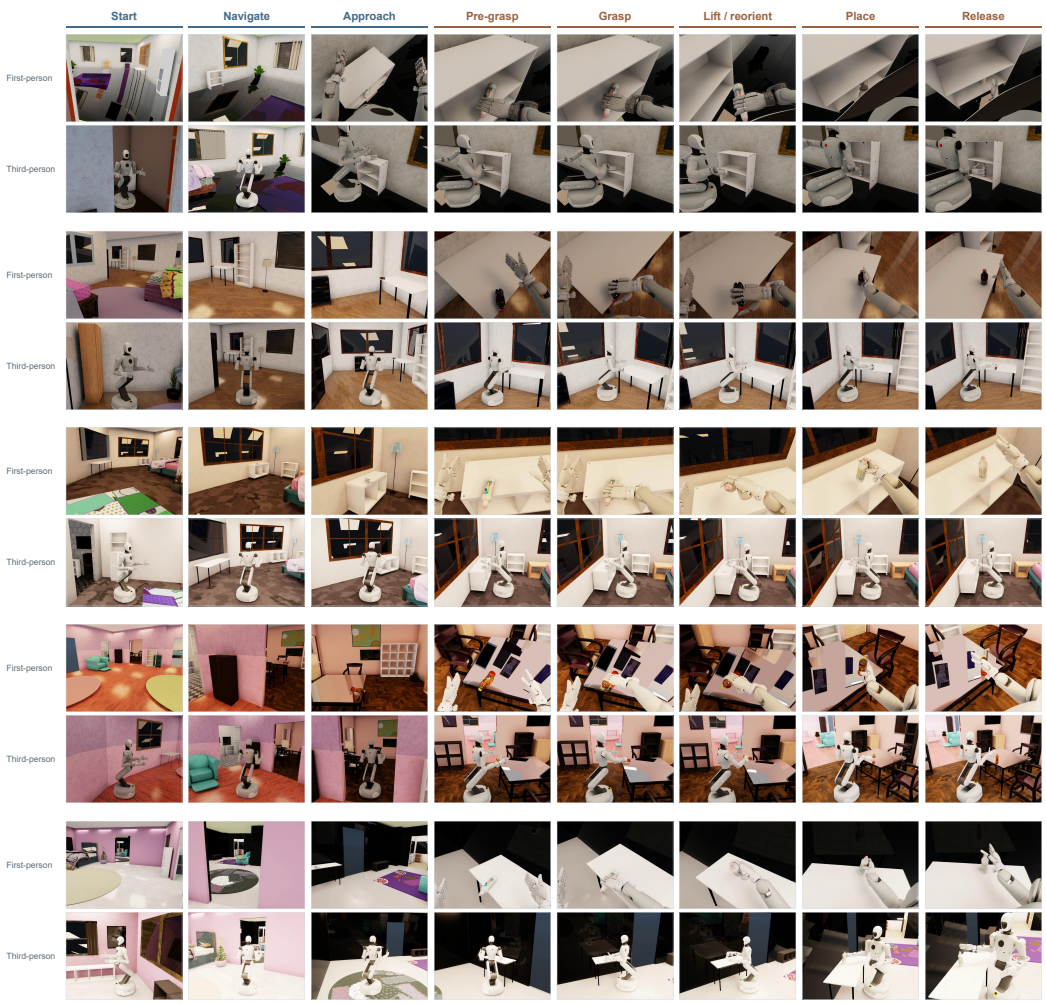

Figure 12: **Simulated dexterous mobile manipulation.** Six examples are shown. Each example occupies two rows, with first-person views above the corresponding third-person views. Eight keyframes per row illustrate the progression from navigation and approach to pre-grasp positioning, grasping, lifting and reorientation, placement, and release. Frames are ordered from left to right.



Figure 13: **Real-world dexterous mobile manipulation demonstration.** The robot first navigates to the target table (frames 1–8) and then completes the object manipulation sequence (frames 9–12). Orange arrows indicate the direction of motion.

E.5 Affordance Visualizations

Our dense manipulation affordance is defined over candidate terminal base positions and orientations in a local bird's-eye-view grid. Each position–yaw cell predicts the likelihood of successful downstream manipulation if the robot stops at that configuration. The map's spatial distribution therefore indicates which direction the robot should move and where it should stop, including the desired terminal yaw, for a manipulation-ready handoff to Nav4Manip-M.

Figure 14 uses occlusion sensitivity to visualize which parts of a real-world head-camera observation influence these dense affordance predictions. Each RGB observation is partitioned into a 4×6 grid, and each region is perturbed in turn using blur occlusion. We measure sensitivity as the mean absolute change from the unperturbed post-sigmoid affordance probabilities, averaged over all 32×32 bird's-eye-view cells and eight yaw bins. The scores are normalized by the maximum within each image, upsampled with bicubic interpolation, and overlaid on the original observation. Warmer colors indicate image regions whose perturbation produces a larger change in the predicted affordance map; they do not directly represent higher affordance values. The examples span corridors, doorways, office workspaces, and lounge areas, illustrating the visual evidence used to infer promising motion directions and manipulation-ready stopping configurations.

F Real-World Deployment Details

F.1 Navigation Deployment

Our real-world navigation stack separates remote model inference from robot-side control. At each decision step, the model service receives the task instruction, the latest head-camera observation, and the history of executed navigation actions, then returns a short action chunk to the robot-side controller. The controller handles camera acquisition, model communication, action scheduling, low-level chassis execution, wheel-odometry feedback, safety arbitration, and logging. The deployment interface streams 640×480 RGB observations at 1 Hz and communicates with the chassis through a protected relay.

Figure 15 shows the robot's camera configuration, which comprises an S56 stereo head camera and an ISX031C camera mounted on each wrist. The figure also presents representative RGB views from both head cameras and both wrist cameras. Consistent with the system design in Appendix B, the navigation policy uses only the designated head-camera RGB stream, whereas Nav4Manip-M uses the wrist-camera streams during manipulation.

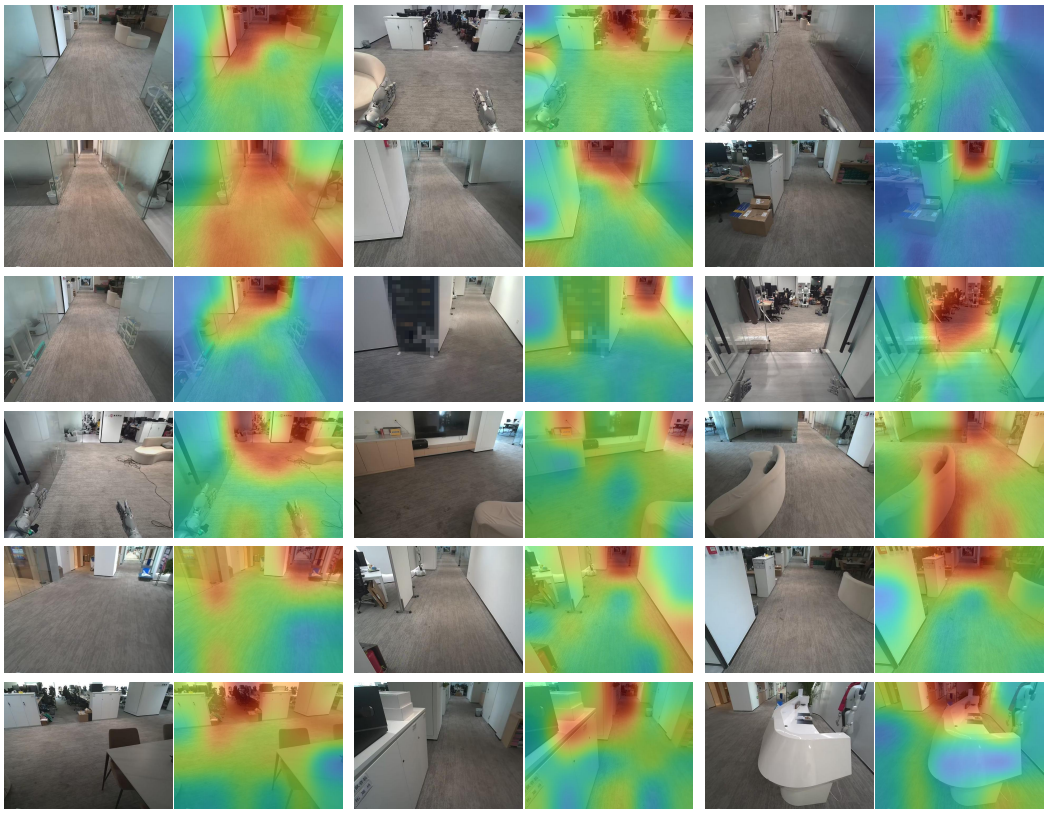

Figure 14: **Occlusion sensitivity of dense manipulation-affordance predictions in real-world scenes.** Each example pairs the original RGB observation (left) with its occlusion-sensitivity overlay (right). Warmer colors highlight image regions that the model relies on more strongly when predicting where the robot should stop. Sensitivity is normalized independently within each example.

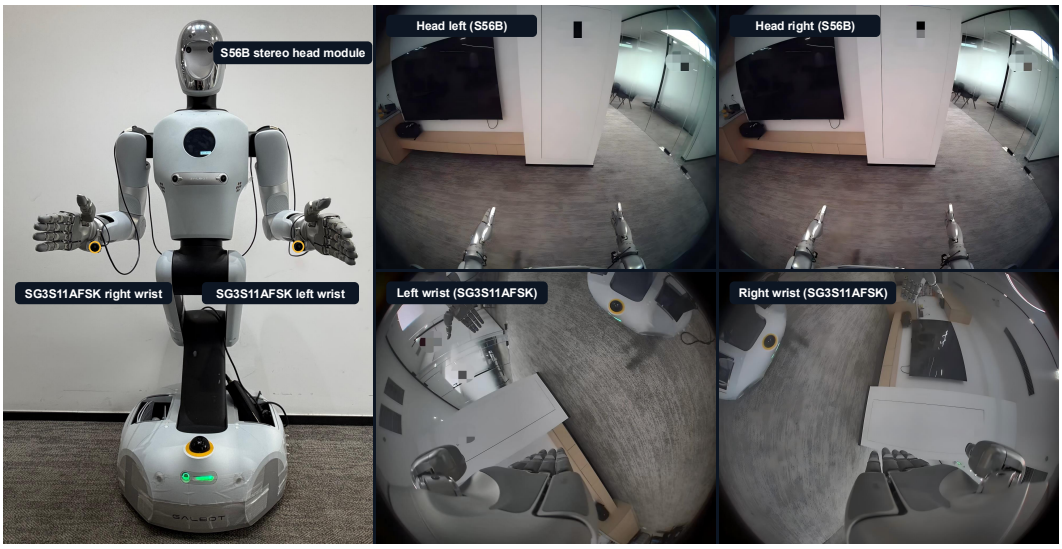

Figure 15: **Real-robot camera configuration and example observations.** Left: camera placement on the robot, including the S56 head stereo camera and two ISX031C wrist cameras. Right: example RGB views from the left and right head cameras (top) and the left and right wrist cameras (bottom).

The navigation policy selects commands from the predefined discrete action space and predicts chunks containing at most nine commands. The robot-side controller executes these chunks with rolling overlap: only the currently committed command is sent to the chassis, while a newer prediction may replace any remaining unexecuted commands. Consequently, the action history contains only commands that were physically executed. Motion commands use fixed translation and rotation increments. The safety gate may replace a proposed motion with `pause` without terminating the episode, whereas `stop` terminates navigation and accepts the current endpoint.

The robot-side controller monitors motion through wheel odometry and verifies the freshness and consistency of sensor observations before issuing each command. Safety interventions take priority over navigation commands and may invalidate pending actions. Once motion is permitted again, navigation resumes from a fresh observation. All physical experiments are conducted under attended operation, with an emergency-stop mechanism available throughout execution.

F.2 Real-Scene Scanning and Data Generation

We scan real-world environments using a Hesai Kosmo scanner (Hesai Technology, 2026) equipped with fisheye cameras and LiDAR. During scanning, the cameras capture RGB observations, while the LiDAR provides sparse geometry and pose-related measurements. A real-time triangular-surface preview allows the operator to monitor scene coverage during acquisition. We reproject the fisheye images into perspective views and convert them, together with the calibrated camera parameters, poses, and sparse 3D points, into a COLMAP-compatible representation for Gaussian Splatting reconstruction. Our reconstruction platform then processes these data to produce the final 3D scene representation. Figure 16 shows representative renderings of reconstructed corridors, office workspaces, and seating areas. Their furniture, floor textures, and lighting variations illustrate the visual realism retained in simulation. In our setting, reconstruction takes approximately $20\times$ the scanning duration; for example, a 30-minute scan requires approximately 10 hours of processing.

From these reconstructed Real2Sim scenes, we generate both nominal and pedestrian-interaction navigation data. The nominal set executes planned robot trajectories in the reconstructed environments without pedestrians. For the interaction set, we add dynamic pedestrians along the navigation routes to create time-varying encounters. We select pedestrian characters and walking or idle animations from Mixamo (Adobe), then assign the characters predefined routes through the reconstructed environments. Route geometry and timing are jointly configured to produce both blocking crossings and non-blocking encounters, in which the pedestrian and robot trajectories are separated in space

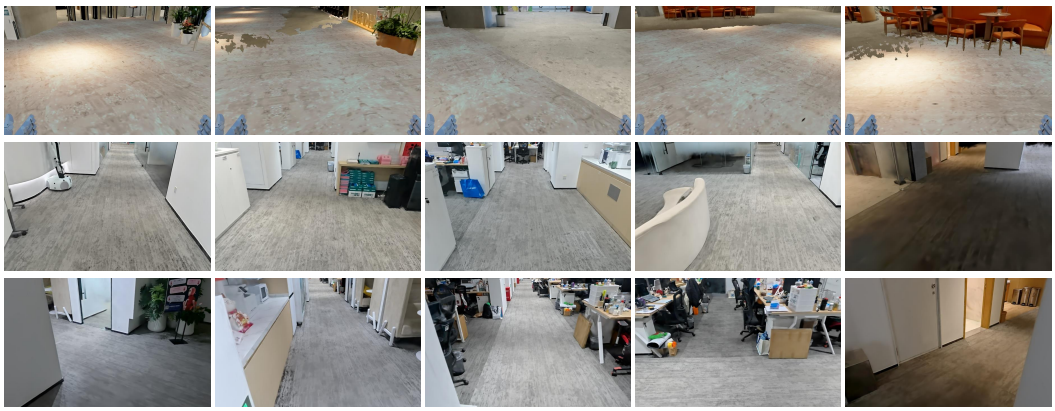

Figure 16: **Qualitative visualization of real-scene reconstruction.** Fifteen views of reconstructed indoor environments illustrate the visual fidelity and cross-view consistency of the resulting scene representation used in simulation.

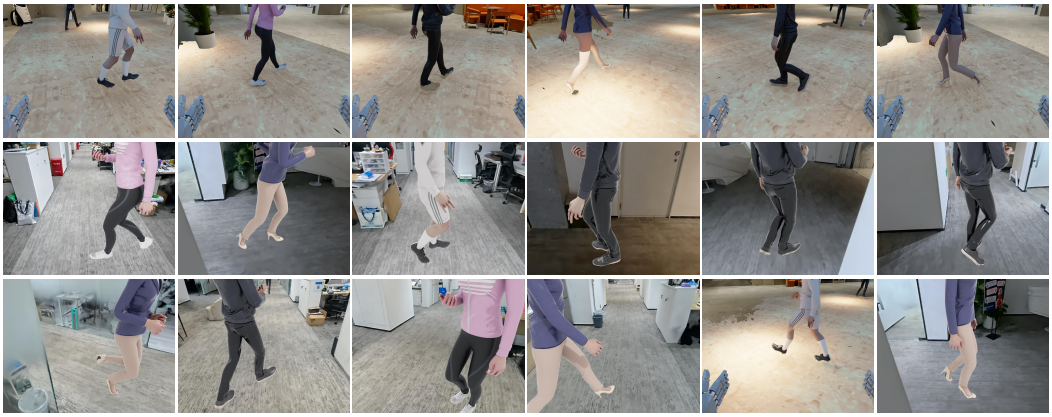

Figure 17: **Simulated pedestrian encounters for obstacle-avoidance data generation.** Eighteen representative simulation frames show animated pedestrians in reconstructed indoor environments, illustrating the dynamic obstacles used to generate pedestrian-interaction navigation data.

or time. The pedestrian motion is synchronized with the robot trajectory according to the specified interaction pattern. When a pedestrian enters the robot's forward safety region, the robot pauses until the pedestrian passes and then resumes the planned trajectory. These rollouts form dynamic navigation episodes in the reconstructed environments. Figure 17 presents representative observations and illustrates how simulated pedestrians are incorporated into scanned scenes for obstacle-avoidance data generation.

G Manipulation Data Collection and Deployment

This section details our arm–hand data collection, Nav4Manip-M executor, and real-time chunking mechanism. Recent work extends world and action models to dexterous manipulation: DexWM models action-conditioned latent dynamics, LDA-1B jointly learns dynamics and control, and WAM-TTT adapts WAMs from human play (Goswami et al., 2025; Lyu et al., 2026; Feng et al., 2026). T-Rex adds high-frequency tactile action refinement (Niu et al., 2026). Although not a WAM, EgoScale provides a complementary human-to-robot transfer approach, learning dexterous motor priors from large-scale egocentric human video (Zheng et al., 2026).

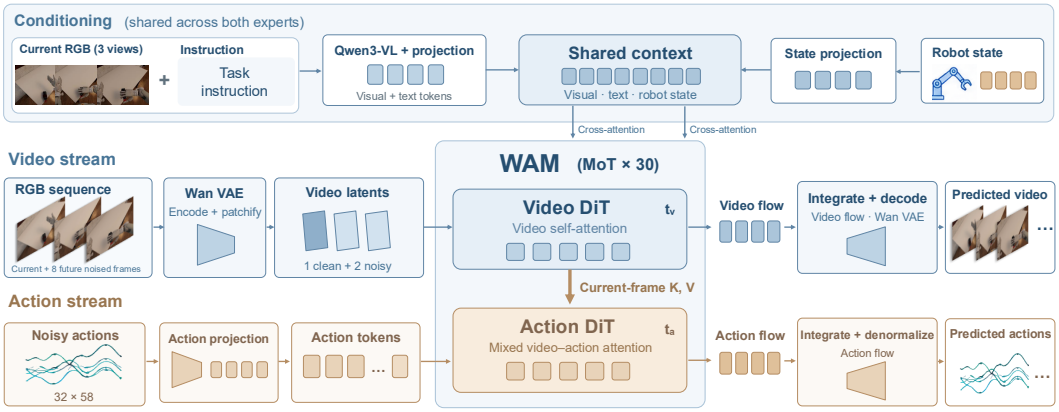

Figure 18: **NAV4MANIP-M architecture.** Three synchronized RGB views, the task instruction, and the robot state form a shared conditioning context for the video and action experts. The Video DiT models interaction dynamics in the latent video stream, while the Action DiT attends to the current-frame features and predicts a chunk of robot actions. The two experts are coupled through a 30-block Mixture-of-Transformers.

G.1 TELEOPERATED MANIPULATION DATA

We collect manipulation demonstrations using a dual-arm Galbot G3 robot equipped with two Sharpa dexterous hands. Galbot's in-house G1 Teleoperation Master–Slave Arm serves as the operator interface and provides 7-DoF control for each arm. During teleoperation, we record synchronized visual observations, observable robot states, and the control commands executed by the robot. The resulting trajectories are converted to the state and action conventions used by NAV4MANIP-M, including absolute arm–hand joint-position targets.

The collection pipeline contains both simulation and real-world stages. Operators first teleoperate the robot in MuJoCo to produce manipulation trajectories with robot states and actions. We replay the physically validated trajectories in NVIDIA Isaac Sim to render the corresponding camera observations, while retaining the original state–action sequence. Operators then perform matched tasks on the physical G3 robot, where we record the real observations, robot states, and executed actions. This procedure provides simulation-rendered data at scale while preserving real-robot demonstrations for deployment-oriented post-training and evaluation.

Demonstrations are collected by trained operators who are familiarized with the teleoperation interface, task instructions, and safety procedures before formal collection. Compensation is set above the applicable local minimum wage. The collection records robot and environment behavior: no operator-facing video, audio, biometric data, or other personally identifiable information is retained. Operators are informed of the intended use of the data and may terminate their participation at any time.

G.2 NAV4MANIP-M EXECUTOR

Once the execution agent accepts the navigation endpoint, it selects a manipulation primitive and assembles the visual–textual context described in Appendix B.3. This context is passed to NAV4MANIP-M, our task-scoped WAM executor augmented with a Qwen3-VL-4B-Instruct VLM encoder that conditions on the selected manipulation primitive. The high-level agent handles task decomposition, executor selection, and outcome verification, while NAV4MANIP-M provides closed-loop arm–hand control.

We first train NAV4MANIP-M on the manipulation trajectories in Nav4Manip-Pairs, then post-train it on demonstrations collected with the real robot to adapt it to physical execution. The trained NAV4MANIP-M is held fixed during feedback-based navigation training.

Figure 18 summarizes the NAV4MANIP-M architecture and its shared conditioning across the video and action streams. NAV4MANIP-M comprises three components: a Qwen3-VL-4B-Instruct

multimodal encoder with condition and proprioception projections, a video branch initialized from the Wan2.2 VAE and Wan2.2-TI2V-5B Video DiT, and a 1B-parameter ActionDiT branch. The video and action experts form a Mixture-of-Transformers with 30 blocks in each branch. At each model step, the policy receives the manipulation instruction, three synchronized RGB views resized to 256×256, and the current 58-dimensional robot state. The three views are concatenated into a 256×768 image. Qwen3-VL encodes this visual–language input into 2,560-dimensional tokens, which are projected to 4,096 dimensions; the robot state is independently projected into a single 4,096-dimensional context token. Both experts cross-attend to the resulting shared context.

The action branch predicts a normalized 32×58 chunk of absolute joint-position targets. After flow integration, the complete chunk is inverse-normalized using the same versioned statistics as in training and sent to the robot controller. The video branch provides learned interaction-dynamics features for action generation. During deployment, the clean current observation is processed once to condition single-step ActionDiT denoising, without explicitly decoding future RGB frames. Simulation and physical deployment therefore use the same three-view observation, proprioception, normalization, and action interfaces.

G.3 NAV4MANIP-M POST-TRAINING CONFIGURATION

For post-training of NAV4MANIP-M, we use 8 NVIDIA RTX PRO 5000 72GB GPUs, bf16, and DeepSpeed ZeRO-2, with microbatch 4 per GPU and 4 gradient-accumulation steps (global batch 128). The Wan VAE is frozen, while the Qwen3-VL VLM, WAM video branch, and ActionDiT remain trainable. AdamW uses $\beta = (0.9, 0.95)$, weight decay 0.01, and learning rates 5×10^{-7} for the VLM interface and 1×10^{-5} for the action model. The budget is up to 10,000 optimizer steps, with 1,000 warmup steps and cosine decay to a minimum learning rate of 1×10^{-7}. Normalization uses training-split statistics only; held-out losses assess prediction quality.

G.4 NAV4MANIP-M INFERENCE CONFIGURATION

We use Training-Time Real-Time Chunking (RTC) (Black et al., 2025) to teach NAV4MANIP-M to maintain continuity when inference overlaps with robot execution. Each NAV4MANIP-M call predicts a chunk of $H = 32$ model-action steps. The client requests the next chunk after a stride of $S = 16$ steps while continuing to execute the remaining actions from the active chunk, leaving an overlap of at most $H - S = 16$ steps to absorb inference and communication delay.

Training-Time RTC samples a delay $d \in [0, 16]$. The first d actions of the target chunk are supplied as a clean prefix with flow timestep zero and are excluded from the action loss; flow matching is applied only to the remaining postfix. The postfix nevertheless attends to the committed prefix, teaching the model to continue smoothly from actions already scheduled for execution. The delay distribution combines 80% samples from empirical request delays measured on the physical robot and 20% uniform samples over $[0, 16]$. The measured delays have a median of six model-action steps, a 95th percentile of nine steps, and a maximum of 13 steps. This construction requires token-wise flow timesteps but introduces no additional trainable parameters or model forward passes.

At inference, we adopt part of the flow-sampling logic from FastWAM (Yuan et al., 2026): the server performs one-step forward-Euler denoising and clamps the clean overlap prefix before and after this integration step, preventing committed actions from changing while allowing the predicted postfix to condition on them. The server conservatively estimates the prefix length as the maximum measured delay among the six most recent requests; before measurements are available, it uses an initial estimate of 12 steps. When a response arrives, the client uses the actual elapsed delay to select the first unexecuted action in the new chunk. If the active chunk expires before the response arrives, the client holds the last command, acquires a fresh observation, and sends a prefix-free request instead of executing a stale postfix. The RTC server and client use the same observation preprocessing and normalized action representation as NAV4MANIP-M training.

H REAL-WORLD DEMONSTRATION TASKS AND PROMPTS

We conducted extensive real-world experiments demonstrating that our system operates effectively in real indoor environments, with representative results presented in the supplementary video and on the

project website. This section provides the task instructions for three selected demonstrations covering navigation–manipulation coordination, long-horizon navigation, and pedestrian-aware navigation. The complete prompts for these demonstrations are listed in Table 18.

Table 18: Task prompts used in our real-world robot demonstrations. The timestamps in the Demo Index column (e.g., 00:50–01:49) indicate where each demonstration appears in the accompanying video (minutes:seconds).

Demo Index	Task Prompt
Demo 1: 00:50-01:49	**Navigation 1:** Face the table, turn left into the main aisle, and walk straight through it. After exiting the aisle, make a right U-turn and walk to the front of a shelf. **Manipulation:** Pick up the blue box of chips on the shelf. **Navigation 2:** Turn right at the shelf and return to the main aisle. Turn left and walk forward through the aisle. After passing the sofa, turn right to return beside the table, then turn left to face the table.
Demo 2: 01:51-02:20	Move forward slightly and turn right to face the glass doorway, then pass through it into the lobby. Go around the pillar, keeping it on your left, and continue forward. Turn right toward the glass entrance with green horizontal stripes and stop at the doorway.
Demo 3: 02:23-02:41	**Navigation:** Turn toward your back-left, go around the table to the doorway, then turn left into the aisle. Walk to the white table with a bowl. **Manipulation:** Pick the bowl up, and place it on the tabletop to your right as you face the table.