Foresight: Iterative Reasoning About Clues that Matter for Navigation

Arthur Zhang[1,†], **Carl Qi**[1,†], **Donne Su**[1], **Xiangyun Meng**[2], **Amy Zhang**[1], **Joydeep Biswas**[1]
[1]UT Austin, [2]FieldAI

Abstract: Open-world mapless navigation from sparse language instructions requires resolving underspecified goals and inferring which environmental cues are relevant for reaching the goal. For instance, reaching an out-of-view destination may require interpreting ramps, signs, or detours that reveal where to go or which route to take. Prior works are limited by their reliance on known navigation factors and closed-set factor categories, or identify cues before motion planning and miss plan-dependent cues. We argue that pretrained Vision-Language Models (VLMs) can discover novel instruction-relevant cues, but require adaptation to focus on which cues matter and how they should influence motion planning. We realize these ideas in FORESIGHT, a test-time framework in which a finetuned VLM alternates between proposing image-space motion plans and critiquing them using the language goal and visual context. Subsequent plans are conditioned on prior critiques, enabling iterative motion refinement before execution. To align plan critiques and refinements with open-set behavior preferences, we learn a reward model from human feedback and use it to post-train the VLM with reinforcement learning in the plan-critique loop. In offline evaluations and 6 real-world environments, FORESIGHT improves average task success by 37% and reduces interventions per mission by 52% relative to state-of-the-art test-time reasoning and foundation-model baselines, while running in real-time on a Jetson AGX Orin. We will release code, data, and training details to support future work on test-time reasoning for robot motion refinement. Additional videos at: https://amrl.cs.utexas.edu/foresight

Keywords: Mapless Navigation, Vision-Language-Models, Test-time Reasoning

1 Introduction

Deploying general-purpose robots in everyday environments requires navigation systems that can follow sparse instructions without high-definition maps, predefined routes, or exhaustive task specifications. This is challenging because successful navigation often depends on unknown open-set visual cues whose relevance is determined by the goal and local context. For example, reaching a building entrance may require interpreting signs, ramps, or badge readers that distinguish the correct route from visually similar alternatives. Such cues may be absent from training data and difficult to encode with predefined semantic classes or rules.

Learning from Demonstration (LfD) offers a potential path toward open-world navigation, but demonstrations provide weak supervision for identifying which visual cues matter and how they should influence routing decisions. Feed-forward policies [1, 2, 3, 4] learn direct observation-to-action mappings that implicitly treat familiar and novel factors equally, limiting understanding of novel cues that cannot be directly matched to prior demonstrations. Pre-emptive reasoning methods [5, 6, 7, 8] infer task-relevant factors before planning, but cue relevance is often plan-dependent: a detour matters only if the current plan follows a blocked path. Iterative methods [9, 10, 11] address this by evaluating and refining plans, but existing approaches rely on human critiques [11] or symbolic plans [9, 10], limiting their applicability to continuous open-world navigation.

In this paper, we present FORESIGHT, an iterative refinement framework and scalable training recipe for continuous robot motion planning. FORESIGHT adapts a pretrained Vision-Language Model

[†]Corresponding authors: {arthurz,carlqi}@cs.utexas.edu.

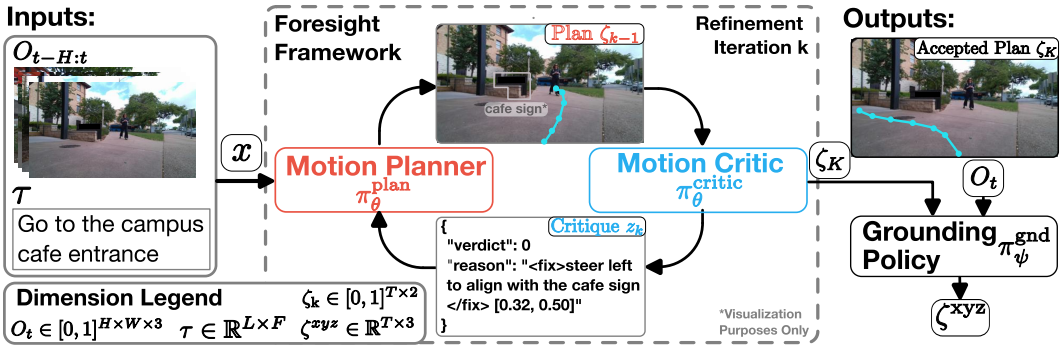

Figure 1: Overview of FORESIGHT framework. Given image observations $o_{t-H:t}$ and language task τ, FORESIGHT alternates between generating image space plans ζ_{k-1} and textual critiques z_k, conditioning on prior plan-critique pairs to refine the motion plan. A lightweight grounding policy π_ψ^{gnd} conditions on the current observation o_t to ground the final plan ζ_K to a cartesian trajectory.

(VLM) into a navigation policy that acts as both planner and critic. It proposes an image-space motion plan, critiques the plan with respect to the goal and scene, and uses the critique to refine the next plan until acceptance or a fixed refinement budget is reached. This loop leverages the ability of pretrained VLMs to identify open-world visual cues and reason about their implications for planning, but effective refinement requires adapting the model to focus on navigation-relevant cues and translate critiques into concrete plan updates. We first use supervised finetuning to teach the policy the structure of plan-critique refinement. However, imitation alone provides limited supervision for multi-step refinement because the space of possible plans, critiques, and updates branches rapidly. Reinforcement learning can improve this iterative planning process with outcome-level supervision, but requires rewards that are difficult to hand-design for open-world navigation. We therefore learn a preference reward model from human feedback and use it to post-train the VLM policy, enabling FORESIGHT to improve its plans, critiques, and refinements without dense ground-truth annotations.

We demonstrate the effectiveness of FORESIGHT through offline and real-world robot experiments on the task of mapless navigation with sparse language goals. In six real-world environments, FORESIGHT **improves average task success by 37%** and **reduces interventions per mission by 52%** relative to state-of-the-art test-time reasoning and robotics foundation-model baselines. Notably, these gains come from short, free-form reasoning traces generated by a VLM policy that **runs in real-time on a Jetson AGX Orin**, outperforming baselines that use larger models, more elaborate reasoning traces, or significantly more training data. Our main contributions are threefold: 1) we formulate iterative plan-critique refinement as a test-time reasoning framework for continuous robot motion planning; 2) we introduce a scalable training recipe that combines supervised finetuning with reinforcement learning from human preferences to adapt VLMs for iterative refinement; and 3) we demonstrate consistent improvements in task success and intervention rate across offline evaluations and closed-loop experiments in six real-world environments.

2 Related Work

Language-conditioned mapless navigation requires translating visual observations and sparse language goals into actions while inferring open-world cues whose relevance is unknown a priori. Feedforward policies adapt pretrained representations or VLMs to map observation-language pairs directly to actions [12, 3, 4, 1], but can struggle when successful navigation depends on cues or cue-action relationships that are under represented in demonstrations. This motivates test-time reasoning methods that allocate additional computation to identify relevant factors, evaluate candidate behaviors, and revise plans before execution.

2.1 Test-time Reasoning for Robotics

Pre-emptive approaches are a test-time reasoning method that use foundation models like VLMs to identify relevant scene factors before acting. This approach improves open-set cue discovery, but relies on structured prompts or predefined factor categories [5, 7], limiting performance when the

relevant cues are unknown a priori. Moreover, deciding which cues matter requires considering how they affect candidate plans. Iterative methods address this plan-dependence by using feedback to revise plans at test time [11] and foundation models to explain failures and refine behavior [13, 9, 10], but typically rely on human/environment feedback or operate over symbolic plans like domain-specific languages and code. Our work follows the iterative refinement strategy, but differs by automatically generating free-form textual critiques with a finetuned VLM and using them to refine continuous motion plans. Furthermore, FORESIGHT does not require pre-defining domain-specific factor classes, allowing our approach to express open-set cue-plan relationships during refinement.

2.2 RL Post-training for Robotics

Beyond test-time reasoning techniques, our work builds on RL post-training methods that adapt pretrained foundation models for robotics tasks. Prior works finetune VLAs from sparse task rewards [14, 15], shaped rewards such as geometric progress [16], VLM-based progress estimates [17], world-model rollouts [18, 19, 20], or preference signals from rollouts and human interventions [21, 22]. Our approach is closest to preference-based post-training as we use preference rewards to adapt a VLM inside an iterative plan-critique-refinement loop, jointly optimizing critiques and motion refinements rather than only aligning with the expert plan. This adaptation is important because pretrained VLMs may recognize open-world cues, but still need to learn which cues are relevant for motion planning and how to translate them into plan improvements. We further combine preference feedback with expert-plan alignment, reflecting that refinements must be both semantically aligned with open-world cues and geometrically consistent with expert robot motion.

3 Problem Definition and Background

We study open-world navigation with sparse language guidance, where the robot receives a context $x = (o_{t-H:t}, \tau)$ consisting of an image observation history $o_{t-H:t}$ and a natural language instruction τ. Starting at timestep t, the robot must execute a Cartesian motion plan $\zeta^{\text{xyz}} = a_{t:t+N}$, where N is the planning horizon and each waypoint $a_t \in \mathbb{R}^3$ specifies a position in the world[1]. Following common practice in vision-based control [12, 23], we represent the planned motion using normalized image-space waypoints $\zeta \in [0,1]^{N\times 2}$, which provide a compact visual specification of the desired path and are grounded into Cartesian waypoints for execution. The objective is to make progress toward the intended goal while respecting the semantic and geometric constraints of the scene. This setting is challenging because sparse instructions may under-specify the goal and route, requiring the robot to infer relevant open-set visual factors and how they influence the planned path.

Chain-of-Thought Reasoning To promote expressive reasoning about novel factors during planning, Chain-of-Thought (CoT) [24] introduces intermediate reasoning traces z between the input context x and final trajectory ζ. Instead of modeling the trajectory distribution as $p(\zeta \mid x)$, CoT explicitly generates a reasoning trace z before producing the trajectory:

$$p(\zeta, z \mid x) = p(\zeta \mid z, x)p(z \mid x) \tag{1}$$

For navigation, z encodes route-relevant information such as subgoals, relevant landmarks, spatial constraints, or high-level task representations, while ζ specifies the resulting motion plan.

Group Relative Policy Optimization. Group Relative Policy Optimization (GRPO) [25] is a policy-gradient method for optimizing a model from reward feedback. Let $\pi_\theta(y \mid x)$ be a policy that generates an output y conditioned on input x, and let $\pi_{\theta_{\text{old}}}$ denote the policy used to sample training outputs. For each input x, GRPO samples a group of G outputs $\{y^i\}_{i=1}^G$ from $\pi_{\theta_{\text{old}}}(\cdot \mid x)$. Each output receives a scalar reward r^i. The rewards are normalized within the group to produce relative advantages $A^i = \frac{r^i - \text{mean}_j(r^j)}{\text{std}_j(r^j)+\epsilon}$, where ϵ is a small constant for numerical stability. The normalized advantage A^i measures whether output y^i is better or worse than other samples for the

[1] We consider 3D positions for a ground robot to accommodate stairs, ramps, and other non-planar terrains

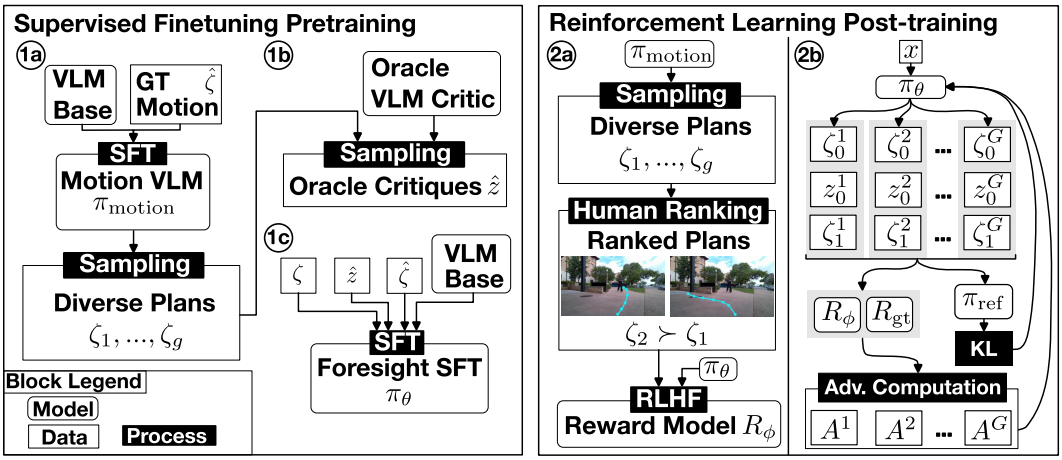

Figure 2: Overview of the FORESIGHT training recipe. During supervised pre-training, we finetune a VLM for the iterative plan-critique (1c) roles using rollouts $(\zeta, \hat{z}, \hat{\zeta})$, the noisy plan, oracle critique, and ground truth plan respectively. In the second reinforcement learning stage, we learn a reward model for ranking motion plans ζ from a human-labeled preference dataset (2a) and optimize the VLM policy π_θ in the plan-critique loop using Group Relative Policy Optimization [25] (2b).

same input, avoiding the need to learn a separate value function. The policy is then updated via:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{y^i\}_{i=1}^G} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{\pi_\theta(y^i \mid x)}{\pi_{\theta_{\text{old}}}(y^i \mid x)} A^i \right] - \beta \, \text{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)) \tag{2}$$

where β controls regularization toward a reference policy π_{ref}.

4 Approach

We introduce FORESIGHT, a test-time motion refinement framework for open-world mapless navigation from sparse language instructions. As shown in Fig. 1, FORESIGHT consists of two high-level components: an image-space refinement policy and a lightweight grounding policy. First, a single VLM with role-specific prompts samples an iterative plan-critique trace $p_\theta(\zeta_{0:K}, z_{0:K-1} \mid x)$ and produces a refined image-space motion plan ζ_K. Second, a lightweight transformer-based grounding policy π_ψ^{gnd} maps ζ_K and current observation o_t to a metric trajectory ζ^{xyz}. Together, this defines the following factorization:

$$p_{\theta,\psi}(\zeta^{\text{xyz}}, \zeta_{0:K}, z_{0:K-1} \mid x, o_t) = \underbrace{\pi_\psi^{\text{gnd}}(\zeta^{\text{xyz}} \mid o_t, \zeta_K)}_{\text{grounding policy}} \underbrace{p_\theta(\zeta_{0:K}, z_{0:K-1} \mid x)}_{\text{FORESIGHT trace}}. \tag{3}$$

Eq. 3 relies on FORESIGHT to accurately steer the grounding policy to generate safe, task-aligned executable motion trajectories. We provide implementation details for π_ψ^{gnd} in the Appendix Sec. F.1. In the remainder of this section, we formalize iterative motion refinement as a chain-of-thought factorization, describe the SFT pre-training procedure, and present the reward design and RL post-training recipe for the plan-critique loop.

4.1 Iterative Motion Refinement as Chain-of-Thought Planning

Let $x = (o_{t-H:t}, \tau)$ denote the navigation context, consisting of an observation history and sparse language instruction. We cast motion refinement as an iterative CoT process indexed by refinement step $k = 0, \ldots, K$, distinct from the robot execution timestep t. A single VLM policy π_θ uses role-specific prompts to define a planner π_θ^{plan} and critic $\pi_\theta^{\text{critic}}$, where the planner generates an initial motion plan ζ_0 from x, the critic generates a textual critique trace z_k that evaluates the current plan ζ_k, and the planner generates the next plan ζ_{k+1} conditioned on both ζ_k and z_k:

$$p_\theta(\zeta_{0:K}, z_{0:K-1} \mid x) = \underbrace{\pi_\theta^{\text{plan}}(\zeta_0 \mid x)}_{\text{initial plan}} \prod_{k=0}^{K-1} \underbrace{\pi_\theta^{\text{plan}}(\zeta_{k+1} \mid \zeta_k, z_k, x)}_{\text{critique-conditioned plan}} \underbrace{\pi_\theta^{\text{critic}}(z_k \mid \zeta_k, x)}_{\text{critique}}. \tag{4}$$

Here, each critique z_k provides textual feedback about visual cues, task constraints, or plan corrections relevant to improving ζ_k. This factorization couples critique generation with planning, encouraging the critic to produce refinement-relevant feedback and the planner to use critique-identified factors to adapt motion plans to cues not specified a priori or seen in demonstrations.

4.2 Supervised Pre-training

We begin with supervised finetuning to warm-start a policy for the proposed plan-critic roles in Eq. 4. As shown in Fig. 2, we first train a base planner $\pi_\theta^{\text{motion}}$ to imitate expert plans $\hat{\zeta}$ and then sample noisy candidate plans $\zeta \sim \pi_\theta^{\text{motion}}(x)$. We use an Gemini-3.1-Flash [26] to generate oracle critiques $\hat{z}$ that describe the relevant factors and how to improve ζ to align with the task instruction. This yields single-step refinement tuples $\mathcal{D}_{\text{SFT}} = \{(x, \zeta, \hat{z}, \hat{\zeta})\}$, where $\hat{\zeta}$ denotes the expert motion plan. We supervise the critic π^{critic} to predict the oracle critique and the planner in two contexts: predicting the expert plan directly from x, and predicting the expert plan conditioned on a noisy plan and critique. The SFT objective is

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,\zeta,\hat{z},\hat{\zeta})\sim\mathcal{D}_{\text{SFT}}}\left[\log \pi_\theta^{\text{plan}}(\hat{\zeta} \mid x) + \log \pi_\theta^{\text{critic}}(\hat{z} \mid \zeta, x) + \log \pi_\theta^{\text{plan}}(\hat{\zeta} \mid \zeta, \hat{z}, x)\right]. \tag{5}$$

This warm-start provides high-quality offline supervision, but pairing oracle critiques of varying relevance with the same expert target plan prevents SFT from learning which critiques best guide refinement. We address this limitation using reinforcement learning to jointly optimize critique generation and critique-conditioned motion planning.

4.3 Preference-Based RL Post-Training

We post-train the SFT policy to optimize complete iterative refinement rollouts from Eq. 4 using Group Relative Policy Optimization (GRPO) [25] with plan-level outcome supervision. In this section, we present a scalable reward design for open-world navigation and a tractable group-sampling procedure for assigning this reward to online plan-critique rollouts.

Reward design. It is challenging to hand-design rewards for open-world navigation as good plans must balance diverse constraints like task alignment, local safety, and long-range visual cues. We therefore learn a plan-quality reward from human preferences over candidate plans as shown in Fig. 2 section 2a. Given a context x and pair of plans (ζ^+, ζ^-), where ζ^+ is preferred, we train a reward model $R_\phi(x, \zeta)$ with the Bradley-Terry [27] objective:

$$\mathcal{L}_{\text{RM}}(\phi) = -\mathbb{E}_{(x,\zeta^+,\zeta^-)}\left[\log \sigma\big(R_\phi(x, \zeta^+) - R_\phi(x, \zeta^-)\big)\right]. \tag{6}$$

Following prior work on reward modeling [13, 21], we initialize R_ϕ from the finetuned VLM π_θ and replace the policy head with a linear reward head over the penultimate hidden state. Since R_ϕ scores only the context and plan, it is agnostic to the refinement history and critique text. We combine this learned reward with an expert-alignment reward $R_{\text{exp}}(x, \zeta)$ that is inversely related to the Hausdorff distance between ζ and the expert plan $\hat{\zeta}$, yielding the final plan reward:

$$R(x, \zeta) = R_\phi(x, \zeta) + \lambda R_{\text{exp}}(x, \zeta), \tag{7}$$

where λ controls the strength of expert alignment. We define R_{exp} in Appendix Sec. C.1.

Practical group sampling and co-training. For each context x, we first sample a shared initial plan $\zeta_0 \sim \pi_\theta^{\text{plan}}(\cdot \mid x)$. We then sample G refinement rollouts conditioned on ζ_0:

$$\rho^i = (\zeta_0, z_0^i, \zeta_1^i, \ldots, \zeta_{\kappa_i}^i, z_{\kappa_i}^i), \qquad i = 1, \ldots, G,$$

where each rollout follows Eq. 4 after the initial plan. The rollout terminates when the critic accepts the current plan or when the refinement budget K is reached; let $\kappa_i \le K$ denote this stopping step, and let $\zeta_{\kappa_i}^i$ denote the selected final plan. Using a shared ζ_0 avoids confounding initial plan quality with the quality of subsequent critiques and refinements. We score each rollout using the outcome reward of its selected final plan and compute group-normalized advantages:

$$r^i = R(x, \zeta_{\kappa_i}^i), \qquad A^i = \frac{r^i - \text{mean}_j(r^j)}{\text{std}_j(r^j) + \epsilon}. \tag{8}$$

We optimize the shared VLM with Group Relative Policy Optimization (GRPO) [25] using A^i as the rollout-level advantage and a KL penalty to the frozen SFT reference policy $\pi_{\theta_{\text{SFT}}}$. While this only provides indirect critique supervision [28], we observe strong empirical gains and motivate this from an information-gain perspective in Appendix Sec. D, drawing parallels between how optimizing the critique for our proposed reward parallels discovering factors that lead to greater information gain.

Approach Summary. As mentioned at the beginning of this section and in Fig. 2, our method consists of the iterative motion refinement formulation defined in Eq. 4, SFT pre-training to encourage expert-aligned plan-critique generation, and RL-post-training that uses a hybrid preference and expert alignment reward R_ϕ to provide outcome-level supervision.

5 Experiments and Results

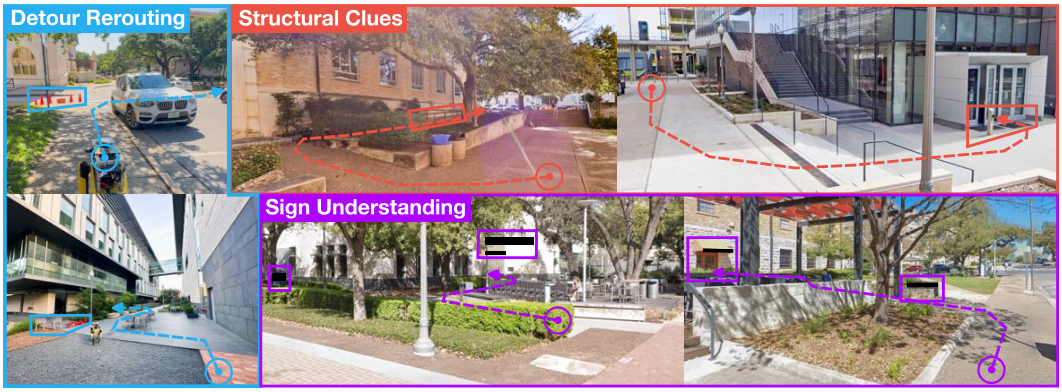

Figure 3: Real-world experiment scenarios. Bounding boxes annotate the key visual clues for each scenario, and dashed arrows show the intended route from the start (circular dot). Annotations for visualization purposes only, not given to the algorithms. We redact building signs for anonymity.

In this section, we describe the evaluation methodology for FORESIGHT and answer the following questions to understand the importance of our contributions and overall performance on the task of mapless navigation with sparse language guidance.

($\mathcal{Q}_1$) Does FORESIGHT improve instruction-following navigation over state-of-the-art approaches, including feedforward or pre-emptive reasoning methods?

($\mathcal{Q}_2$) Does reinforcement learning post-training improve FORESIGHT's ability to critique and refine plans around open-world navigation cues?

($\mathcal{Q}_3$) Does FORESIGHT's proposed reward design improve refinement learning, or are verifiable learned or geometric rewards alone sufficient?

We investigate these questions through offline and real-world robot experiments using state-of-the-art open and closed-source model baselines. For the offline benchmark, we collect an offline instruction following navigation dataset with expert demonstrations in a variety of urban environments.

Experimental Setup. We conduct all real-world tests using the Boston Dynamics Spot Robot using an Azure Kinect RGB-D camera to obtain RGB observations at 15Hz and odometry at 50Hz from the Boston Dynamics API. We use the same pure pursuit controller [29] and grounding model π_{gnd} for baselines and provide additional architecture and training details in the Appendix Sec. F.1. For details on compute hardware and inference, please see Appendix Sec. F.4.

Training and Evaluation Metholodogy. We co-train all model baselines on SCAND [30] and 1.5 hours of teleoperated instruction following demonstrations (The FORESIGHT Dataset). Additional dataset details, including qualitative samples, can be found in Appendix Sec. E. To focus RL post-training on scenarios that require clue understanding, we use only the FORESIGHT dataset and annotate 1 hour of demonstrations with ranked trajectory preferences. Additional methodological details are provided in Appendix Sec. F.3. We generate diverse language instructions, oracle textual critiques, and CoT reasoning traces using Gemini-3.1-Flash [31].

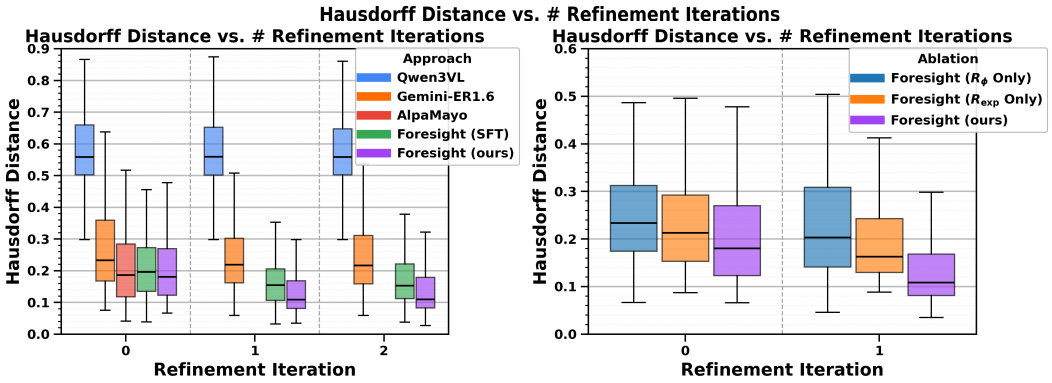

Figure 4: Mean Hausdorff distance compared to expert demonstration with varying refinement iterations. The left plot compares FORESIGHT against state-of-the-art baselines, where FORESIGHT (SFT) is our approach but with only supervised pre-training. The right plot ablates different reward designs: R_ϕ - only using learned rewards, R_{exp} - only using expert alignment reward.

For offline evaluation, we randomly withhold a subset of scenarios from the FORESIGHT dataset to obtain 984 evaluation samples and use Hausdorff distance with the expert trajectory in bird's eye view (BEV) space. We use the same grounding policy π_{gnd} to convert image space plans to BEV trajectories for consistency. In addition, we average 12 randomly generated rollouts for each test sample for more accurate performance estimatation. For more details on trajectory ranking, dataset examples, and prompts, please see Appendix Sec. E.

For the real-world robot experiments, we evaluate in 6 environments categorized by the factor and navigation behavior being tested. Fig. 3 provides qualitative examples for the three categories: detour re-routing, structural clues, and sign understanding. We include 1 seen and unseen environment for each category and conduct 4 trials per baseline in each environment. We permit at most 3 interventions before deeming the test unsuccessful and only intervene when the robot becomes stuck, before catastrophic collisions, or after executing plans that make reaching the goal infeasible.

Model Baselines. We compare against several state-of-the-art language conditioned navigation baselines. We finetune LeLaN [4], a feedforward model pre-trained on internet navigation demonstrations. We reproduce AlpaMayo [7], a VLM that performs structured CoT reasoning about pre-defined factor classes and provides high-level directional guidance. To adapt Alpamayo for onboard reasoning in pedestrian friendly scenarios, we finetune Qwen3-VL-2B-Instruct [32] to generate oracle CoT traces from Gemini-3.1-Flash and expert image space plans. Lastly, we compare against the closed-source Gemini-ER1.6 [26] VLM for offline evaluations, but do not include this for real-robot experiments due to internet connectivity and latency limitations. All FORESIGHT models adapt the Qwen3-VL-2B-Instruct model for consistency. Additional baseline implementation details can be found in Appendix Sec. F.

Category / Baseline	Sign Under. Succ. / Trials	Sign Under. # Int.	Structural Succ. / Trials	Structural # Int.	Detour Succ. / Trials	Detour # Int.
LeLaN [4]	1 / 8	3.00	2 / 8	2.00	2 / 8	2.00
Alpamayo [7]	4 / 8	2.00	3 / 8	1.67	4 / 8	2.00
FORESIGHT, no ref/rl	2 / 8	2.00	4 / 8	2.00	3 / 8	2.33
FORESIGHT, no rl	4 / 8	1.00	6 / 8	1.67	5 / 8	1.80
FORESIGHT (ours)	**7 / 8**	**0.71**	**7 / 8**	**1.14**	**6 / 8**	**1.00**

Table 1: Task success counts over total trials (Succ. / Trials) and # of Interventions (# Int.) across sign understanding (Sign Under.), structural clues, and detour rerouting real-world robot experiments. All baselines are allowed up to 3 interventions before failure. Here, no ref/rl indicates no reflections or RL finetuning.

5.1 Results and Analyses

Effectiveness of Iterative Plan-Critiques ($\mathcal{Q}_1$). Fig. 4 compares the performance of FORESIGHT against other CoT baselines. Furthermore, it ablates the performance impact of iterative refinement and the relative importance of the motion and critic for downstream motion planning. Our approach trained only using SFT (FORESIGHT SFT) outperforms the Gemini-ER1.6 and AlpaMayo baselines

on average after 1 refinement iteration despite generating freeform reasoning traces, using fewer model parameters, and training on less data. Additionally, we observe that FORESIGHT greatly outperforms a zero shot Qwen3VL model, corroborating our hypothesis that while VLMs contain some capacity for factor discovery and refinement, finetuning is necessary for understanding the most relevant factors and how they influence motion planning. We supplement this analysis with a real-world failure taxonomy for Alpamayo and FORESIGHT in Appendix Sec. B.3. These results further support our claim that reasoning is both cue and plan dependent, highlighting how our method of jointly co-training the critic and planner reduces the percentage of critic failures by 23% compared to pre-emptive CoT reasoning. For additional ablation studies on the relative importance of the motion planner versus critic, we refer readers to Appendix Sec. B. We provide qualitative comparisons in Appendix Sec. B.1.

Effectiveness of RL Post-training ($\mathcal{Q}_2$). Comparing RL post-training (RL/RL) to only SFT fine-tuning (SFT/SFT) in Fig. 4, we find that RL post-training reduces the final planning error by 26% on average. Our real-world experiments in Table 1 corroborate this finding, showing that RL improves the success rate by 20% and decreases the number of interventions per mission by 36% on average relative to SFT only (FORESIGHT-rl). Thus, we conclude that our proposed RL post-training recipe is effective for improving real-world motion planning performance.

Reward Design for RL Post-training ($\mathcal{Q}_3$). Fig. 4 compares different reward combinations used for RL post-training by measuring offline planning error with respect to expert demonstrations. While using only geometric rewards (R_{exp}) reduces planning error, it is substantially less effective than combining learned and geometric rewards (ours). In contrast, using the learned reward (R_ϕ) alone is also ineffective. We hypothesize that learned and geometric rewards provide complementary signals that compensate for their individual weaknesses: geometric rewards do not capture non-geometric factors such as terrain or semantic affordances, leading to corner cutting and other undesirable behaviors, whereas learned rewards may capture acceptable trajectory modes without necessarily selecting the task-optimal plan. These findings suggest that combining learned and geometric rewards is important for stable RL convergence.

6 Conclusion

We presented FORESIGHT, an iterative plan-critique framework for open-world mapless navigation from sparse language instructions. By adapting a VLM to critique and refine image-space plans, FORESIGHT identifies novel plan-relevant visual factors and translates them into task-aligned motion updates. We train this loop with supervised pre-training and preference-based RL, enabling joint optimization of critique generation and critique-conditioned planning without dense critique-refinement annotations. Across offline and real-world experiments, FORESIGHT improves task success by 37% and reduces interventions per mission by 52%, suggesting that iterative VLM self-critique is an effective mechanism for open-world robot motion refinement.

7 Limitations and Future Work

While FORESIGHT makes significant strides towards scalable mapless navigation, several challenges remain. First, outcome-level supervision makes credit assignment difficult in multi-step refinement, since gains may arise from better critiques, critique-conditioned planning, or critique following. Process level supervision [25] can provide denser supervision to mitigation spurious correlations and enhance instruction following. Second, limited memory and multi-view understanding can hurt long-horizon navigation when relevant cues are sparse or observed briefly. Alternate forms of memory like retrieval augmented generation [33] can provide more informative context and co-training on multi-view reasoning tasks [34] can further improve multi-view cue grounding and reasoning. Finally, our experiments focus primarily on static environments, leaving dynamic settings that require faster policies and understand temporal relationships for future work.

Acknowledgments

This work has taken place in the Autonomous Mobile Robotics Laboratory (AMRL) and Machine Decision-making through Interaction Laboratory (MIDI) at UT Austin. AMRL research is supported in part by NSF (CAREER-2046955, IIS-2416461, PARTNER-2402650), ARO (W911NF-24-2-0025), and FieldAI. MIDI research is supported in part by NSF (CAREER-2340651, PARTNER-2402650), DARPA (HR00112490431), and ARO (W911NF-24-1-0193). We thank Arjun Guha and Amirreza Shaban for their support, advice, comments, and discussions during the project. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] A. Zhang, H. Sikchi, A. Zhang, and J. Biswas. Creste: Scalable mapless navigation with internet scale priors and counterfactual guidance. In *Proceedings of Robotics: Science and Systems XXI*. Robotics: Science and Systems, 2025.

[2] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[3] N. Hirose, C. Glossop, D. Shah, and S. Levine. Omnivla: An omni-modal vision-language-action model for robot navigation. *arXiv preprint arXiv:2509.19480*, 2025.

[4] N. Hirose, C. Glossop, A. Sridhar, O. Mees, and S. Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild video. In *8th Annual Conference on Robot Learning*, 2024.

[5] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, pages 3157–3181. PMLR, 2025.

[6] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[7] Y. Wang, W. Luo, J. Bai, Y. Cao, T. Che, K. Chen, Y. Chen, J. Diamond, Y. Ding, W. Ding, et al. Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail. *arXiv preprint arXiv:2511.00088*, 2025.

[8] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi\_\{0.5\}$: a vision-language-action model with open-world generalization. *eprint arXiv: 2504.16054*, 2025.

[9] J. Kim, C. Min, B. Kim, and J. Choi. Pre-emptive action revision by environmental feedback for embodied instruction following agents. In *8th Annual Conference on Robot Learning*, 2024.

[10] M. Han, Y. Zhu, S. Zhu, and Y. Wu. Interpret: Interactive predicate learning from language feedback for generalizable task planning. In *2024 IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024.

[11] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, pages 1769–1782. PMLR, 2023.

[12] A. Zhang, X. Meng, L. Calliari, D.-K. Kim, S. Omidshafiei, J. Biswas, A. Agha, and A. Shaban. Ventura: Adapting image diffusion models for unified task conditioned navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[13] C. Qi, X. Wang, S. Yong, S. Sheng, H. Mao, M. Nambi, A. Zhang, Y. Dattatreya, et al. Self-refining vision language model for robotic failure detection and reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[14] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani. Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3617–3624. IEEE, 2025.

[15] H. Li, Y. Zuo, J. Yu, Y. Zhang, Z. Yang, K. Zhang, X. Zhu, Y. Zhang, T. Chen, G. Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

[16] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs. Poliformer: Scaling on-policy rl with transformers results in masterful navigators. In *Conference on Robot Learning*, pages 408–432. PMLR, 2025.

[17] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, et al. $\pi_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[18] H. He, Y. Ma, W. Wu, and B. Zhou. From seeing to experiencing: Scaling navigation foundation models with reinforcement learning. *arXiv preprint arXiv:2507.22028*, 2025.

[19] S. Yong, S. Sheng, C. Qi, X. Wang, E. Sheehan, A. Shivaprasad, Y. Xie, K. Sycara, and Y. Dattatreya. Generalizable dense reward for long-horizon robotic tasks. *arXiv preprint arXiv:2604.00055*, 2026.

[20] H. Li, P. Ding, R. Suo, Y. Wang, Z. Ge, D. Zang, K. Yu, M. Sun, H. Zhang, D. Wang, et al. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025.

[21] Z. Zhang, K. Zheng, Z. Chen, J. Jang, Y. Li, S. Han, C. Wang, M. Ding, D. Fox, and H. Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

[22] W. Xia, Y. Yang, H. Wu, X. Ma, T. Kong, and D. Hu. Human-assisted robotic policy refinement via action preference optimization. *Advances in Neural Information Processing Systems*, 38: 36746–36768, 2026.

[23] J. Lee, J. Duan, H. Fang, Y. Deng, B. Li, S. Liu, B. Fang, J. Zhang, Y. R. Wang, S. Lee, et al. Molmoact: Action reasoning models that can reason in space. In *Workshop on Making Sense of Data in Robotics: Composition, Curation, and Interpretability at Scale at CoRL 2025*, 2025.

[24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[25] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[26] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[27] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324, 1952. URL https://api.semanticscholar.org/CorpusID:125209808.

[28] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[29] Y. Huang, Z. Tian, Q. Jiang, and J. Xu. Path tracking based on improved pure pursuit model and pid. In *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, pages 359–364. IEEE, 2020.

[30] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 7(4):11807–11814, 2022.

[31] S. Pichai, D. Hassabis, and K. Kavukcuoglu. A new era of intelligence with gemini 3. *Mountain View, CA: Google). Available online at: https://blog. google/products-andplatforms/products/gemini/gemini-3/(Accessed February 1, 2026)*, 2025.

[32] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[33] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[34] A.-C. Cheng, Y. Fu, Y. Chen, Z. Liu, X. Li, S. Radhakrishnan, S. Han, Y. Lu, J. Kautz, P. Molchanov, et al. 3d aware region prompted vision language model. *arXiv e-prints*, pages arXiv–2509, 2025.

[35] M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975. doi:https://doi.org/10.1002/cpa.3160280102. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160280102.

[36] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

[37] B. Koonce. Efficientnet. In *Convolutional neural networks with swift for Tensorflow: image recognition and dataset categorization*, pages 109–123. Springer, 2021.

[38] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[39] G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

A Appendix

This appendix supplements the main paper with additional experimental analysis, reward derivations, dataset and prompt details, and model implementation details. Specifically, we provide qualitative comparisons, quantitative ablations and a real-world failure taxonomy in Sec. B; define the expert-alignment reward in Sec. C and analyze its connection to critique-induced information gain in Sec. D; and describe the dataset, prompts, model training, reward learning, and deployment stack in Sec. E and Sec. F.

B Additional Results

B.1 Qualitative Comparisons

We provide additional qualitative comparisons between FORESIGHT and the highest performing baseline, Alpamayo [7] in Fig. 5. We observe that Alpamayo is capable of inferring open-world visual clues like paths and doorway entrances, but often fails to focus on the most critical clues, like crosswalks, leading to suboptimal behavior compared to FORESIGHT. Furthermore, we find that FORESIGHT tends to refine the initial plan even if only small adjustments are needed. We hypothesize this occurs because reinforcement learning post-training optimizes our policy to make any improvements to the plan, regardless of their necessity. While this may improve robustness in scenarios with little margin for error, it may be potentially wasteful and motivates alternate reward designs that balance the inference cost of refinement with the current plan quality.

Figure 5: Qualitative comparison between FORESIGHT and Alpamayo [7] across various experiments. We annotate the visual clue for visualization only.

B.2 Training Recipe Ablations

In Fig. F.3.3, we conduct additional ablation sutdies to understand the importance of our training recipe decisions. Here, we adopt the naming convention A / B, where A represents the model used for motion planning and B represents the model used for the critic. For brevity, we use ZS to indicate that the model is tested Zero Shot, SFT to indicate that the model has undergone supervised finetuning (SFT), and RL to indicate that the model has undergone SFT and reinforcement learning (RL). We use Qwen3-VL-2B-Instruct as the base model for all ablations.

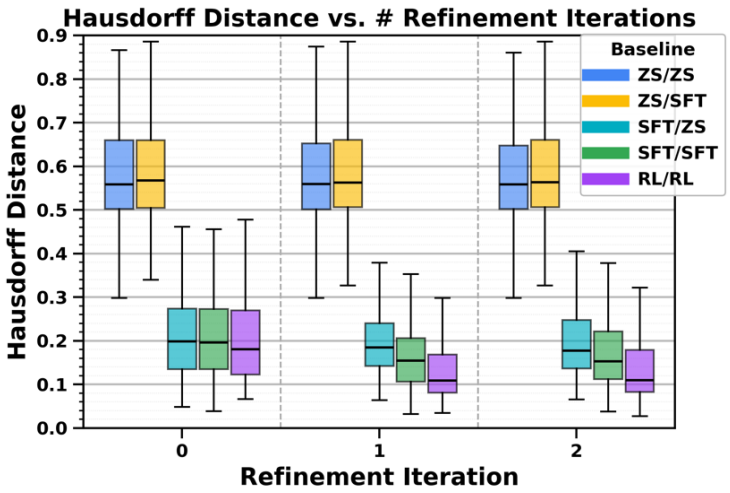

Figure 6: Average Hausdorff distance error compared to expert demonstration with respect to the number of refinement iterations. For a full explanation of the naming convention used in the legend, we refer the reader to Appendix Sec. B.

We observe in Fig. F.3.3 that all finetuned motion planners perform significantly better than zero shot. Furthermore, across iteration 0 (no refinements) to iteration 1, SFT/SFT outperforms SFT/ZS, demonstrating the importance of high quality critiques for motion refinement. Lastly, the planning error in ZS/SFT does not decrease with additional refinements, suggesting that simply finetuning the critic is not sufficient for motion refinement. We hypothesize this occurs because while the critiques may contain task-relevant factors for motion planning, the motion planner is unable to translate these factors to refine the motion plan.

B.3 Real-world Experiment Failure Taxonomy

To better understand the causes real-world failure modes, we manually annotate the top-performing baseline, AlpaMayo [7] and FORESIGHT to identify the categories of failures encountered. We order the diagram by if the cause of failure was due to poor CoT reasoning (critic) or motion planning (planner). Critic failures are either caused by ambiguous feedback or simply omitting the required factors needed for downstream planning. Planner failures are attributed to poor instruction following. In Fig. 7, we see that FORESIGHT reduces the relative percentage of critic failures by 23% from 81% to 58%. These results corroborate our claim that CoT reasoning is dependent on the visual cues and plans available to the policy. This also highlights the need for techniques that explicitly optimize for better motion planner instruction following and critics that understand not only what is in the scene, but what kind of feedback is helpful to guide motion planning.

C Reward Design

C.1 Expert-alignment Reward Definition

Because our image-space plans are represented in normalized image coordinates, each waypoint lies in the unit square $[0,1]^2$. We measure the geometric distance between a predicted plan ζ and expert plan $\hat{\zeta}$ using the symmetric Hausdorff distance

$$d_{\mathrm{H}}(\zeta,\hat{\zeta}) = \max\left\{ \sup_{p\in\zeta}\inf_{q\in\hat{\zeta}} \|p-q\|_2,\ \sup_{q\in\hat{\zeta}}\inf_{p\in\zeta} \|q-p\|_2 \right\}. \tag{9}$$

Since both trajectories lie in $[0,1]^2$, the maximum possible pointwise distance is the unit-square diagonal $\sqrt{2}$. We convert this distance into a bounded reward by linearly mapping zero error to 1 and a distance of $\sqrt{2}/2$ to -1, then clipping larger errors:

$$R_{\exp}(x,\zeta) = \mathrm{clip}\left(1 - \frac{4}{\sqrt{2}} d_{\mathrm{H}}(\zeta,\hat{\zeta}), -1, 1\right). \tag{10}$$

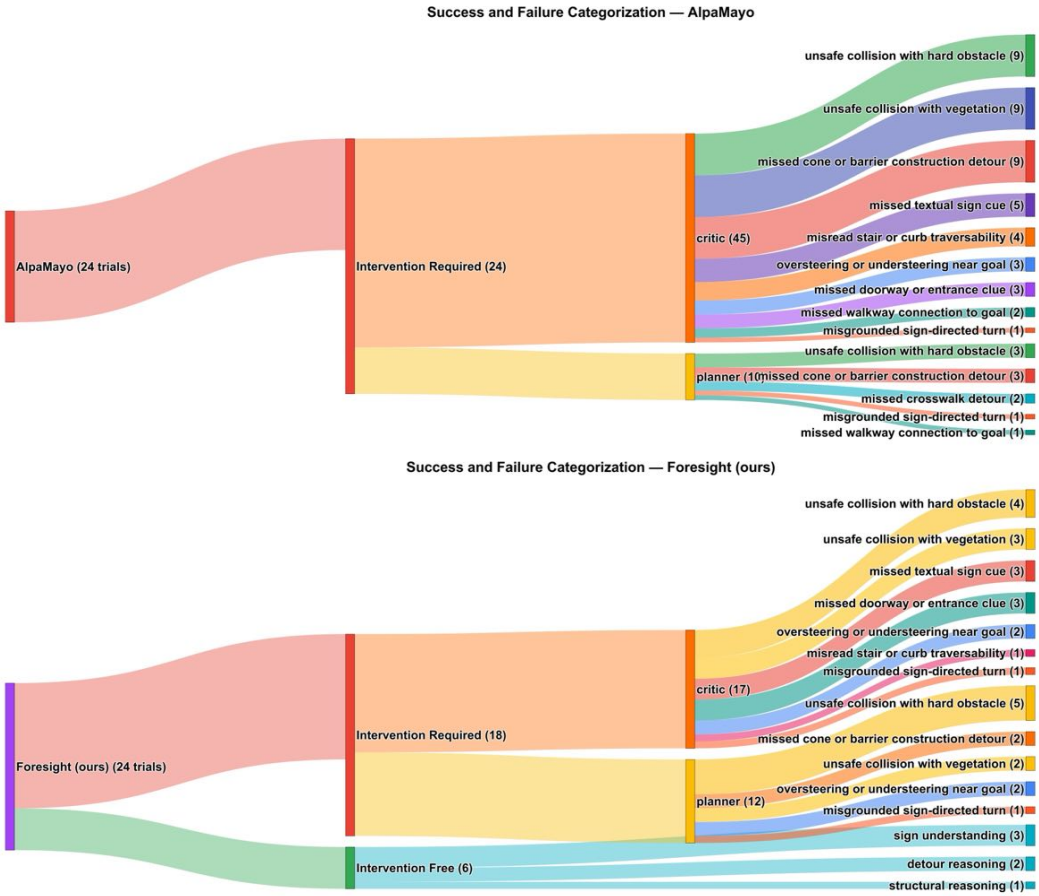

Figure 7: Success and Failure Taxonomy for Real World Experiments. We categorize the failures based on if they are caused by the critic or planner before describing the exact scene factor that caused each intervention.

Thus, plans that closely match the expert receive high reward, while plans whose Hausdorff distance exceeds half the image diagonal receive the minimum reward.

D Reward Improvement and Information Gain

We derive the connection between the expected delta reward and the information gain induced by a critique. For a fixed context x, current trajectory ζ, and critique z, define

$$p(\zeta) \;=\; \pi_\theta^{\text{refine}}(\zeta \mid z, x), \qquad q(\zeta) \;=\; \pi_\theta^{\text{plan}}(\zeta \mid x), \tag{11}$$

so that $\mathrm{IG}(z) = \mathrm{KL}(p \,\|\, q)$ and $\overline{\Delta r}(z) = \mathbb{E}_p[r] - \mathbb{E}_q[r]$. The Donsker–Varadhan [35] variational representation of KL divergence states that for any function f with $\mathbb{E}_q[e^f] < \infty$,

$$\mathrm{KL}(p \,\|\, q) \;\geq\; \mathbb{E}_p[f] - \log \mathbb{E}_q[e^f], \tag{12}$$

with equality when $f = \log(p/q) + \text{const}$. The inequality holds for every admissible f, so we are free to choose any reward function r and obtain a valid lower bound.

Substituting $f = \lambda r$ with a free parameter $\lambda > 0$:

$$\mathrm{IG}(z) \;\geq\; \lambda \mathbb{E}_p[r] - \log \mathbb{E}_q[e^{\lambda r}]. \tag{13}$$

Decomposing $\mathbb{E}_p[r] = \overline{\Delta r}(z) + \mathbb{E}_q[r]$ on the right-hand side:

$$\mathrm{IG}(z) \;\geq\; \lambda\, \overline{\Delta r}(z) + \lambda \mathbb{E}_q[r] - \log \mathbb{E}_q[e^{\lambda r}]. \tag{14}$$

Equivalently,

$$\mathrm{IG}(z) \geq \lambda \overline{\Delta r}(z) + C_q, \qquad C_q = \lambda \mathbb{E}_{\zeta \sim q}[r(\zeta)] - \log \mathbb{E}_{\zeta \sim q}[\exp(\lambda r(\zeta))]. \tag{15}$$

For a fixed policy snapshot, context x, and current trajectory ζ_t, the baseline distribution q is fixed when comparing different critiques z_t. Therefore, C_q is constant with respect to the critique. Maximizing the expected delta reward $\overline{\Delta r}(z_t)$ therefore maximizes a Donsker–Varadhan lower bound on the information gain induced by the critique, under the current policy.

During training, the policy parameters change and therefore the baseline distribution q also changes. We interpret Eq. 15 as a local justification for the reward objective: at each policy snapshot, critiques that yield larger expected reward improvement correspond to larger values of a lower bound on critique-induced information gain.

E FORESIGHT Dataset and Prompts

In this section, we describe the environments of interest, oracle critique dataset generation procedure in Sec. E.1, and role-specific FORESIGHT prompts in Sec. E.2 used for training and inference.

Environments of Interest. The FORESIGHT dataset consists of 88 missions in 25 unique urban/-campus environments on a variety of goal-reaching navigation tasks that require sign understanding, inferring structural clues, and identifying detour routes. Fig. 8 provides a gallery of scenarios with the natural language task, expert demonstration, and key visual clue highlighted for visualization purposes only.

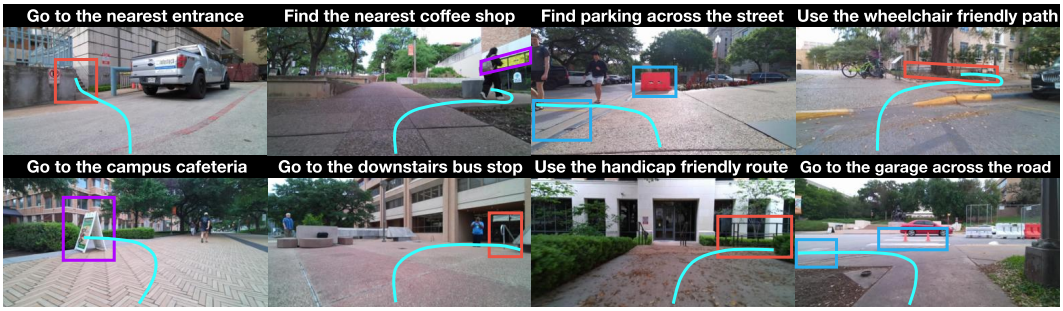

Figure 8: Examples from the FORESIGHT dataset. For visualization purposes only, we highlight key visual clues for satisfying the language task using bounding boxes: Red for structural clues, purple for sign understanding, and blue for detours. The expert demonstration path is drawn in cyan.

E.1 Oracle Critique Dataset Generation Procedure.

We prompt the oracle critic VLM (Gemini-3.1-Flash) by using a history of 4 image observations, annotating the last image with 5 motion plans sampled from our finetuned motion planner $\pi_\theta^{\text{motion}}$. Fig. 9 shows example annotated images and the corresponding critic prompt below.

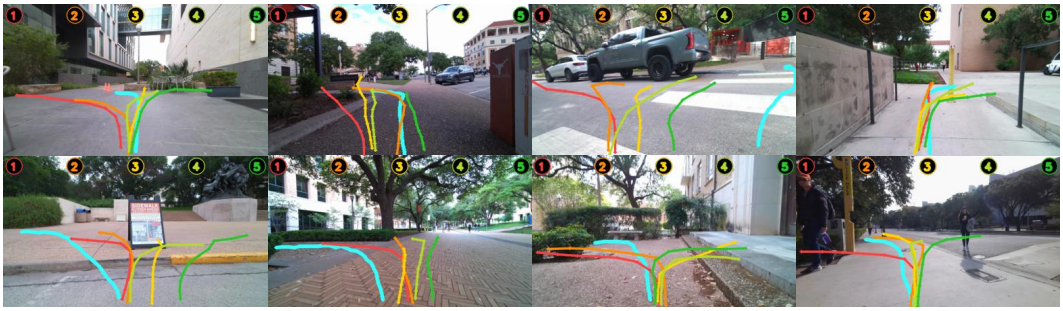

Figure 9: Qualitative examples of the the context images provided to the oracle critic VLM to use for generating critiques. Each motion plan is sampled from our motion planner and assigned an index. The critic VLM compared the plan with respect to the expert (cyan) and generates a short critique.

Critic Prompt

```
Attached is the egocentric robot image annotated with the planned path in yellow. The path coordinates
are normalized xy path coordinates:
```

```
<|motion_start|><|motion_end|>

Think crucially about the language instruction and the path to analyze if the path is safe and appears
to follow navigation cues that lead to the goal. If the path is acceptable, output "1" with the string
"good" for the reason. If the path is unacceptable, output "0" and describe SPATIALLY AND SEMANTICALLY
SPECIFIC VISIBLE CUES that are important to pay attention and how to improve them for key invalid
points. The reason string should be like the following:
<fix> </fix> [x_{i}, y_{i}]
...
<fix> </fix> [x_{i+N}, y_{i+N}]

Critique points to consider: obstacle collisions, path plans that are likely to lead to the goal,
traversable terrain for a pedestrian, or unrealistic robot navigation movements. Prescribe a direction
to move in the correction.
Constraints: Ensure that no points are above the sky horizon line

OUTPUT RULES (MUST FOLLOW):
- Output ONLY valid JSON on ONE LINE.
- Output must start with { and end with }.
- Use exactly two keys: "verdict" and "reason".
- Do not critique more than two points
- Do NOT write "Verdict:" or "Reason:" or any extra text.

JSON template:
{"verdict":"0|1","reason":"<short image-grounded reason>"}

Generate the critique now.
```

E.2 FORESIGHT Prompts used for Training and Inference

We provide the exact prompts used for planning, critic, and refinement roles below in Sec. E.2.1, Sec. E.2.2, and Sec. E.2.3 respectively.

E.2.1 Motion Planning Prompt

Motion Planning Prompt

```
Attached are egocentric navigation images from a robot navigating to a goal. The images are in
chronological order, where the last image is the current observation. Your task is to sample a unique
motion trajectory from the distribution of trajectories that follows the language instruction:
(<|language_goal|>)
while satisfying the following constraints.

Constraints:
- The trajectory must be in normalized pixel coordinates. Each point is [x,y] with 0<=x<=1, 0<=y<=1.
- Use == 10 points. The first point MUST start near the bottom of the image.
- Points must on the walkable ground, but can be behind obstacles if the obstacle is passable (like a
door or a wall).

Output ONLY JSON with exactly one key "trajectory". No extra text.
Format:
{"trajectory":[[x0, y0], ... [xn, yn]]}
```

E.2.2 Motion Critic Prompt

Critic Prompt

```
Attached is the egocentric robot image annotated with the planned path in yellow. The path coordinates
are normalized xy path coordinates:
<|motion_start|><|motion_end|>

Think crucially about the language instruction and the path to analyze if the path is safe and appears
to follow navigation cues that lead to the goal. If the path is acceptable, output "1" with the string
"good" for the reason. If the path is unacceptable, output "0" and describe SPATIALLY AND SEMANTICALLY
SPECIFIC VISIBLE CUES that are important to pay attention and how to improve them for key invalid
points. The reason string should be like the following:
<fix> </fix> [x_{i}, y_{i}]
...
<fix> </fix> [x_{i+N}, y_{i+N}]

Critique points to consider: obstacle collisions, path plans that are likely to lead to the goal,
traversable terrain for a pedestrian, or unrealistic robot navigation movements. Prescribe a direction
to move in the correction.
Constraints: Ensure that no points are above the sky horizon line

OUTPUT RULES (MUST FOLLOW):
```

```
- Output ONLY valid JSON on ONE LINE.
- Output must start with { and end with }.
- Use exactly two keys: "verdict" and "reason".
- Do not critique more than two points
- Do NOT write "Verdict:" or "Reason:" or any extra text.

JSON template:
{"verdict":"0|1","reason":"<short image-grounded reason>"}

Generate the critique now.
```

E.2.3 Motion Refinement Prompt

Motion Refinement Prompt

```
Reflect on the previous planned path and the motion plan critique. If the verdict is 1, output the
same path. If the verdict is 0, consider the issues mentioned in the critique to sample an improved
motion plan that fixes valid issues while following the language instruction:
(<|language_goal|>)
while satisfying the earlier constraints:
- The trajectory must be in normalized pixel coordinates. Each point is [x,y] with 0<=x<=1, 0<=y<=1.
- Use == 10 points. The first point MUST start near the bottom of the image.
- Points must on the walkable ground, but can be behind obstacles if the obstacle is passable (like a
door or a wall).

OUTPUT THE REFINED TRAJECTORY AS JSON IN THIS FORMAT:
{"trajectory":[[x0, y0], ... [xn, yn]]}
```

F Model Implementation Details

In this section, we provide the grounding policy implementation details in Sec. F.1, supervised pre-training details in Sec. F.2, reward model training procedure in Sec. F.3, and supplementary deployment controller and inference latencies in Sec. F.4.

F.1 Grounding Policy

We closely follow prior work [12] to implement our image-plan-conditioned grounding policy π_ψ^{gnd}. Let o_t be the current image, $o_{\zeta_K} \in \mathbb{R}^{H\times W\times 1}$ be a boolean image annotated with the FORESIGHT predicted plan, and $\zeta^{\text{xyz}} \in \mathbb{R}^{T\times 3}$ be a sequence of Cartesian XYZ waypoints. The grounding policy predicts Cartesian waypoints from the current observation and image-space plan as

$$\pi_\psi^{\text{gnd}}(\zeta^{\text{xyz}} \mid o_t, o_{\zeta_K}) = g_\psi\Big(\text{SSM}\Big(\Big[f^{\text{depth}}(o_t); f_\psi^{\text{plan}}(o_{\zeta_K})\Big]\Big)\Big), \tag{16}$$

where f^{depth} is a frozen DepthAnythingv2 [36] image encoder, f_ψ^{plan} is a finetuned EfficientNet-B0 [37] encoder for the image-space plan, $[\cdot;\cdot]$ denotes channel-wise feature concatenation, $\text{SSM}(\cdot)$ is a spatial softmax [38], and g_ψ is a transformer encoder that maps the resulting latent features to Cartesian waypoints.

Concretely, we encode o_t using the frozen DepthAnythingv2 encoder and o_{ζ_K} using the finetuned EfficientNet-B0 encoder. We stack the encoded features channel-wise and apply a spatial softmax to obtain latent features with dimension $K \times F$, where K is the number of keypoints and F is the feature dimension. Finally, we process the latent features with a transformer encoder to predict ζ^{xyz}. We empirically determine that setting $K = 384$, $F = 384$, and the number of transformer encoder layers/heads to 8 yields the lowest average mean squared error (MSE) of 0.018m on the test set. This error indicates that the average error for each waypoint in the trajectory is less than 2 centimeters, which is acceptable for our motion planning domain.

We train π_ψ^{gnd} on the SCAND and FORESIGHT datasets to predict the expert Cartesian waypoints conditioned on the expert image-space plan $\hat{\zeta}^{\text{img}}$. We obtain $\hat{\zeta}^{\text{img}}$ by projecting the robot odometry into image space using known camera intrinsics and extrinsics.

F.2 Supervised Pre-training

We initialize all FORESIGHT models and the reproduced Alpamayo [7] model from Qwen3-VL-2B-Instruct. We summarize the supervise finetuning model, dataset, and optimization parameters for

Table 2: Supervised finetuning training parameters.

Category	Value
Model	
Base model	Qwen3-VL-2B-Instruct
Parameters	1.8B total, 0.6M trainable
Trainable modules	LoRA on vision backbone, projector, and LLM
Planner/critic	Shared VLM backbone
LoRA	
Rank / alpha / dropout	64 / 64 / 0.0
Target modules / bias	`all-linear` / none
Input/output	
Image input	4 frames at 224×392
Trajectory output	10 normalized image-space waypoints in $[0, 1]^2$
Critic output	Binary verdict + free-form critique
Max sequence length	4096 / 8192 tokens
Max output length	192 motion tokens, 256 critic tokens
Dataset	
Training / validation examples	34,164 / 6,009
SFT targets	Plans, critiques, and critique-conditioned refinements
Optimization	
Optimizer / scheduler	AdamW / cosine decay
Learning rate / warmup / weight decay	1×10^{-4} / 0.0 / 0.01
Global / per-device batch size	512 / 256
Max epochs / grad. clipping	15 / 1.0
Precision	bfloat16
Compute	
Hardware / time	1× NVIDIA GH200 / ∼4 hours
Framework	Volcano Engine Reinforcement Learning (verl) [39]

FORESIGHT in Table 2. While we allow up to 15 training epochs, we find that all models trained via SFT typically converge and begin overfitting within 8 training epochs.

F.2.1 AlpaMayo Implementation Details

We train AlpaMayo on the same SFT dataset as FORESIGHT, finetuning the base Qwen3vl model to predict a structured Chain-of-Causation (CoC) thinking trace first before predicting the image space motion plan. We use the following prompt for generating the oracle CoC traces using Gemini-3.1-Flash. We generate oracle traces for the same dataset split as was used for training the FORESIGHT critic for fairness.

Oracle Alpamayo CoC Thinking Prompt

```
Attached are egocentric navigation images from a robot navigating to a goal. The images are in
chronological order, where the last image is the current observation. The robot must follow the
language instruction:
(<|language_goal|>)

The ground-truth motion plan has been drawn directly on the current observation as a cyan/teal
polyline connecting the robot's planned waypoints from its current position toward the goal. Use this
overlay as the authoritative reference for what the correct next action is: your reasoning must be
consistent with and explain why this specific path was chosen given the scene. When generating the
reasoning trace, do not mention the motion play overlay in your reasoning.

Generate a structured chain-of-causation reasoning trace that can be used to condition a subsequent
motion plan, so it must commit to a concrete decision and clearly identify the causal factors driving
it.

Field definitions:
- scene_caption: short caption describing traversable surfaces, doorways, dynamic obstacles, and
lighting in the current observation.
- entities: grounded list of up to three distinct visual entities visible in the current image. Each
entity has a "name" (snake_case noun) and a single "location" pixel point [x, y] in normalized
coordinates with 0<=x<=1 and 0<=y<=1, marking the centroid of the entity.
```

```
- clues: chain-of-causation dictionary identifying which subset of entities causally drives the next
decision. Keys MUST exactly match names from "entities". Values are short role descriptions (e.g.,
"target landmark for current subgoal", "dynamic obstacle, keep safe distance from"). List up to three
clues.
- spatial_reasoning: one short sentence explaining the spatial relationship the agent must respect,
grounded in the visible cyan motion-plan overlay (e.g., "the plan curves left around the pedestrian
toward the open corridor entrance").
- meta_action: discrete decision pair with two keys: "longitudinal" in {stop, slow, maintain,
accelerate} and "lateral" in {keep, turn_left, turn_right, sidestep_left, sidestep_right}. MUST be
consistent with the direction of the drawn motion-plan polyline.
- rationale: one sentence linking the selected clues to the chosen meta_action causally and to the
language instruction.

Constraints:
- Only reference entities visible in the current observation; do not hallucinate landmarks.
- Pixel locations must lie on the entity in the image and must not be above the sky horizon line. Each
point is [x,y] with 0<=x<=1, 0<=y<=1.
- "clues" keys MUST be a subset of "entities" names.
- "rationale" MUST reference at least one clue, the chosen meta_action, the language instruction, and
why the visible motion plan leads through the identified entities.
- Use SPATIALLY AND SEMANTICALLY SPECIFIC VISIBLE CUES; avoid vague descriptions such as "be cautious"
or "watch out".
- Do not list weather, road type, or generic rule-based factors as causes unless they are directly
visible and decision-driving.

OUTPUT RULES (MUST FOLLOW):
- Output ONLY valid JSON.
- Output must start with { and end with }.
- Use exactly six keys: "scene_caption", "entities", "clues", "spatial_reasoning", "meta_action",
"rationale".
- Do NOT exceed three entries in "entities" or three entries in "clues".
- Do NOT write field labels, commentary, code fences, or any extra text outside the JSON.

JSON template:
{
  "scene_caption": "<short caption>",
  "entities": [
    {"name": "<snake_case>", "location": [x, y]}
  ],
  "clues": {
    "<entity_name>": "<causal role>"
  },
  "spatial_reasoning": "<one sentence>",
  "meta_action": {
    "longitudinal": "stop|slow|maintain|accelerate",
    "lateral": "keep|turn_left|turn_right|sidestep_left|sidestep_right"
  },
  "rationale": "<one sentence>"
}

Generate the reasoning trace now.
```

Then, we use the following prompt when finetuning the VLM to predict the CoC trace. We use the same motion prompt as FORESIGHT for predicting the motion plan.

Training/Inference Alpamayo Thinking Prompt

```
Attached are egocentric navigation images from a robot navigating to a goal. The images are in
chronological order, where the last image is the current observation. The robot must follow the
language instruction:
(<|language_goal|>)

Generate a structured chain-of-causation reasoning trace that can be used to condition a subsequent
motion plan, so it must commit to a concrete decision and clearly identify the causal factors driving
it.

Field definitions:
- scene_caption: short caption describing traversable surfaces, doorways, dynamic obstacles, and
lighting in the current observation.
- entities: grounded list of up to three distinct visual entities visible in the current image. Each
entity has a "name" (snake_case noun) and a single "location" pixel point [x, y] in normalized
coordinates with 0<=x<=1 and 0<=y<=1, marking the centroid of the entity.
- clues: chain-of-causation dictionary identifying which subset of entities causally drives the next
decision. Keys MUST exactly match names from "entities". Values are short role descriptions (e.g.,
"target landmark for current subgoal", "dynamic obstacle, keep safe distance from"). List up to three
clues.
- spatial_reasoning: one short sentence explaining the spatial relationship the agent must respect
(e.g., "hallway entrance is to the left, behind the pedestrian; wait for clearance before turning").
```

```
- meta_action: discrete decision pair with two keys: "longitudinal" in {stop, slow, maintain,
accelerate} and "lateral" in {keep, turn_left, turn_right, sidestep_left, sidestep_right}.
- rationale: one sentence linking the selected clues to the chosen meta_action causally and to the
language instruction.

Constraints:
- Only reference entities visible in the current observation; do not hallucinate landmarks.
- Pixel locations must lie on the entity in the image and must not be above the sky horizon line. Each
point is [x,y] with 0<=x<=1, 0<=y<=1.
- "clues" keys MUST be a subset of "entities" names.
- "rationale" MUST reference at least one clue, the chosen meta_action, and the language instruction.
- Use SPATIALLY AND SEMANTICALLY SPECIFIC VISIBLE CUES; avoid vague descriptions such as "be cautious"
or "watch out".
- Do not list weather, road type, or generic rule-based factors as causes unless they are directly
visible and decision-driving.

OUTPUT RULES (MUST FOLLOW):
- Output ONLY valid JSON.
- Output must start with { and end with }.
- Use exactly six keys: "scene_caption", "entities", "clues", "spatial_reasoning", "meta_action",
"rationale".
- Do NOT exceed three entries in "entities" or three entries in "clues".
- Do NOT write field labels, commentary, code fences, or any extra text outside the JSON.

JSON template:
{
  "scene_caption": "<short caption>",
  "entities": [
    {"name": "<snake_case>", "location": [x, y]}
  ],
  "clues": {
    "<entity_name>": "<causal role>"
  },
  "spatial_reasoning": "<one sentence>",
  "meta_action": {
    "longitudinal": "stop|slow|maintain|accelerate",
    "lateral": "keep|turn_left|turn_right|sidestep_left|sidestep_right"
  },
  "rationale": "<one sentence>"
}

Generate the reasoning trace now.
```

F.2.2 LeLaN Implementation Details

We initialize LeLaN using the open-source pre-trained checkpoint to ensure the base model contains internet-pretrained navigation knowledge. The original LeLaN model predicts a horizon of linear and angular velocities $[v, w] = \mathbb{R}^{T\times 2}$ and convert this to a sequence of Cartesian xy waypoints using Euler integration. We finetune the model end-to-end to predict linear and angular velocities on the same SCAND and FORESIGHT dataset split as for FORESIGHT. We also convert the model predictions to Cartesian xy waypoints using Euler integration and track these using the same low-level motion controller described in Appendix Sec. F.4.

F.3 Preference Reward Learning

In this section, we provide additional details for the trajectory annotation tool in Sec. F.3.1 and training procedure for the learned reward model R_ϕ in Sec. F.3.2.

F.3.1 Trajectory Ranking Procedure.

Fig. 10 shows the web tool used for ranking candidate motion plans. In the tool, the human annotator is shown the observation history, natural language task, and ground truth image plan. The annotator can use the Rollout button to sample sets of candidate plans from the finetuned VLM motion planner $\pi_\theta^{\text{motion}}$. Then, the annotator uses the dropdown to rank motion plans on a Likert scale from 1 (most preferred) to 5 (least preferred). For this work, we use a single human annotator and rank the trajectories relative to each other rather than using an absolute scale. We allow ranking ties, but do not use this for reward learning. In total, we annotate 1150 samples with 5 trajectories for each sample.

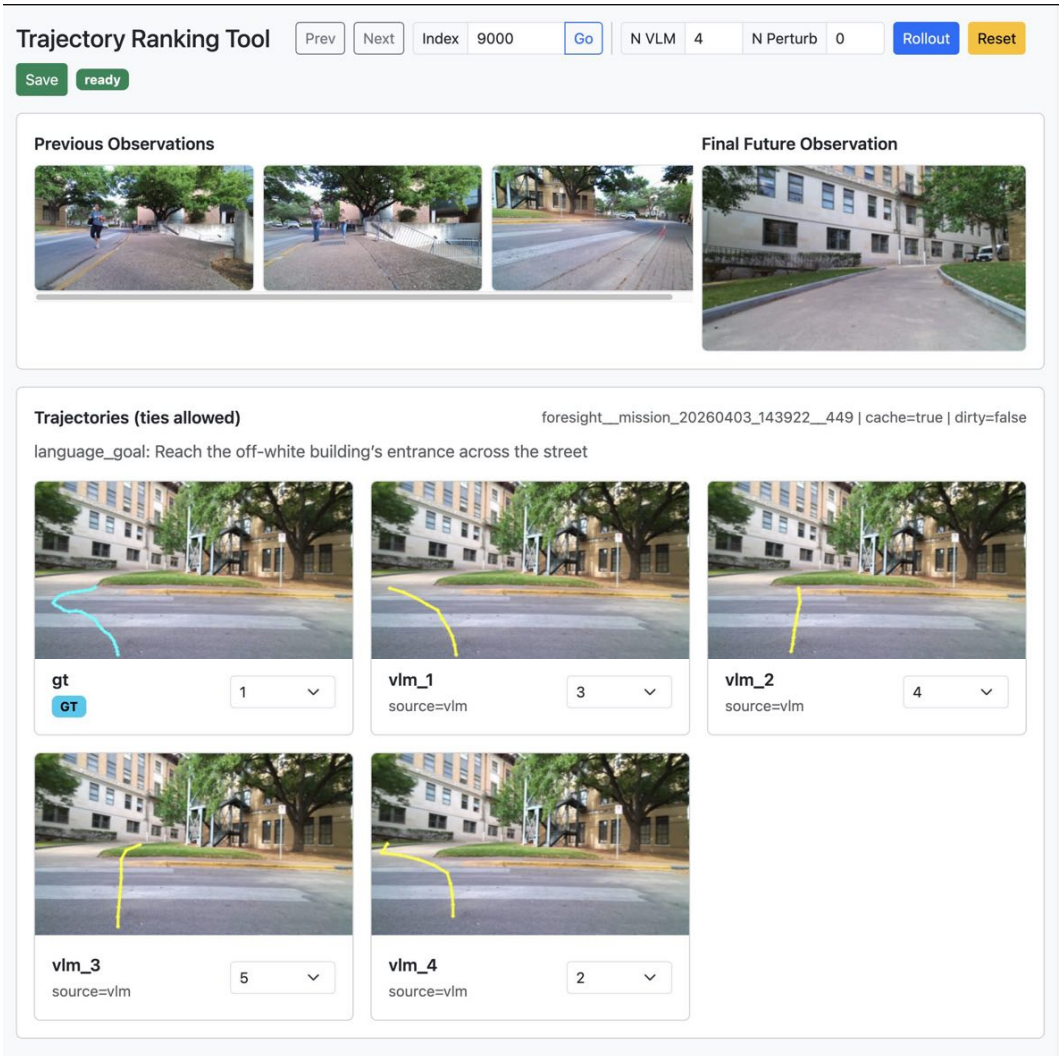

Figure 10: Web tool used for ranking motion plan candidates. The human annotator is shown the observation history, language goal, ground truth image plan (cyan), along with K alternate plans samples from our VLM motion planner.

F.3.2 Reward Learning Details.

Following prior work on finetuning VLMs as reward models [13], we initialize a linear layer that predicts a scalar reward from the penultimate feature from the langauge model backbone. We initialize our reward model from π_θ, our model that has already been supervise finetuned for planning, critiquing, and refinement. Empirically, we observe that initializing from the SFT-ed is necessary to prevent overfitting compared to using the base Qwen3-VL-2B-Instruct model. We prompt the reward model using the same context as the critic policy $\pi_\theta^{\text{critic}}$, which contains the predicted motion plan in the context history.

F.3.3 Reinforcement Learning Details.

We provide the model architecture, training hyperparameters, and dataset details in Table 3. For this work, we limit the maximum number of refinement steps $K = 1$ as we empirically observe in that the model sees the most improvement on the first refinement round during supervised pre-training.

In regards to hyperparameter settings, we find that while the model is not particularly sensitive to the reward weighting parameters, the qualitative performance of the model is better when the reward weighting is biased more in factor of R_ϕ compared to R_{exp}. In addition, we observe that while accumulated reward continues to increase steadily without plateauing until roughly the end of

Table 3: GRPO reinforcement learning training parameters.

Hyperparameter	**Value**
Initialization	SFT checkpoint Sec. F.2
Trainable modules	LoRA on vision backbone, projector, and language backbone
Reference policy	Frozen SFT checkpoint Sec. F.2
Reward model	Frozen preference reward model Sec. F.3
Algorithm	Group Relative Policy Optimization (GRPO)
Rollout unit	Initial Plan ζ_0, Critique z_0, Refined Plan ζ_1
Group size G	8 rollouts per prompt
Max refinement steps K	1
Reward weighting	0.8 (R_ϕ), 0.2 (R_{exp})
Advantage normalization	Group-relative normalization
Loss aggregatation	Sequence mean, then token mean
KL coefficient β	0.01
Clip range ϵ	0.2
Policy learning rate	8×10^{-5}
Optimizer / scheduler	AdamW / cosine decay
Global batch size	64 prompts, 512 rollouts total
Mini-batch size	16
PPO/GRPO epochs	10
Sampling temperature	1.0
Top-p	0.95
Max sequence length	4096 / 8192 tokens
Max output length	192 motion tokens, 256 critic tokens
Gradient clipping	1.0
Precision	bfloat16
Hardware / time	1× NVIDIA GH200 / 12 hours
Framework	Volcano Engine Reinforcement Learning (verl) [39]

epoch 7, the qualitative performance is best after 2-3 epochs of training. We hypothesize this occurs because we use the same motion planner for sampling the initial plan and for plan refinement. As a result, the motion planner begins to ignore the critiques later in training because the initial motion plan sampled does not require refinement. As mentioned in the limitations section, we acknowledge this is caused by a lack of credit assignment to incentivize following the critique and believe it can be addressed by computing auxiliary rewards to penalize ignoring the critique.

F.4 Deployment Implementation Details

This provides additional details on the compute hardware, inference latency benchmarking, and trajectory execution procedure.

We perform synchronous model inference and control solely using the onboard Nvidia AGX Orin (64GB) with a 12-core Arm Cortex A48AE CPU. On average across two environments, FORESIGHT requires 3.05 seconds to do inference for each plan-critique loop, where 1.66 seconds are spent generating the plan and 1.39 seconds for the critique. Alpamayo takes 5.82 seconds to do inference, spending 1.77 seconds on planning and 4.05 seconds to generate the Chain-of-Causation reasoning trace.

We standardize all model baselines to output a sequence of 10 Cartesian xyz waypoints. Because LeLaN originally predicts a horizon of linear and angular velocity commands, we integrate these predictions to obtain a trajectory and resample this trajectory to 10 waypoints. We track all waypoint plans using the same pure pursuit motion controller. Our navigation stack automatically follows each plan to the last waypoint, aligns the final heading with the heading angle between the penultimate and final waypoint, and re-queries the model for a new motion plan. This process is repeated until the robot reaches the goal or until the robot gets stuck and requires an intervention. For each intervention, we reset the robot to the closest nearby position and heading angle such that at least one visual clue is present for determining the future motion plan direction.