GemNav: Discrete-Token Visual Robot Navigation using a Multimodal Large Language Model

Peter Bohm*, Saimunur Rahman, Abdelwahed Khamis,
Sagun Man Singh Shrestha, Chris McCool, Peyman Moghadam
CSIRO Technology, Australia

Abstract:
Visual navigation policies built on large pretrained models have so far followed a common recipe: a dedicated visual encoder, a bespoke action head, and training on thousands of hours of cross-embodiment datasets. We ask whether this recipe is necessary. In this paper, we introduce *GemNav*, a visual robot navigation policy that adapts a frozen Multimodal Large Language Model (MLLM) for short-to-medium horizon waypoint navigation using Low-Rank Adaptation (LoRA) on the language tower alone, with no auxiliary visual encoder and no continuous regression head. Waypoints and categorical navigation signals share a single discrete token vocabulary generated by the language-model head, and a soft-decoded auxiliary loss recovers the metric structure that pure cross-entropy training discards. On a single 8.7-hour open corpus, roughly three orders of magnitude smaller than competing training sets, the policy transfers zero-shot to four physically distinct unseen environments and stops within 0.25–0.42 m of the goal across 20 real-world trials covering an open carpark, an obstacle carpark, a long outdoor chemical yard, and an indoor warehouse. Conditioning on short image histories improves offline metrics but yields no robot benefit, pointing to a ceiling on what temporal context adds once pretrained vision features are in place. These results indicate that discrete-token adaptation of frozen MLLMs can provide a data-efficient, deployable alternative for foundation model robot navigation.

Keywords: Visual Navigation, Data-Efficient Robot Learning, Multimodal Large Language Model (MLLM)

1 Introduction

Robot navigation has traditionally been built around a tightly coupled stack of perception, state estimation, and control, tuned for a particular robot, sensor setup, or environment [1]. Large pretrained vision-language-action models have shifted this framing, treating navigation as a cross-embodiment learning problem: a single policy trained on hundreds to thousands of hours of trajectories pooled across robots, environments, and sensor configurations [2, 3, 4, 5, 6].

Recent visual navigation policies built on large pretrained models follow a common recipe: a strong visual backbone (often a separate dedicated encoder) is paired with a continuous-action regression head and supervised on a union of multiple navigation datasets collected across embodiments [5, 4, 7]. Yet the scalability of this approach raises a deeper question about where the true source of generalization lies. These policies achieve expansive transferability, but their success has come bundled with an ever-growing need for large demonstration data, specialized action heads, and visual encoders trained from scratch on robotic data. Recent benchmarking further suggests that the visual encoder is the dominant bottleneck for downstream VLA performance [8, 9], pointing toward a trajectory of increasing visual scale as the primary source for improvement. In this paper, we take a different view. The pretraining of modern multimodal large language models [10, 11, 12, 13] has

*Corresponding author: peter.bohm@csiro.au

already produced visual representations of remarkable breadth and robustness; the question we pose is whether those representations, held frozen, are sufficient for robot navigation, and if the gap between a general-purpose Multimodal Large Language Model (MLLM) and a deployable navigation policy can be closed by rethinking the action representation rather than by scaling perception further.

We ground this question in short-to-medium-horizon waypoint navigation: a higher-level planner supplies the next local goal as a 2D pose, a goal image, or both, and the policy predicts a short robot-frame trajectory before the planner re-queries. This setting covers most structured deployments, defines success at the scale of approximately one meter, and pairs naturally with conventional collision avoidance and re-planning without the learned policy reasoning about global topology.

To investigate this, we present *GemNav*, a visual navigation policy that adapts a pretrained Multimodal Large Language Model (MLLM) for waypoint navigation while keeping its visual backbone frozen, formulating navigation as multimodal sequence prediction in which observations and goal specifications map to discrete action tokens from the language-model head. Aligning metric waypoints and navigation-state decisions with the model's native token interface tests whether pretrained multimodal grounding can be made actionable without scaling or retraining the visual encoder. *GemNav* is trained on a single open dataset of approximately 8.7 hours, roughly three orders of magnitude smaller than the cross-embodiment datasets used by OmniVLA [5], with no manually collected or simulation data, and evaluated in closed-loop deployment.

In summary, the main contributions of this paper are as follows:

- We test whether the vision-encoder bottleneck identified in recent VLA benchmarking can be bypassed for waypoint navigation by relying on the pretrained MLLM grounding rather than retraining the vision tower or adding an auxiliary visual encoder. Real-robot results support this: the frozen-vision policy reaches goals on all 20 pose trials across four unseen environments, including long outdoor yards and indoor warehouses.
- We introduce a discrete value tokenization that unifies continuous waypoint prediction and categorical stop signals (`<goal_reached>` and `<goal_unreachable>`) in a single 18-token sequence produced by the language-model head, together with a soft-decoded auxiliary loss that restores metric structure to the bin distribution and reduces validation displacement error by 23% over standard cross-entropy.
- We report a concrete negative result alongside the positive ones: a two- or four-frame history window improves offline validation metrics but provides no deployment benefit over a single-frame policy, suggesting temporal context yields diminishing returns when the pretrained vision features are already strong.

2 Related Work

Cross-embodiment visual navigation. Navigation foundation models train dedicated policies on robot trajectories: GNM learns a shared action space across six robots [14], ViNT uses a CNN–Transformer architecture [2], NoMaD unifies exploration and goal reaching through diffusion [3], and NavFoM scales to 8M samples across sensor configurations [4]. OmniVLA supports language, pose, and image goals via an L1 regression head trained on 9,500 hours [5]. JanusVLN [15] couples a frozen Qwen2.5-VL backbone with a frozen 3D-geometry encoder for continuous VLN. *GemNav* instead applies LoRA to a pre-trained MLLM with a frozen vision tower and no auxiliary 3D encoder, generating waypoints and categorical signals through the native LM head.

Data and navigation supervision. LeLaN augments action-free video with VLM-generated language and counterfactual actions [16]. CityWalker [17] further scales this by learning urban navigation from web-scale video data. CAST uses counterfactual hindsight relabeling to address language posterior collapse [18]. Both are complementary to *GemNav*, which uses pose and image goals with cross-trajectory negatives for visually disconnected targets. We train on the image and pose streams of SCAND, an ~ 8.7-hour teleoperated Spot/Jackal dataset [19], a data-efficient test of frozen visual features relative to OmniVLA's cross-embodiment corpus.

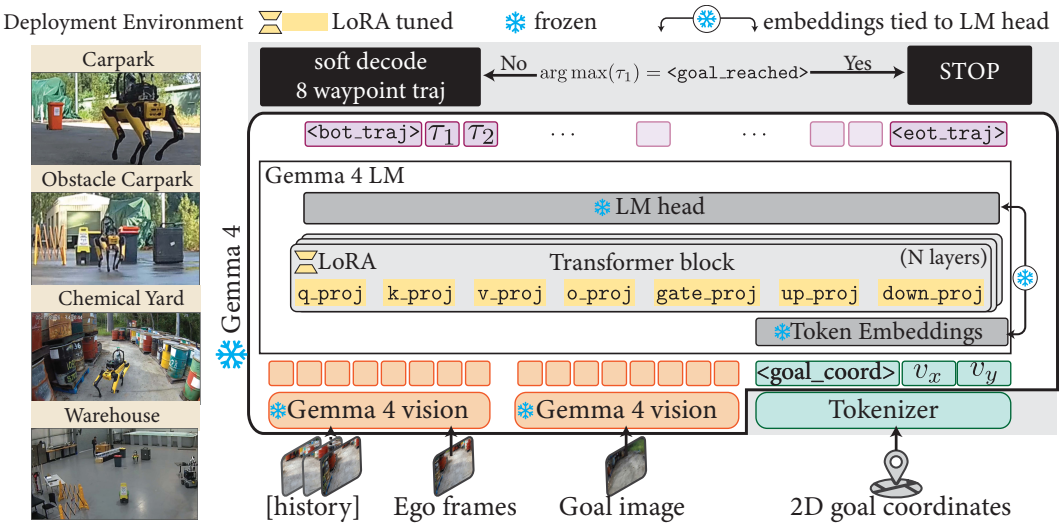

Figure 1: *GemNav* architecture. Inputs: current camera view, optional goal image and/or 2D goal coordinate, and (grayed, not yet implemented) a language prompt. Images pass through Gemma 4's frozen vision tower; 2D goal coordinates reuse the trajectory value-bin alphabet, prefixed by `<goal_coord>`. The language model is adapted via LoRA on its linear layers (r=32, α=16). Outputs are an 8-waypoint trajectory (`<bot_traj>` + 16 value-bin tokens + `<eot_traj>`, decoded to body-frame (x, y) via bin centers) or a stop signal (`<goal_reached>` / `<goal_unreachable>`).

VLM to action policies. VLM2VLA adapts a pre-trained VLM with LoRA and represents continuous actions as natural language [20]; *GemNav* shares the action-as-tokens premise but uses a compact value-bin alphabet with explicit `<goal_reached>` / `<goal_unreachable>` tokens. Dita [21] denoises continuous actions iteratively; *GemNav* produces continuous-valued waypoints in a single decode via soft-decoded value bins through the stock LM head. VLM4VLA identifies visual representation quality as a key determinant of VLA performance [8]; we instead test whether a frozen pretrained vision tower suffices for robot navigation. BridgeVLA aligns point-cloud manipulation inputs with a 2D VLM backbone [22], a complementary task and modality.

3 Method

We introduce *GemNav*, a visual navigation policy that adapts a frozen Multimodal Large Language Model (MLLM) for short-to-medium horizon waypoint navigation using LoRA on the language tower alone; there is neither auxiliary visual encoder nor a continuous regression head. We then cast visual navigation as conditional sequence prediction by predicting the next eight waypoints in the robot's local frame. We additionally introduce two special tokens `<goal_reached>` and `<goal_unreachable>` to indicate when the goal conditions are met or not.

3.1 Problem formulation

At each control step, the high-level planner provides a goal specification g which may take three forms in *GemNav* pipeline: (a) a goal image I_g (egocentric view from the target pose), (b) a 2D goal coordinate $\mathbf{p}_g = (x_g, y_g) \in \mathbb{R}^2$ in the robot-local frame, or (c) both, denoted $g \in \{(I_g, \cdot), (\cdot, \mathbf{p}_g), (I_g, \mathbf{p}_g)\}$. We refer to these as the *ego*, *pose*, and *ego+pose* goal modalities, for consistency with OmniVLA evaluation protocol [5].

The current view is an ordered sequence of $h{+}1$ egocentric frames $\mathbf{I}_c = (I_t, I_{t-\delta}, \ldots, I_{t-h\delta})$, where $h \geq 0$ is a configurable history depth and δ is a per-corpus frame stride. $h = 0$ reduces $\mathbf{I}_c$ to the current frame, and we also experiment with $h \in \{2, 4\}$. Given $\mathbf{I}_c$ and goal g, the policy outputs either an eight-waypoint trajectory $\tau = \big((x_k, y_k)\big)_{k=1}^{8}$ in the body frame at step t (so it never reasons about global coordinates), or a stop signal $s \in \{$`<goal_reached>`, `<goal_unreachable>`$\}$. The former

is when the robot is within the goal radius, the latter when the goal image is visually disconnected from the current view (a refuse-to-plan signal for cross-trajectory negatives, Section 4.1).

3.2 Trajectory tokenization and prompt construction

Rather than attaching a regression head to the LM hidden states, we extend the model's vocabulary so that trajectories are first-class token sequences. We allocate 69 slots from Gemma 4's reserved `<unusedN>` token range: $K = 64$ *value-bin tokens* plus five role tokens (`<bot_traj>`, `<eot_traj>`, `<goal_reached>`, `<goal_coord>`, `<goal_unreachable>`). Each waypoint axis is independently quantized to one of K uniform bins on $[-B, B]$ with $B = 15$ m; the bin index and its representative center are

$$b(v) = \min\big(K-1,\ \lfloor \tfrac{v+B}{2B} \cdot K \rfloor\big), \qquad c(b) = -B + (b + \tfrac{1}{2}) \cdot \tfrac{2B}{K}. \tag{1}$$

With these values, each bin spans 0.469 m and the half-bin reconstruction error is 0.234 m per axis. The tokenization hyperparameters, including $K = 64$ and $B = 15$ m, are specified in Appendix K. The target sequence for a trajectory sample is:

$$\mathcal{T}(\tau) = \big[\texttt{<bot\_traj>},\ v_{b(x_1)}, v_{b(y_1)},\ \ldots,\ v_{b(x_8)}, v_{b(y_8)},\ \texttt{<eot\_traj>}\big], \tag{2}$$

of length 18 tokens for $N = 8$ waypoints, plus `<bot_traj>` and `<eot_traj>`. Stop samples output the three-token sequence $[\texttt{<bot\_traj>}, s, \texttt{<eot\_traj>}]$, wrapped as trajectory targets, so the decoder processes all outputs through a uniform interface. Encoding the 2D goal pose with the same tokens the model produces as output (`<goal_coord>`) ensures the Language Model (LM) encounters no representational gap between conditioning and generation, which is what enables the zero-shot pose-plus-image performance reported in Section 5.

To construct our prompt, a single training example is rendered as a Gemma-4 user-turn prompt: a brief task description, the $h + 1$ ordered image frames (current plus optional history, oldest-to-newest), an optional goal view, an optional goal-coordinate token sequence prefixed by `<goal_coord>`, and a final `Trajectory:` marker that triggers the assistant turn. The target sequence $\mathcal{T}$ is appended as token IDs to preserve the learned `<unusedN>`-derived embeddings. Image inputs are resized to 224×224 and capped at 256 visual tokens each via Gemma 4's native image tokenizer, so visual-token cost grows linearly with image count in the prompt. The verbatim prompt template is provided in Appendix C.

3.3 Model training and adaptation

We adapt the public Gemma-4-E2B-it model with Low-Rank Adaptation (LoRA) [23] (the larger E4B variant is provided as ablation studies). LoRA is applied to all $\{\texttt{q\_proj}, \texttt{k\_proj}, \texttt{v\_proj}, \texttt{o\_proj}, \texttt{gate\_proj}, \texttt{up\_proj}, \texttt{down\_proj}\}$ linear layers in the language tower ($r = 32$, $\alpha = 16$, dropout 0). The vision tower remains frozen, with no LoRA targets, unfrozen parameters, or auxiliary encoder. Unlike VLAs that attach a separate vision backbone to a language model [24, 8], we use the pre-trained model's existing multimodal alignment. Trajectories are predicted through the stock LM head, with no additional heads or projections.

The policy is supervised by token-level cross-entropy $\mathcal{L}_{\text{CE}}$ on the target sequence (Eq. 2), with prompt positions label-masked to -100 and stop / unreachable tokens trained by the same cross-entropy as value bins, so the model can refuse to plan when appropriate. We augment this with an auxiliary soft-decoded regression term that reintroduces the ordinal structure pure cross-entropy discards on the metric value bins. Let $\ell^{(k,a)} \in \mathbb{R}^V$ denote the logits over the model's full token vocabulary (size V) at the position whose target token is $v_{b(a_k)}$, with $a \in \{x, y\}$, and $\mathbf{c} \in \mathbb{R}^K$ the vector of bin centers from Eq. 1. Restricting to the K value-bin token IDs and softmaxing yields $\pi^{(k,a)} \in \Delta^{K-1}$, and the soft-decoded prediction is

$$\hat{a}_k = \mathbf{c}^\top \pi^{(k,a)} = \sum_{j=1}^{K} c_j \cdot \pi_j^{(k,a)}. \tag{3}$$

The auxiliary loss

$$\mathcal{L}_{\text{aux}} = \frac{1}{2N} \sum_{k=1}^{N} \sum_{a \in \{x,y\}} \left(\hat{a}_k - a_k\right)^2, \tag{4}$$

is computed only on trajectory samples (stop and unreachable samples are skipped). Adjacent bin errors are penalized lightly, while distant bin errors incur larger penalties. The total training loss is $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}$ with $\lambda_{\text{aux}} = 0.1$ throughout. Unlike continuous-action VLAs that regress through a dedicated action head, this term decodes through the existing value-bin tokens via Eq. 3, preserving the discrete-token interface without adding parameters; the ablation in Appendix E.1 shows it reduces ego-only validation ADE by 23% relative to pure cross-entropy.

4 Experimental Setup

4.1 Data pipeline

We train on SCAND [19], using only its image and pose information. Each sample comprises a trajectory $\mathcal{D}$, a current frame index t, and a goal frame index $t_g > t$. Eight target image waypoints are extracted at $t + k\sigma$, $k = 1, \ldots, 8$ (with σ the per-corpus waypoint spacing in frames). These are transformed into the body frame at time t. The 2D goal coordinate, when used, is the same body-frame transform applied to the world-frame position at t_g. The goal index t_g is selected by a *distance*-based sampler: we draw a target arc-length $d \sim \text{Uniform}(0.5, 14)$ m and advance along the trajectory until the world-frame displacement from $\mathbf{p}_t$ first exceeds d. The resulting t_g generally does not coincide with any waypoint frame, so the training horizon spans the full deployment-relevant distance range rather than a fixed frame count. The legacy frame-aligned sampler ($t_g = t + \text{offset} \cdot \sigma$, $\text{offset} \sim \text{Uniform}\{1, \ldots, 8\}$) is reported as an ablation in Appendix F. Image history, sample filters and augmentation, manifest construction, and the dataset splits are in Appendix K.

4.2 TokenWalker dataset (TW)

The TokenWalker (TW) corpus is a navigation dataset we collected to serve as an out-of-distribution validation set complementary to SCAND. It consists of 5 recording sessions captured outdoors over 3 days, approximately 83 minutes of teleoperation, and 9,924 synchronized image and pose samples. Data was recorded on a Boston Dynamics Spot quadruped with an Intel RealSense color camera streaming 1280×720 frames at 2 Hz; per-frame pose telemetry (3D position plus roll/pitch/yaw and the corresponding unit quaternion) is logged alongside each image. We will release the synchronized image streams, pose ground truth, and evaluation splits to support reproducibility and future out-of-distribution evaluation.

4.3 Robot platform

We evaluate on the same Boston Dynamics Spot + Intel RealSense color-camera configuration used to collect the TokenWalker dataset. 1280×720 frames at 2 Hz, scaled/cropped to 224×224 by the streamer's runtime preprocessor (the flagship runs use *stretch* mode, a full-frame rescale; see Appendix E.3 for the three modes we evaluate) before being passed to the model.Body-frame pose is recovered online from a continuous-time 3D LiDAR-inertial SLAM system [25]. provenance and lighter production alternatives are in Appendix L.

4.4 Models, baselines, and modalities

Deployed policy. Our deployed policy is the single-frame ($h = 0$) step-4 (final-phase, C_2) checkpoint of the four-phase warm-start chain (Appendix P), a Gemma-4-E2B-it model with SCAND-only `_d` LoRA training (run identifier in Appendix L), run with *stretch* crop and continuous (softmax-expectation over value bins) decoding. The same configuration is used across all results reported in this paper in Section 5.2. Other configuration details are reported as ablations in Section 6. We evaluate three goal modalities, *ego* (image-only), *pose* (coordinate-only), and *ego+pose* (both), with coverage varying by environment (Section 4.5).

Table 1: Per-modality validation performance of the deployed single-frame checkpoint. ADE in m on the 500-sample evaluation subsets; stop accuracy on the corresponding <goal_reached> subset. Lower is better.

Split	Pose	Ego+Pose	Ego	Combined	Stop Accuracy
In-distribution (SCAND)	0.37	0.33	0.76	0.49	50/50
Out-of-distribution (TW)	0.78	0.75	2.09	1.19	80/108

Baselines. We compare against OmniVLA [5], a published cross-embodiment navigation VLA, and a human teleoperator reference. For OmniVLA, we deploy the released checkpoint under its native inference stack. Because OmniVLA does not support fusing a goal image with a goal coordinate, the ego+pose modality is structurally absent; its post-hoc reconstruction of time, path length, and displacement is described in Appendix L. For the human-operated baseline, an operator remotely drives the same robot to the goal, providing a soft upper bound on achievable time, path length, and stopping precision. We collect this reference only in the two long, complex environments, since the Carpark environments are trivial for a human and therefore uninformative as a comparison.

4.5 Deployment Environments

We deploy *GemNav* model in four increasingly difficult environments, all distinct from training scenes: *Carpark* (open space with an unobstructed route, $\sim$ 10–12 m; base capability; all modalities), *Obstacle Course Carpark* (same layout with added obstacles, $\sim$ 18 m; short-range avoidance; pose only), *Outdoor Chemical Yard* (outdoor, $\sim$ 30–34 m; narrow passages and dead-ends; pose and ego+pose), and *Indoor Warehouse* (indoor, $\sim$ 32 m; obstacle course; pose and ego+pose). Goal distance is the straight-line distance from the robot's starting position to the goal. When ego and pose modalities use different goal targets, we report it as a range rather than a single value.

4.6 Protocol and metrics

For each model, goal modality, and deployment environment, we conduct $n = 5$ independent trials. We report *Success rate* (SR) as a *stop/partial/miss* tally. A *stop* denotes a trial in which the policy autonomously emits <goal_reached> within the accepted goal region, and is counted as a successful trial. A *partial* denotes a trial in which the robot reaches the vicinity of the goal but does not issue a stop signal, for example, by oscillating near the goal, driving past it, or requiring operator termination. A *miss* denotes a trial in which the robot does not reach the goal vicinity. We also report the elapsed time *Time* t (s) in seconds, *path length* d (m) in meters, and *final displacement* Δ (m) in meters. For *GemNav* and the human teleoperator, these quantities are averaged over successful trials. For OmniVLA, which does not autonomously generate stop signals, t and d are averaged over all trials, and Δ is reported as the Euclidean distance from the stopping pose to the goal. For offline validation, we report average and final displacement error (ADE and FDE) [26, 27], the mean Euclidean distance between the predicted and ground-truth waypoint trajectories over all predicted waypoints (ADE) and at the final waypoint only (FDE). We also report stop accuracy on samples whose target output is <goal_reached>. Additional metrics are provided in Appendix L, and the offline validation protocol is described in Section 5.1.

5 Results

5.1 Validation Performance

We evaluate offline performance on two modality-balanced validation subsets. The in-distribution split is drawn from SCAND, using a 10% trajectory-disjoint hold-out of approximately 29,000 samples. The out-of-distribution split is drawn from TokenWalker (TW), which is held out entirely from training and contains approximately 9.9k validation samples. For both splits, we evaluate on fixed 500-sample subsets with balanced coverage across the ego, pose, and ego+pose modalities. Filtering and augmentation follow Section 4.1, and per-modality results are reported in Table 1.

Table 2: Real-world deployment, SR = stop / partial / miss over $n = 5$ trials ($n = 4$ for Warehouse OmniVLA). Human expert performance is reported as an upper-bound reference where available. For OmniVLA, Δ shows mean final displacement with mean nearest approach in parentheses, and t / d are over all trials; Human SR is task completion. Bold marks the best results.

Environment	Model	SR ↑	t (s) ↓	d (m) ↓	Δ (m) ↓
Carpark	OmniVLA	$0/0/5$	8.7 ± 0.7	7.28 ± 0.17	$10.44\ (9.54)$
	GemNav (Ours)	$\mathbf{5/0/0}$	$\mathbf{17.5\pm1.1}$	$\mathbf{11.98\pm0.36}$	$\mathbf{0.39\pm0.11}$
Obstacle Carpark	OmniVLA	$0/0/5$	28.6 ± 2.5	20.58 ± 2.32	$3.10\ (0.77)$
	GemNav (Ours)	$\mathbf{5/0/0}$	$\mathbf{29.0\pm3.8}$	$\mathbf{17.77\pm0.67}$	$\mathbf{0.27\pm0.15}$
Chemical Yard	Human	$\mathbf{5/0/0}$	$\mathbf{33.9\pm1.6}$	$\mathbf{30.75\pm0.71}$	0.92 ± 0.37
	OmniVLA	$0/0/5$	37.4 ± 6.8	20.28 ± 4.56	$17.86\ (12.99)$
	GemNav (Ours)	$\mathbf{5/0/0}$	42.8 ± 8.2	33.62 ± 4.43	$\mathbf{0.25\pm0.21}$
Warehouse	Human	$\mathbf{5/0/0}$	$\mathbf{34.2\pm1.2}$	$\mathbf{31.83\pm0.56}$	0.58 ± 0.30
	OmniVLA	$0/0/4$	25.9 ± 15.8	14.49 ± 5.32	$29.18\ (27.31)$
	GemNav (Ours)	$\mathbf{5/0/0}$	37.9 ± 2.0	32.14 ± 0.86	$\mathbf{0.42\pm0.15}$

GemNav achieves a combined ADE of 0.49 m on SCAND, with a correctly predicted goal-reached signal on all stop samples ($50/50$). On TokenWalker, the combined ADE increases to 1.19 m and stop accuracy remains $80/108$, indicating a moderate degradation under distribution shift rather than a collapse. Across both splits, the ego-only modality (image-goal) is the most challenging, which is consistent with the real-robot results in Section 5.2: pure image-goal navigation often reaches the goal vicinity but does not reliably generate the goal-reached signal.

5.2 Real-world deployment performance

Cross-environment comparison. We compare *GemNav*, OmniVLA, and the human teleoperator under the pose-goal modality, which is the only modality evaluated for all three references across all environments (Table 2). *GemNav* succeeds in every pose-goal trial, achieving a $5/0/0$ stop/partial/miss tally in each of the four environments. Across the open carpark, obstacle carpark, long outdoor chemical yard, and indoor warehouse, it stops within 0.25–0.42 m of the target, indicating consistent goal-reaching accuracy across both short and medium-horizon settings. On the two medium-horizon environments where a human expert reference is available, *GemNav* approaches this upper bound in time and path length while achieving low final displacement. In the Chemical Yard and Warehouse, *GemNav* stops within 0.25m and 0.42m of the goal, respectively, compared with 0.92m and 0.58m for the human expert. These numbers should be interpreted as a task-level reference rather than a direct competition, since the human expert stops based on perceived completion rather than minimizing coordinate error.

Modality breakdown (Carpark). The Carpark is the only environment where all three goal modalities are evaluated (Appendix N, Table 16). Image-only navigation is the weakest setting: *GemNav* often reaches the goal vicinity but generates `<goal_reached>` only once ($1/5$), while OmniVLA never stops ($0/5$). Providing a goal coordinate improves *GemNav*'s stopping behavior ($4/5$ for ego+pose, $5/5$ for pose), whereas OmniVLA fails under the pose goal by moving away from the target after the required $\sim 90^\circ$ turn.

Obstacle avoidance. The Obstacle Carpark, Chemical Yard, and Warehouse all require routing around obstacles between the start pose and the goal, and the policy does so while still stopping on the goal coordinate (Table 2, $5/0/0$ in all three); the observed routing is attributable to the model's waypoint outputs rather than to Spot's near-contact safety condition.

6 Ablations

Decode mode. We first study how discrete waypoint tokens should be decoded into continuous robot actions. We compare the deployment default, continuous decoding via the softmax expectation over value-bin centres, against bin-snapped decoding using greedy argmax over bins. Continuous

decoding improves validation ADE by approximately 5 cm across modalities and improves closed-loop performance in the more complex ego+pose setting, increasing Warehouse ego+pose success from $3/5$ to $5/5$ relative to bin-snapped decoding, while leaving pose FDE essentially unchanged. Full offline and on-robot results are provided in Appendix H.

History depth. We next evaluate whether adding visual history improves deployment performance. We trained single-frame ($h = 0$), two-frame ($h = 2$), and four-frame ($h = 4$) history variants, each its own four-phase SCAND _d chain (Section P), and deployed all three on the Carpark with the inference window matched to the training depth (Appendix I, Table 12). Offline and closed-loop results show opposite trends. On held-out OOD validation, combined ADE improves monotonically with history depth (1.45 m for $h = 0$, 1.38 m for $h = 2$, and $\mathbf{1.15}$ m for $h = 4$). In contrast, real-robot deployment the single-frame policy: $h = 0$ is fastest and most reliable, $h = 2$ remains reliable but is approximately 5 s slower, and $h = 4$ drops to $4/5$ success on pose-bearing modalities with substantially larger final displacement. We attribute this discrepancy to a time-lagged trajectory effect: past visual context encourages the policy to continue the previous motion, while offline ADE, computed on logged trajectories rather than closed-loop rollouts, does not penalize this delay. An unmatched-window control confirms that the past-frame context is the source of the degradation (Appendix I). The full on-robot breakdown is given in Appendix I, Table 12.

Goal sampling. Finally, we study the distance-based goal sampler used in the our method. This sampler selects the goal at a fixed metric distance along the trajectory, rather than at a fixed frame offset. To the best of our knowledge, this is the first use of metric-distance goal sampling in the visual-navigation foundation-model setting. The legacy frame-based sampler achieves lower in-distribution combined ADE (0.26 m versus 0.49 m), but this offline advantage does not translate to deployment. Its OOD pose FDE increases to 2.88 m, compared with 0.19 m for distance-based sampling, and the resulting policy fails across all robot modalities. By contrast, the distance-based sampler deploys successfully across all four environments (Appendix F). This result shows that in-distribution ADE alone is not a reliable indicator of deployability; goal-localization FDE and closed-loop performance are more informative for waypoint navigation. Additional eval-only ablations on crop mode, LoRA rank, model size, and the untrained baseline are reported in Appendix E.

7 Limitations

GemNav is trained on a single 8.7-hour corpus from one platform and deployed on one robot. As a result, cross-embodiment transfer remains untested. Auto stop on a pure image goal (like prior SOTA methods) is unreliable, so the policy depends on a goal coordinate. The 2D (x, y) waypoint output cannot express in-place rotation and filters goals behind the robot (Appendix K). We also observe a time-lagged trajectory effect in closed-loop deployment, which can cause overshoot problem.

8 Conclusion

We introduced *GemNav*, a visual robot navigation policy that reformulates waypoint prediction as discrete-token generation with a frozen MLLM. The main finding is that, for short-to-medium-horizon waypoint navigation, strong pretrained MLLM representations can be made actionable without training a new visual encoder or adding a continuous regression head. Instead, metric waypoints, goal-reaching behavior, and unreachable-goal decisions can be expressed through a shared token interface and decoded by the language-model head. Despite being trained only on the 8.7-hour SCAND dataset, *GemNav* transfers to four unseen real-world environments and completes all pose-conditioned trials, including obstacle, outdoor, and indoor settings. The results also show that better offline validation scores do not always translate to better closed-loop behavior: adding short visual histories improves validation ADE but does not improve real-robot deployment. Together, these findings suggest that action representation, rather than further visual scaling alone, is an important design axis for adapting MLLMs to robot navigation. Future work will extend this study along three directions: evaluating cross-embodiment transfer across additional robot platforms, and improving pure image-goal stopping without relying on a goal coordinate.

References

[1] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5):701–739, 2025.

[2] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning (CoRL)*, pages 711–733, 2023.

[3] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[4] J. Zhang, A. Li, Y. Qi, M. Li, J. Liu, S. Wang, H. Liu, G. Zhou, Y. Wu, X. Li, Y. Fan, W. Li, Z. Chen, F. Gao, Q. Wu, Z. Zhang, and H. Wang. Embodied navigation foundation model. *arXiv preprint arXiv:2509.12129*, 2025.

[5] N. Hirose, C. Glossop, D. Shah, and S. Levine. Omnivla: An omni-modal vision-language-action model for robot navigation. *IEEE International Conference on Robotics and Automation*, 2026.

[6] R. Doshi, H. R. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference on Robot Learning*, pages 496–512. PMLR, 2025.

[7] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pages 2679–2713. PMLR, 2025.

[8] J. Zhang, X. Chen, Q. Wang, M. Li, Y. Guo, Y. Hu, J. Zhang, S. Bai, J. Lin, and J. Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *International Conference on Learning Representations*, 2026.

[9] S. Dey, J.-N. Zaech, N. Nikolov, L. Van Gool, and D. P. Paudel. Revla: Reverting visual domain limitation of robotic foundation models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8679–8686, 2025.

[10] Gemma Team. Gemma 4 model. https://ai.google.dev/gemma/docs/core/model_card_4, 2026. Google DeepMind.

[11] 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.

[12] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction in robotics. In *8th Annual Conference on Robot Learning*.

[13] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pages 8469–8488, 2023.

[14] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[15] S. Zeng, D. Qi, X. Chang, F. Xiong, S. Xie, X. Wu, S. Liang, M. Xu, X. Wei, and N. Guo. Janusvln: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. *International Conference on Learning Representations*, 2025.

[16] 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*.

[17] X. Liu, J. Li, Y. Jiang, N. Sujay, Z. Yang, J. Zhang, J. Abanes, J. Zhang, and C. Feng. Citywalker: Learning embodied urban navigation from web-scale videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6875–6885, 2025.

[18] C. Glossop, W. Chen, A. Bhorkar, D. Shah, and S. Levine. Cast: Counterfactual labels improve instruction following in vision-language-action models. *arXiv preprint arXiv:2508.13446*, 2025.

[19] 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.

[20] A. J. Hancock, X. Wu, L. Zha, O. Russakovsky, and A. Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. *International Conference on Learning Representations*, 2026.

[21] Z. Hou, T. Zhang, Y. Xiong, H. Duan, H. Pu, R. Tong, C. Zhao, X. Zhu, Y. Qiao, J. Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7686–7697, 2025.

[22] P. Li, Y. Chen, H. Wu, X. Ma, X. Wu, Y. Huang, L. Wang, T. Kong, and T. Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *Advances in Neural Information Processing Systems*, 38:63635–63673, 2026.

[23] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 1(2):3, 2022.

[24] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

[25] M. Ramezani, K. Khosoussi, G. Catt, P. Moghadam, J. Williams, P. Borges, F. Pauling, and N. Kottege. Wildcat: Online continuous-time 3d lidar-inertial slam. *arXiv preprint arXiv:2205.12595*, 2022.

[26] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[27] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European conference on computer vision*, pages 683–700. Springer, 2020.

Supplementary Material for GemNav: Discrete-Token Visual Robot Navigation using a Multimodal Large Language Model

A TokenWalker Dataset (TW) Description

TokenWalker (TW) dataset covers outdoor paved access roads, sidewalks, and industrial yards, captured under daytime cloudy-to-overcast lighting. Typical scenes are dominated by a paved path receding into the distance, bordered by dense vegetation or by industrial structures (corrugated-steel sheds, stacked IBC tanks, painted line markings, oil drums on pallets). The visual character is highly repetitive within a session: many consecutive frames share the same path-receding composition under slow forward motion, so the corpus stresses pose-grounded navigation rather than diverse scene-level perception. This composition makes TW a natural OOD complement to SCAND: the platform, capture rate, and scene class differ noticeably from SCAND's broader indoor and outdoor coverage, providing a clean image-distribution shift without changing the goal-conditioned navigation task.

B Staged Complexity Results

To complement the main-paper deployment results (Table 2), where OmniVLA already fails to reach the goal in the complex-obstacle setting, we ran an additional set of warehouse experiments that vary routing difficulty in controlled stages: an open run, a single obstacle, and a dual obstacle, with the start, goal, and modality held fixed.

Table 3: Staged-complexity warehouse deployment. Pose+goal-image modality used for both models . SR = stop / partial / miss; the SR denominator gives the trial count n per cell. All metrics are aggregated over all trials; Δ shows mean final displacement with mean nearest approach in parentheses. **Bold** marks the best result among completing models.

Scenario ($\rightarrow$ harder)	Model	SR $\uparrow$	t (s) $\downarrow$	d (m) $\downarrow$	Δ (m) $\downarrow$
Open (no obstacle)	OmniVLA	$2/0/2$	11.0 ± 0.7	9.48 ± 0.51	2.87 ± 0.70 (0.65)
	GemNav (Ours)	$\mathbf{4/0/0}$	22.6 ± 0.3	9.34 ± 0.17	$\mathbf{0.40\pm0.01}$ $(\mathbf{0.38})$
Single obstacle	OmniVLA	$0/0/3$	18.3 ± 2.0	14.20 ± 0.88	6.32 ± 0.09 (2.80)
	GemNav (Ours)	$\mathbf{4/0/1}$	28.1 ± 16.7	18.48 ± 11.08	$\mathbf{2.08\pm3.41}$ $(\mathbf{0.47})$
Dual obstacle	OmniVLA	$0/0/3$	31.7 ± 21.6	13.28 ± 2.25	6.08 ± 0.36 (2.72)
	GemNav (Ours)	$\mathbf{6/0/0}$	24.6 ± 9.2	14.89 ± 3.97	$\mathbf{0.43\pm0.20}$ $(\mathbf{0.43})$

Note: GemNav's single-obstacle row is skewed by its one miss (57 s, 37.4 m, stopped 8.9 m away); excluding it, the other four runs give $t = 20.8$ s, $d = 13.75$ m, $\Delta = 0.38$ m (nearest 0.47 m).

Table 3 summarizes the results. We can see that in the *Open (no obstacle)* scenario OmniVLA moves as fluently as *GemNav* and its path sweeps through the goal region, but it stops on the goal in only half the runs and otherwise walks straight through and past it. *GemNav* stops on the goal every time. The gap here is purely terminal, reaching the goal versus committing to it. *Single obstacle.* A required lateral detour erases this residual competence entirely (Fig. 2). *GemNav* curves around the obstacle and settles on the goal, whereas OmniVLA skirts the obstacle, holds its original heading, and comes to rest near the far wall, never re-acquiring the goal direction. It does not approach and narrowly miss; it leaves the goal behind. *Dual obstacle.* The collapse repeats with a second obstacle, confirming the behaviour is systematic rather than tied to one layout: OmniVLA again halts several metres past the goal while *GemNav* reaches it. Across all three stages the times and path lengths are comparable, so the divergence is not locomotion but metric-goal grounding: as the scene demands more deviation, OmniVLA loses first the ability to stop on the goal and then the ability to head toward it at all, while *GemNav* keeps closing the loop on the coordinate.

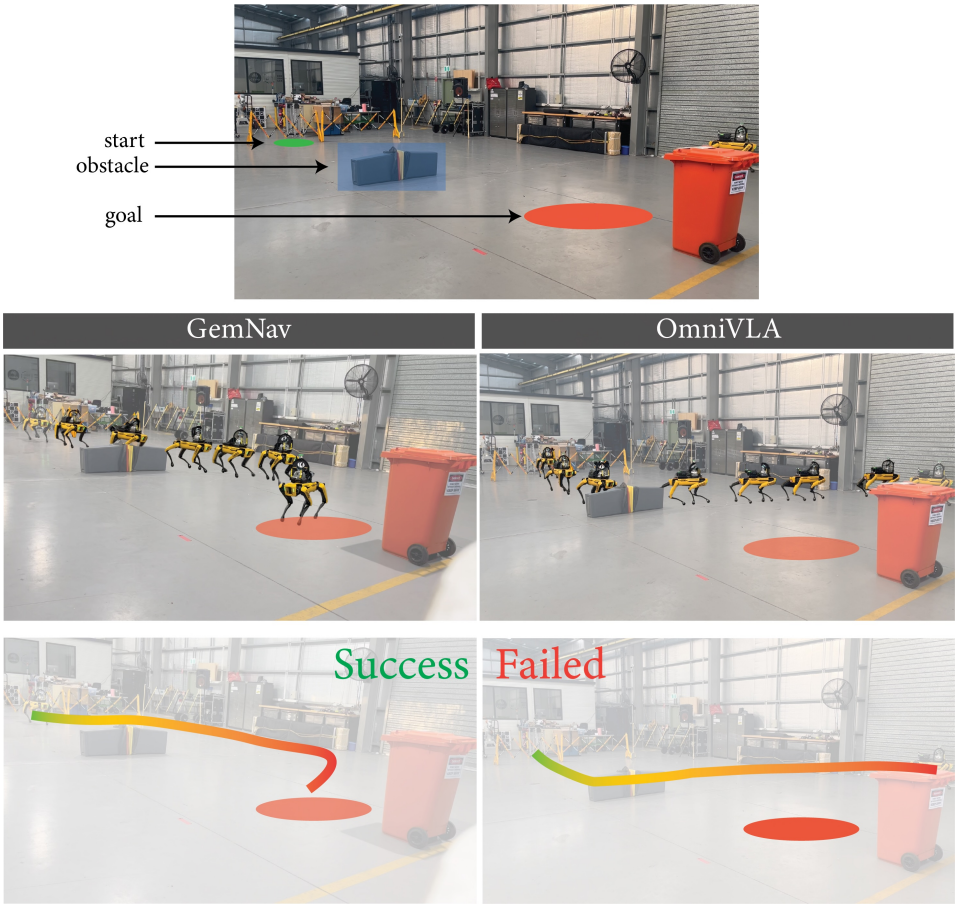

Figure 2: **Qualitative single-obstacle warehouse run.** Top: scene with start, obstacle, and goal. Middle: time-lapse of GemNav (left) reaching the goal and OmniVLA (right) walking past it to the far wall. Bottom: overlaid trajectories, GemNav routing around the obstacle and stopping on the goal (success), OmniVLA holding its heading past the goal (failure)

Because OmniVLA already fails in the presence of obstacles, the medium-horizon regime is evaluated for *GemNav* (Figure. 3). This run raises every axis at once: the course is roughly 32,m, the scene is previously unseen and densely cluttered with bins, cones, workbenches, machinery, and a seated bystander, and the goal is not visible from the start. It therefore probes whether the policy can hold and pursue a purely metric goal it never sees, threading sustained clutter while still terminating on the coordinate. *GemNav* does exactly this. It routes around the central structure and the scattered obstacles in one continuous trajectory and stops on the target. Three things follow. The goal coordinate acts as a persistent anchor rather than a momentary heading, since the policy commits to a target it cannot observe for most of the run. Local avoidance and global goal-seeking compose, as many successive detours are chained without losing the objective. And terminal accuracy holds at horizons several times longer than the staged trials, so stopping precision is not a short-range artifact.

C Full Prompt Template

For reproducibility we reproduce the prompt template used at training and inference time. The template is rendered as a single Gemma-4 user-turn chat message via the model's stock processor; the assistant turn (trajectory or stop sequence) is appended as token IDs rather than text strings to preserve the `<unusedN>`-derived special tokens (Section 3.2).

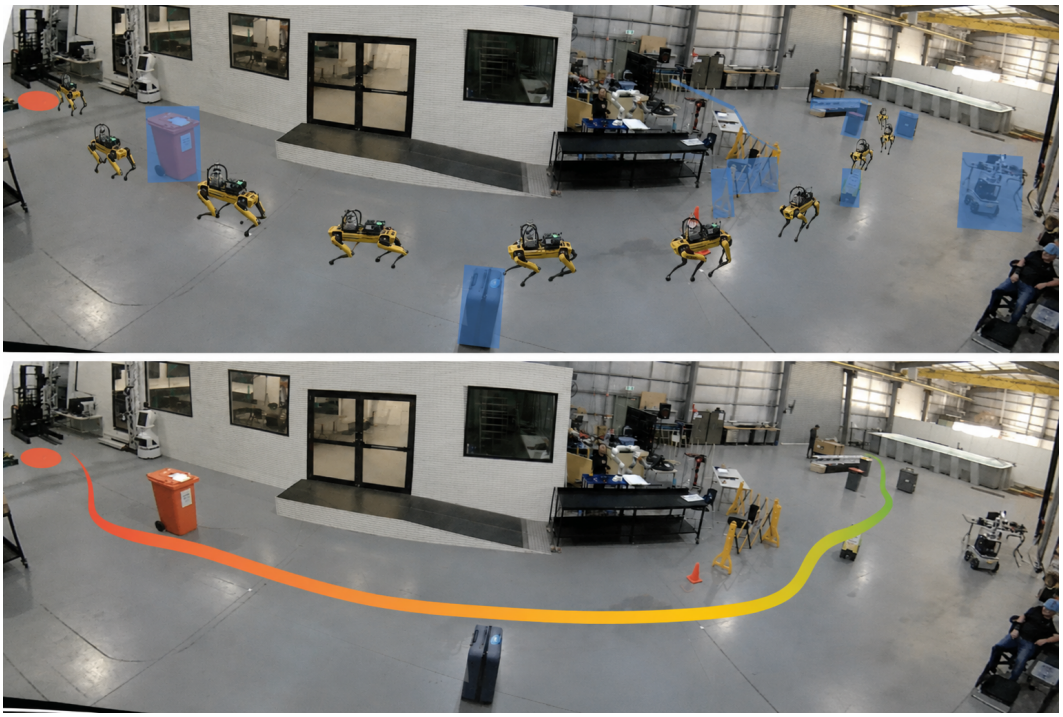

Figure 3: **Medium-horizon indoor warehouse run (32 m, unseen scene).** Top: time-lapse of GemNav on Spot traversing the course from start (right) to goal (left). Despite dense clutter (bins, cones, workbenches, machinery, and a seated bystander, highlighted in blue) and a goal not visible from the start, the policy reaches the target successfully. Bottom: executed path inferred from visual inspection.

Assistant turn (appended as token IDs). Trajectory and stop targets share the same outer <bot_traj> / <eot_traj> brackets.

```
[trajectory target, length 18 tokens]
<bot_traj> <vx_1> <vy_1> ... <vx_8> <vy_8> <eot_traj>

[stop targets, length 3 tokens each]
<bot_traj> <goal_reached>     <eot_traj>   (goal-arrival condition met)
<bot_traj> <goal_unreachable> <eot_traj>   (cross-trajectory negative)
```

System intro text (two variants). The no-history wording is held byte-for-byte stable so that $h = 0$ checkpoints remain comparable to the pre-history template.

```
[h > 0]
You are a VLA navigation agent. Given a current camera view, optional
recent past views, and either a goal image or a goal coordinate, output
an 8-waypoint trajectory in the robot-local North/East frame (meters).
Emit <goal_reached> instead of waypoints if the goal is reached.

[h = 0]
You are a VLA navigation agent. Given a current camera view and either
a goal image or a goal coordinate, output an 8-waypoint trajectory in
the robot-local North/East frame (meters). Emit <goal_reached> instead
of waypoints if the goal is reached.
```

User-turn structure. The user message is a single interleaved sequence of text segments and image placeholders, with three optional blocks gated on history depth and goal modality:

```
<intro text from above>

[present iff h > 0]
Recent views (oldest -> newest, h frames):
<image at t - h*delta>
<image at t - (h-1)*delta>
...
<image at t - delta>

Current view:
<image at t>

[present iff goal modality includes an image]
Goal view:
<image at t_g>

[present iff goal modality includes a coordinate]
Goal coord: <goal_coord><value_bin_x><value_bin_y>

Trajectory:
```

D Zero-shot Multi-modal Fusion Across Training Chains

Table 4 Reports zero-shot multimodal fusion findings showing the joint pose+ego (modality 5) ADE across the SCAND _d chains at three crop modes. For each chain we report the phase-B (step 2) checkpoint's ADE on the held-out joint pose+ego split (zero-shot transfer) alongside the phase-C1 (step 3) checkpoint trained explicitly on the joint modality (specialized). OOD numbers are reported on the held-out TokenWalker validation set (Appendix A).

Table 4: Joint pose+ego (modality 5) ADE in meters at 500-sample evaluation limit. Step-2 ckpt was trained on disjoint $(1, 0, 1)$ pose-only + ego-only samples and has never been exposed to the joint modality; step-3 ckpt was warm-started from step-2 and specialized on the joint modality $(0, 1, 0)$. OOD evaluations on step-3 rows use the rebuilt full-TW manifest (Section 5.1). † marks step-2 zero-shot values still measured against the earlier (lagged-position) TokenWalker manifests, pending re-eval. **Bold** marks the best value per column.

	Step-2 ckpt (zero-shot)		Step-3 ckpt (specialized)	
Training chain	in-dist	OOD	in-dist	OOD
SCAND (_d, top crop)	$\mathbf{0.43}^{\dagger}$	$\mathbf{0.98}^{\dagger}$	0.35	0.70
SCAND (_d, stretch crop)	$0.45^{\dagger}$	$1.46^{\dagger}$	**0.34**	0.68
SCAND (_d, center crop)	$0.46^{\dagger}$	$1.43^{\dagger}$	0.35	**0.67**

Two observations stand out. First, all three SCAND chains' step-3 specialized in-distribution ADEs collapse to 0.34–0.35 m, confirming that the joint-modality benefit is robust to the deployment-side crop-mode choice. Second, on the rebuilt full-TW OOD eval the three crops sit close together (top 0.70, stretch 0.68, center 0.67 m), well inside the deployment-relevant regime.

E Additional Ablations

E.1 Auxiliary regression loss

We isolate the auxiliary term's contribution by training without it ($\lambda_{\text{aux}} = 0$, equivalent to pure causal-LM CE) and with it at our default weight ($\lambda_{\text{aux}} = 0.1$, MSE), keeping all other settings identical (ego-only modality, LoRA rank $r = 32$, 40,000 steps).

Table 5: Auxiliary regression-loss ablation on ego-only validation. Same modality, same 40,000-step budget, same LoRA rank.

Loss configuration	val ADE [m]
CE only ($\lambda_{\text{aux}} = 0$)	1.05
CE + soft-decoded regression ($\lambda_{\text{aux}} = 0.1$)	**0.81**

The auxiliary term reduces ADE by 23% (1.05 m to 0.81 m) without any other change to the recipe. We further tracked stop-class behavior at intermediate checkpoints to verify that the ordinal pressure does not destabilize the discrete stop signal: across 12,000, 20,000, and 40,000 steps the spurious-stop count fell monotonically ($34 \rightarrow 17 \rightarrow 13$ on 431 trajectory samples), and the missed-stop count remained low and stable ($3 \rightarrow 7 \rightarrow 4$). We attribute the gain to the ordinal structure that pure CE lacks: predicting an adjacent value-bin token incurs a small penalty under the soft-decoded prediction in (3), while CE penalizes adjacent and far errors equally regardless of the metric distance between bins.

E.2 LoRA capacity

We test whether the language-tower adapter is itself a primary capacity bottleneck by doubling the LoRA rank from $r = 32$ to $r = 64$ at fixed scaling $\alpha = 16$, leaving every other setting unchanged.

Table 6: LoRA rank ablation on ego-only validation. Doubling r at fixed α regresses every metric we track.

LoRA rank	val ADE [m]	val FDE [m]	stop accuracy	missed stops
$r = 32,\ \alpha = 16$	**0.81**	**1.47**	**65/69**	**4**
$r = 64,\ \alpha = 16$	0.92	1.67	60/69	9

The higher-rank configuration is uniformly worse: ADE rises by 14%, FDE rises by 14%, and missed stops more than double. We attribute this to LoRA's effective per-update scaling factor α/r: doubling r at constant α halves the ratio (from 0.5 to 0.25), producing more parameters but a smaller per-matrix contribution per training step. The numbers are therefore consistent with under-training rather than gained capacity. A genuine capacity test would require either scaling α together with r or substantially extending the step budget; given that ego-only ADE under $r = 32$ already lands well inside our deployment-relevant regime (~ 1 m success threshold for short-to-medium-horizon waypoint navigation), neither is justified here, and we retain $r = 32$ as the default for all subsequent experiments.

E.3 Crop-mode ablation

When source images are not natively 224×224 (SCAND has 1280×720 Spot frames and 1280×1024 Jackal frames), we have three plausible preprocessing choices: *stretch* (resize the full image to 224×224, accepting horizontal/vertical squashing; preserves all FOV at the cost of geometric distortion), *top* (keep the bottom 50% of rows then center-crop to square; preserves geometry at the cost of discarding the upper FOV), and *center* (center-crop to a square from the smaller dimension; preserves geometry while keeping center FOV in both axes). We ran the same four-stage chain at each of the three crop modes on SCAND `_d`, against the same train/val split.

On *single-modality* metrics in-distribution, stretch and center cluster together and both beat top: in-distribution ego ADE 0.87 m (stretch) / 0.85 m (center) / 1.13 m (top) at step 1. The extra navigation context preserved by full or near-full FOV outweighs any geometric concern when only one signal is present in-distribution. On OOD ego at step 1 the picture differs: top 2.16 m, center 2.42 m, stretch 2.44 m; top wins by a small margin, with center and stretch close behind.

On the joint pose+ego *step-3 specialized* fusion (Table 4), the in-distribution ADE collapses to 0.34–0.35 m across all three crops; on OOD (rebuilt full-TW manifest) the three sit close together (top 0.70, stretch 0.68, center 0.67 m), well inside the deployment-relevant regime. Step-2 *zero-shot* OOD numbers in the table are still those measured against earlier (lagged-position) TokenWalker manifests (pending re-eval), and so the pre-specialization crop-rank pattern at step-2 is not safe to interpret in this revision. The step-4 chain-endpoint OOD numbers in the following paragraph are also still on the earlier manifest for the top and center crops (only stretch was re-evaluated against the full-TW manifest, with combined ADE 1.19 m); pending re-eval of top and center step-4 against the rebuilt manifest, the cross-crop step-4 OOD comparison should be read with a manifest-mismatch caveat.

At the chain endpoint (step 4, mix `1,1,1`), the same three crops produce three different ranking patterns across modalities. In-distribution: stretch wins on combined ADE (0.49 m, vs center 0.53, top 0.61) and on ego-only (0.76 m, vs center 0.89, top 1.13); pose and pose+ego in-distribution ADE collapse to within noise across all three (0.32–0.38 m). On OOD: top wins on combined ADE (1.27 m, vs center 1.37, stretch 1.45), on ego (2.19 m, vs center 2.49, stretch 2.64), and on pose+ego (0.78 m, vs center 0.83, stretch 0.86); center wins on pose-only OOD (0.81 m, vs top 0.85, stretch 0.86). Mod-6 calibration recovery is perfect for all three. We report this primarily as a numerical observation rather than a load-bearing finding; the per-modality rank order varies by modality and split, and the deployable-side trade-offs depend on which modality dominates the rollout pattern.

F Frame-based Goal-sampling: Ablation and On-robot Deployment

An earlier chain was trained under the legacy frame-aligned goal-sampling mode (Section 4.1), in which the goal index always coincides with one of the eight GT waypoint frames. Offline validation numbers under that mode were competitive, but the resulting chain failed on real-robot deployment: the policy emitted `<goal_reached>` on every modality, effectively refusing to plan irrespective of the actual goal. We attribute this to the fixed training horizon implied by frame alignment (~ 32 frames, or ~ 1.6 m at the corpus's frame rate and typical robot speed), which produces a trajectory-length prior that does not extend to deployment-time goal distances. The distance-based sampler now used by all reported runs (Section 4.1) was introduced to fix this.

On-robot characterization. The structured on-robot comparison against the distance-mode chain has since been collected. The frame-mode chain failed on every modality, emitting `<goal_reached>` immediately or drifting ~ 1 m before stopping, irrespective of the goal, confirming the preliminary observation above; the distance-mode chain deploys across all four environments (Section 5.2).

Eval contrast. Table 7 reports the offline eval contrast that explains the gap: frame-mode wins in-distribution ADE but lacks FDE-identity (OOD pose FDE 2.88 m vs distance-mode's 0.19 m) and collapses on OOD ADE.

G Deployment-path Optimizations and Latency Bench

We measure the time required for one policy inference across several hardware platforms. The A100 and A6000 represent workstation-class GPUs, while Thor and Orin represent embedded platforms intended for onboard deployment. We also examine how inference time changes with model size, goal input, history length, and weight quantization. Table 8 reports the average time over 500 runs

Table 7: Goal-sampling ablation: distance-mode (the deployed sampler) vs the legacy frame-mode.

Metric	distance-mode (deployed)	frame-mode
in-dist combined ADE	0.49	**0.26**
in-dist pose ADE / FDE	0.37 / **0.18**	**0.26** / 0.24
OOD combined ADE	**1.45**	1.73
OOD pose ADE / FDE	**0.86 / 0.19**	1.68 / 2.88
real-robot (all modalities)	deploys (5/5, all four envs)	fails

after an initial warm-up. All experiments use the same trained policies and discrete action-token interface as the real-robot experiments.

We use two implementation changes to reduce inference time. First, we compile the autoregressive forward pass and use a key-value cache with fixed tensor shapes. This allows the model to reuse the same execution graph at each decoding step. Second, when the goal image does not change, we compute its visual features once and reuse them in later control steps. This avoids running the vision encoder repeatedly for the same image. These changes only affect how the model is executed. They do not change the model weights, prompts, predicted tokens, or robot trajectory.

Table 8: Average inference time in milliseconds, measured over 500 runs after warm-up. "–" indicates that a configuration was not tested on that hardware.

Configuration	A100	A6000	Thor	Orin
Deployed E2B model				
Baseline inference	1768	2058	1479	6557
Optimized inference	**421**	**500**	**1129**	–
Model size				
E4B baseline inference	2216	2968	1760	8887
E4B optimized inference	**549**	**736**	**1383**	–
Goal input (E2B)				
Pose only	1710	2112	1407	6139
Pose and ego image	1779	2212	1475	6514
History length (E2B)				
$h = 2$	1828	–	1763	7668
$h = 4$	1868	–	1763	8800
Weight quantization (E2B)				
4-bit	2480	3027	1977	9231
8-bit	4783	5634	3939	16580

The implementation changes provide the largest reduction in inference time. For E2B, the average falls from 1768 to 421 ms on A100 and from 2058 to 500 ms on A6000. On Thor, it falls from 1479 to 1129 ms. We observe the same pattern for E4B, although E4B remains slower than E2B on each platform. The larger model takes longer to run on all four platforms. Without the implementation changes, moving from E2B to E4B adds 448 ms on A100, 910 ms on A6000, 281 ms on Thor, and 2330 ms on Orin. The difference remains after applying the implementation changes.

Interestingly, the form of the goal input has much less influence on inference time. Using both the pose and ego image is less than 7% slower than using the pose alone on every platform. This suggests that most of the time is spent running the MLLM, rather than processing the additional goal image. History length also has little effect on A100 and Thor. Increasing the history from $h = 2$ to $h = 4$ adds 40 ms on A100 and produces the same measured time on Thor. However, Section I shows that the longer history reduces closed-loop performance. The drop in performance is therefore unlikely to be caused by slower inference. On Orin, the same change increases inference time from 7668 to 8800 ms.

Howwver, weight quantization does not make inference faster in these experiments. Both the 4-bit and 8-bit models are slower than the full-precision model on every platform tested. Quantization may still be useful when GPU memory is limited, but it does not reduce inference time here. We therefore use the full-precision E2B model in the deployment experiments.

H Decode Mode

At inference the value-bin logits can be turned into a waypoint two ways: *bin-snapped* (greedy argmax over bins) or *continuous* (the softmax-weighted expectation over bin centers). Continuous decoding recovers sub-bin precision and is the deployed choice; Table 9 reports the eval-side gain and Table 10 the on-robot gain.

Table 9: Decode-mode eval ADE (m) on SCAND in-distribution for the deployed (aux-on) checkpoint. Continuous decoding gains ~ 5 cm of ADE; pose FDE is unchanged.

Decode mode	pose ADE	mod-5 ADE	pose FDE
bin-snapped (greedy)	0.37	0.33	0.18–0.19
continuous (expectation)	**0.32**	**0.28**	0.18–0.19

Table 10: Decode mode on robot. SR = autonomous goal-reach over $n = 5$; t / d / Δ over goal-emitting trials. On the short Carpark the two decoders are within noise; on the Warehouse continuous is ~ 5 s faster and ~ 1.4 m shorter on pose, and lifts ego+pose from $3/5$ to $5/5$.

Environment · modality	Decode	SR	t (s)	d (m)	Δ (m)
Carpark · pose	continuous	$5/5$	**17.5**	**12.0**	0.39
	bin-snapped	$5/5$	17.6	12.3	**0.33**
Warehouse · pose	continuous	$5/5$	**37.9**	**32.1**	0.42
	bin-snapped	$5/5$	43.1	33.5	**0.31**
Warehouse · ego+pose	continuous	**5/5**	45.8	33.4	0.35
	bin-snapped	$3/5$	43.8	34.1	0.38

I History Depth

Section 6 reports a discrepancy between offline evaluation and real-robot deployment: adding visual history improves held-out trajectory metrics but does not improve closed-loop behavior. This appendix provides the corresponding offline, matched-window, and unmatched-window results.

Table 11 shows the chain-endpoint evaluation results. On the OOD TokenWalker split, combined ADE decreases from 1.45 m for $h = 0$ to 1.15 m for $h = 4$, a 21% improvement. In-distribution performance remains largely unchanged across all three depths.

Table 11: History-depth evaluation performance at the chain endpoint. OOD ADE improves monotonically with history depth.

Checkpoint	In-distribution ADE	OOD ADE
$h = 0$	0.49	1.45
$h = 2$	0.50	1.38
$h = 4$	**0.49**	**1.15**

In contrast, the deployment results show the opposite ordering (Table 12). The single-frame policy is the fastest and most reliable configuration. The two-frame policy preserves success rate but increases traversal time, while the four-frame policy drops to $4/5$ success on pose-bearing modalities and produces substantially larger final displacement. Thus, the offline improvement from additional history does not translate to improved closed-loop performance.

Table 12: History depth on the Carpark with the inference window matched to the training history. SR denotes successful autonomous goal reaching over $n = 5$ trials.

	pose				ego+pose				ego
h	SR	t(s)	d(m)	Δ(m)	SR	t(s)	d(m)	Δ(m)	SR
0	$5/5$	$\mathbf{17.5}$	12.0	0.39	$4/5$	$\mathbf{15.3}$	$\mathbf{9.85}$	0.33	$\mathbf{1/5}$
2	$5/5$	22.2	11.9	$\mathbf{0.34}$	$\mathbf{5/5}$	19.9	11.7	$\mathbf{0.30}$	$0/5$
4	$4/5$	22.4	$\mathbf{11.7}$	0.86	$4/5$	24.1	14.1	0.73	$\mathbf{1/5}$

We further evaluate unmatched inference windows to test whether the effect is associated with the history-trained checkpoint or with the presence of past-frame context at deployment. Table 13 shows that both the $h = 2$ and $h = 4$ checkpoints recover $5/5$ success when evaluated with a single-frame input. Conversely, running the $h = 0$ checkpoint with a two-frame inference window increases pose traversal time despite leaving the learned weights unchanged.

Table 13: Unmatched inference-window evaluation on the Carpark. Each cell reports SR, t(s), d(m), and Δ(m).

Configuration	Carpark · pose	Carpark · ego+pose
$h = 2$ checkpoint, single-frame inference	$5/5$, 15.7, 10.34, 0.33	$5/5$, 16.4, 10.27, 0.27
$h = 4$ checkpoint, single-frame inference	$5/5$, 15.3, 10.19, 0.33	$5/5$, 16.3, 10.48, 0.18
$h = 0$ checkpoint, two-frame inference	$5/5$, 21.3, 10.52, 0.42	$5/5$, 14.2, 9.58, 0.32

Together, these results support the interpretation proposed in Section 6. Removing the history window largely restores the behavior of the single-frame policy, even for checkpoints trained with additional history. This is consistent with the time-lagged trajectory effect discussed in the main text, in which past visual context encourages the policy to continue previous motion despite changing closed-loop conditions.

J Dataset-mix Ablation (SCAND vs. SCAND+TokenWalker)

We tested whether adding the TokenWalker corpus (Appendix A) to the SCAND training mix improves deployment. It did not: it regressed the image-bearing modalities. This ablation stays in the supplementary because both arms are $h = 2$ checkpoints (history-confounded relative to the single-frame deployment model) and the robot test was a single 16 m forward-goal-with-obstacles scenario rather than the four-environment battery of Section 5.2; we record it because the result is counter-intuitive. The arms are a SCAND-only $h = 2$ chain endpoint and a SCAND+TokenWalker mixed $h = 2$ warm-start endpoint (1:3 SCAND:TW per-batch weighting).

Table 14: Real-world autonomous stops over $n = 10$ on the 16 m forward-goal obstacle scenario. Adding TokenWalker hurt both modalities, image-only most ($4/10 \rightarrow 1/10$).

Goal Modality	SCAND-only $h = 2$	SCAND+TW $h = 2$
pose-only	$\mathbf{9/10}$	$7/10$
image-only	$\mathbf{4/10}$	$1/10$

Mixing in TokenWalker neither helped in-distribution (a mild regression, combined $+6\%$, ego $+13\%$) nor on the robot (image-only $4/10 \rightarrow 1/10$, pose-only $9/10 \rightarrow 7/10$). The likely mechanism is that TokenWalker trajectories are long and nearly straight, so the policy acquires a forward-march prior that overrides the foreground goal-image target: image-only "drives past and ignores" the goal, while pose-only stays more robust because the coordinate goal anchors it. The caveats are that both arms are $h = 2$ (history-confounded with the single-frame deployment model), the robot test was a single scenario, and $n = 10$;

Table 15: Eval for the dataset-mix arms. The mix's TW val ADE (0.35) is not a generalization signal: with TW in its training set the TW val split is scene-overlapping rather than zero-shot, unlike the SCAND-only arm's 1.38. The honest comparison is the SCAND in-distribution column (tied or slightly worse for the mix) plus the robot (worse).

Metric	SCAND-only $h = 2$	SCAND+TW $h = 2$
SCAND in-dist combined ADE	**0.50**	0.53
SCAND in-dist ego ADE	**0.79**	0.89
TW val combined ADE	1.38 (zero-shot OOD)	0.35 (TW in train)

K Data Pipeline: History, Filters, Manifests, and Splits

This section expands the data pipeline notes summarized in Section 4.1.

Value-bin choices. $K = 64$ keeps the half-bin error inside the ~ 1 m deployment success criterion and the 0.5 m goal-arrival radius; $B = 15$ m gives a $\sim 7\%$ margin above the largest training goal arc-length ($d_{\max} = 14$ m, Section 4.1). The remaining 30 `<unusedN>` slots are kept free for future modalities.

Image history. When history is enabled ($h > 0$), the manifest records the h past-frame indices $t - k \cdot \delta$ for $k = 1, \ldots, h$, with stride $\delta = 1$ throughout. Trajectories with insufficient history at the chosen t clamp underflowing indices to 0 (repeat-first), so every sample carries exactly $h + 1$ images. Manifests built with $h > 0$ carry an `_h{N}` suffix in the filename, allowing the $h = 0$ baseline and history-enabled variants to coexist on disk for paired evaluation. Past frames in $\mathbf{I}_c$ act as visual context only and are not re-frame-transformed; the predicted waypoints remain in the body frame at the current step t.

Filters, augmentation, and negatives. Two data-quality filters drop samples the policy cannot learn: a *behind-the-robot* filter rejects body-frame goals with $x_g < -0.3$ m (these would require a U-turn first), and a *rotation-in-place* filter rejects `<goal_reached>` samples whose current/goal yaw differs by more than 30° (pure rotation is not expressible in our (x, y)-only output). We apply left-right flip augmentation at the manifest level, mirroring images and reflecting waypoints via $y \mapsto -y$, doubling the corpus. For approximately 10% of image-bearing samples, we replace the goal image with a frame from a different trajectory and emit `<goal_unreachable>`, training a refuse-to-plan response for semantically disconnected goals.

Manifest. A single offline pass over each corpus produces a shared training configuration that records the modality weights, train/val split, flip flag, and target type for every sample. This configuration is used as the source of truth during training and is shared across all GPUs

Splits. Our headline configuration trains on SCAND only: SCAND trajectories are split by trajectory ID (not by sample) with a 10% validation fraction, ensuring the policy is evaluated on unseen routes rather than unseen frames of a known route. The TokenWalker corpus (Appendix A) is held out entirely: no TW trajectories enter training, and OOD numbers reported in this work are measured on the full TW corpus unless otherwise stated. The SCAND+TW mix is retained as the dataset ablation of Appendix J; that configuration regressed image-only behavior on real-robot deployment, which motivates the SCAND-only headline.

L Experimental Setup: Provenance and Conventions

Localization. Body-frame pose is recovered online using a continuous-time 3D LiDAR-inertial SLAM system [25]. This estimator maintains consistency with the pose-estimation pipeline used to record the corpus, so that corpus poses and deployment-time poses are generated under comparable

assumptions. For the navigation task considered here, alternative pose sources, such as visual-inertial odometry, the robot's onboard state estimator, or GPS with appropriate coordinate-frame conversion, could also be used.

Modality identifier convention. For cross-reference with the OmniVLA codebase, the modalities of Section 4.4 carry integer identifiers in our manifests: *ego* is modality 6, *pose* is modality 4, and *ego+pose* is modality 5, aligned with Hirose et al. [5].

Reporting conventions. We do not control illumination during deployment. Runs are conducted under the daylight conditions present at the time of evaluation, and the observed conditions are reported for each set. This reflects realistic outdoor deployment but also introduces occasional direct-sun saturation, which we note in the corresponding per-set discussion. Operator interventions are described separately and are not included in the autonomous stop / partial / miss tallies.

M Validation Setup Detail

This appendix expands the validation overview of Section 5.1.

In-distribution (SCAND data). We evaluate in-distribution performance on the SCAND [19] validation split, a 10% trajectory-disjoint hold-out containing approximately $29,000$ samples after filtering and left–right flip augmentation. Results are computed on a fixed 500-sample subset with balanced coverage across the ego, pose, and ego+pose modalities. We report ADE and FDE for decoded trajectories, decode rate for valid trajectory or stop-token outputs, and stop accuracy on samples whose target output is `<goal_reached>`.

Out-of-distribution (TokenWalker data). We evaluate out-of-distribution (OOD) performance on TokenWalker (Appendix A), which is held out entirely from training. The validation pool contains approximately 9.9k samples after the same filtering and augmentation as SCAND. As in the in-distribution setting, results are computed on a fixed 500 sample modality-balanced subset using the same metrics.

N Per-modality Deployment Breakdown: Carpark Environment

Table 16 gives the full Carpark environment modality breakdown summarized in Section 5.2; its pose rows coincide with the Carpark row of Table 2.

Table 16: Carpark base-test modality breakdown ($n = 5$ per cell). OmniVLA has no ego+pose fusion mode (Section 4.4), so that cell is structurally absent. Cell and SR conventions as in Table 2; **bold** marks the better SR per modality block.

Modality	Model	SR	t (s)	d (m)	Δ (m)
ego (image goal)	*GemNav*	$\mathbf{1/4/0}$	14.2 ± 1.5	9.45 ± 0.59	–
	OmniVLA	$0/5/0$	12.2 ± 0.6	10.19 ± 0.57	–
ego+pose	*GemNav*	$4/0/1$	15.3 ± 1.5	9.85 ± 0.20	0.33 ± 0.16
pose (coordinate goal)	*GemNav*	$\mathbf{5/0/0}$	17.5 ± 1.1	11.98 ± 0.36	0.39 ± 0.11
	OmniVLA	$0/0/5$	8.7 ± 0.7	7.28 ± 0.17	$10.44\ (9.54)$

O Environment Descriptions

Full per-environment descriptions for the four deployment courses summarized in Section 4.5.

Goal acquisition. For image-bearing modalities, the goal image is captured at session start by walking the robot manually to the target pose, taking a photo through the on-board camera, and returning to the start line; the same goal image is then used for every trial in that session. Goal coordinates are expressed in the robot's local frame at trial start.

Carpark (base capability, $\sim$ 10–12 m). A flat, open parking area, clear of natural obstacles, with a single orange bin near the center as the target landmark. The goal pose sits directly in front of the bin and requires a hard $\sim 90^\circ$ counter-clockwise turn; the goal image is captured from a mild clockwise bearing, so the pose-bearing modalities are the harder ask despite the short distance.

Obstacle Carpark (short-range obstacle avoidance, $\sim$ 18 m). The same physical area as the Carpark, with obstacles introduced between the start pose and the goal. Here the orange bin becomes one of the obstacles and the goal pose is placed behind it, so the robot must route around the bin rather than stop at it. Only the pose modality is exercised in this environment.

Chemical Yard (long-horizon outdoor, $\sim$ 30–34 m). An outdoor chemical-storage yard, the same industrial setting visible in the TokenWalker imagery (Appendix A): long and visually complex, with narrow passages, dead-ends, and a goal not visible from the start. One branch toward the goal leads into a no-through trap that can lose a policy lacking global context (observed for the OmniVLA baseline). Pose and ego+pose are exercised; image-only is omitted as the goal viewpoint shares no content with the start.

Warehouse (long-horizon indoor, $\sim$ 32 m). A complex indoor warehouse-style obstacle course with multiple obstacles to avoid and a goal that is far from, and not visible at, the start pose. Pose and ego+pose are exercised; image-only is omitted as above.

P Implementation Details

Training is performed in bf16 with gradient checkpointing. The optimizer is AdamW with learning rate 1×10^{-4}, 500-step linear warmup, batch size 1 per device, and a fixed 40,000-step budget per phase, each phase warm-starting from the previous phase's LoRA adapter. The four phases are *A* (fresh LoRA on the $(1, 0, 1)$ pose-and-ego manifest), *B* (warm-start from a single-modality ego-only seed on the same manifest), *C1* (warm-start from B on the modality-5 $(0, 1, 0)$ manifest, specializing on joint pose+ego), and *C2* (warm-start from C1 on the full $(1, 1, 1)$ manifest, broadening to all three modalities and yielding the deployable adapter). Further chain-construction notes (phase-A role, history-depth chain independence) are in Appendix Q.

Q Warm-start Chain Construction Details

This appendix expands the chain protocol summarized in Appendix P. Phase A is retained as a fresh-vs-warm comparator rather than as a deployable: phase B (warm-start on the same manifest) outperforms phase A on both modalities at 40,000 steps, and we observe the same warm-start advantage when adding any new modality to an existing chain. The seed used at phase B is the same single-modality ego-only reference used in the auxiliary-loss and LoRA-rank ablations of Appendix E.1 and Appendix E.2.

The four-phase composition (fresh-mix $\rightarrow$ warm-start $\rightarrow$ specialize $\rightarrow$ broaden) yields the final deployable adapter at phase C2; the same protocol is applied at each history depth $h \in \{0, 2, 4\}$ for the deployment-side variants reported in Section 4. Each history-depth variant runs its own independent four-phase chain, cold-started from the base Gemma-4-E2B-it model at phase A; the within-chain warm-starting (A $\rightarrow$ B $\rightarrow$ C1 $\rightarrow$ C2) is the only warm-start mechanism applied, and the $h > 0$ chains are not warm-started from the corresponding $h = 0$ checkpoint.