VAMOS: A Hierarchical Vision-Language-Action Model for Capability-Modulated and Steerable Navigation

Mateo Guaman Castro[1], Sidharth Rajagopal[1], Daniel Gorbatov[1],
Matt Schmittle[1], Rohan Baijal[1], Octi Zhang[1], Rosario Scalise[1], Sidharth Talia[1],
Emma Romig[1], Celso de Melo[2], Byron Boots[1], Abhishek Gupta[1]
[1] University of Washington, [2] DEVCOM ARL

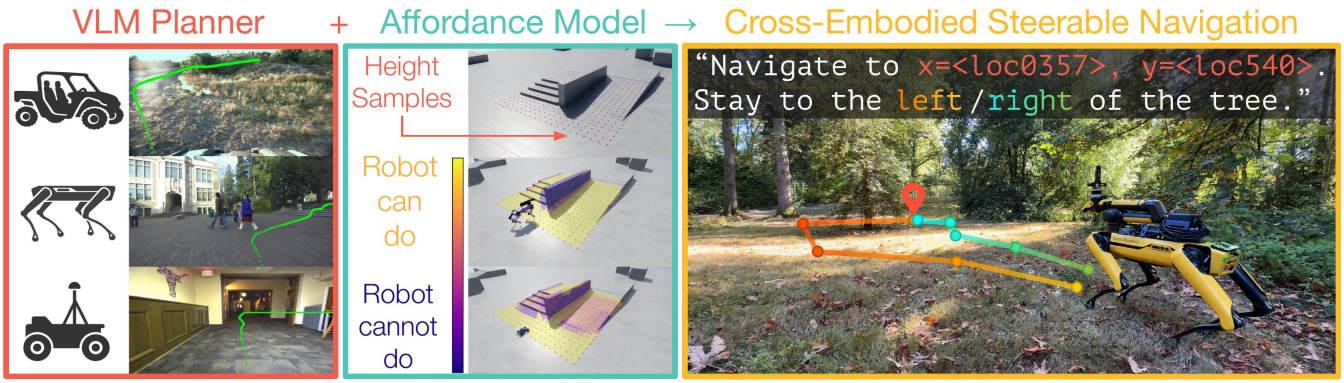


Fig. 1: We present **VAMOS**, a general-purpose hierarchical VLA for navigation. Our key idea is to decouple semantic planning from embodiment grounding. We achieve this by training a high-level VLM planner with diverse, heterogeneous real-world data that proposes trajectory candidates as 2D paths, which are then re-ranked by an embodiment-specific affordance model trained cheaply and safely in simulation. This yields robust, cross-embodied and steerable open-world navigation controllers.

***Abstract*— A fundamental challenge in robot navigation lies in learning policies that generalize across diverse environments while conforming to the unique physical constraints and capabilities of a specific embodiment (e.g., quadrupeds can walk up stairs, but rovers cannot). We propose VAMOS, a hierarchical VLA that decouples semantic planning from embodiment grounding: a generalist planner learns from diverse, open-world data, while a specialist affordance model learns the robot's physical constraints and capabilities in safe, low-cost simulation. We enabled this separation by carefully designing an interface that lets a high-level planner propose candidate paths directly in image space that the affordance model then evaluates and re-ranks. Our real-world experiments show that VAMOS achieves higher success rates in both indoor and complex outdoor navigation than state-of-the-art model-based and end-to-end learning methods. We also show that our hierarchical design enables cross-embodied navigation across legged and wheeled robots and is easily steerable using natural language. Real-world ablations confirm that the specialist model is key to embodiment grounding, enabling a single high-level planner to be deployed across physically distinct wheeled and legged robots. Finally, this model significantly enhances single-robot reliability, achieving 3× higher success rates by rejecting physically infeasible plans. Website: https://vamos-vla.github.io/**

I. Introduction

A core problem in robotics is determining how robots can navigate to a goal location while traversing non-trivial terrain and obstacles. The promise of general-purpose robot navigation— i.e., performing well across diverse environments, different embodiments, and being easy to steer during deployment—has motivated a shift from hand-designed modular stacks to learning-based approaches that leverage large-scale data. Recent advances in robotic foundation models have shown that performance scales with the amount of diverse data provided [1], [2], [3], [4]. However, as datasets scale, so does their heterogeneity. This becomes a critical challenge when a downstream robot is physically incapable of achieving the entirety of behaviors recorded in a pooled, multi-robot dataset. For instance, data from a quadruped navigating stairs is of limited use to a wheeled robot. This creates a bottleneck that prevents us from naively combining all available data and achieving reliable navigation performance. In this work, we tackle the problem of effectively leveraging large-scale, combined datasets of heterogeneous locomotion capabilities for learning general-purpose cross-embodiment and steerable navigation policies.

To this end, we propose VAMOS, a hierarchical vision-language-action (VLA) model. Our key insight is that navigation can be decomposed: high-level heuristics (e.g., reaching a goal, avoiding large obstacles) are generalizable across embodiments, while low-level traversability is strictly dependent on the robot's physical capabilities. VAMOS operationalizes this insight with two main components, i.e., a *high-capacity vision-language model (VLM)* that acts as a generalist high-level planner, and a *lightweight, per-embodiment affordance model* that evaluates the feasibility of the planner's proposed actions. We train the VLM planner on diverse, real-world datasets to instill broad semantic understanding, and we train each embodiment's affordance model in simulation for efficiency and safety (Fig. 1). The interface between

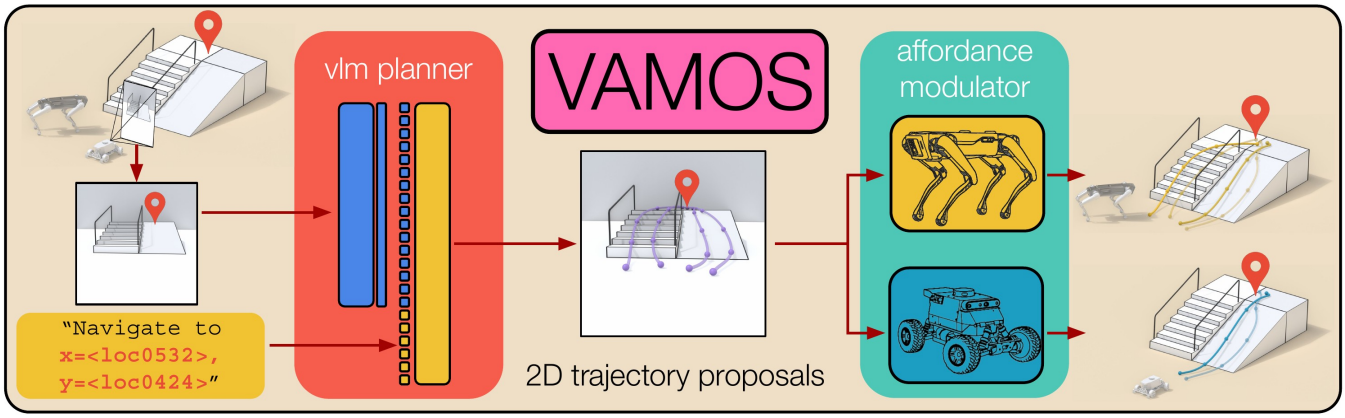


Fig. 2: The high-level planner is a VLM trained to take an image and a goal coordinate (encoded as text) as input, optionally appending natural language preferences, and to output a set of candidate paths in pixel space. These paths are encoded as strings of location token pairs, then decoded and projected from 2D pixel space to the 3D ground plane. Finally, a capability-aware affordance function evaluates and re-ranks the 3D candidate paths to determine which path the robot should execute in the real world based on low-level policy capabilities.

these models is a predicted 2D path. This path provides a structured yet flexible representation that enables our planner to leverage heterogeneous data while allowing the affordance model to modulate plans based on embodiment-specific constraints.

Through extensive real-world experiments, we demonstrate that our hierarchical approach, VAMOS, yields a new state-of-the-art in general-purpose robot navigation. We show for the first time that a structured VLA can outperform both heavily tuned modular stacks and monolithic foundation models on challenging indoor and outdoor courses. The key to this superior performance is the hierarchical design choices that successfully disentangle general planning from specific physical affordances to enable cross-embodiment transfer: we achieve high performance on both wheeled and legged robots by reusing the same high-level planner and swapping only a lightweight, specialized affordance model. Our use of a VLM also permits intuitive, natural language steerability at test time. Further, our ablations validate our core design choices, confirming that training with heterogeneous data provides significant positive transfer and that our affordance model is crucial for robust navigation.

II. Related Work

Our work builds upon three key areas of research: classical modular navigation, end-to-end learning for navigation, and hierarchical vision-language models.

Classical Modular Navigation. Navigation has traditionally been approached using modular systems with distinct components, e.g., state-estimation, perception, planning, and control [5], [6]. These methods have become the established standard in complex real-world systems due to their reliability and interpretability [7], [8]. To improve their generalization, recent efforts have incorporated learning-based components, e.g., in perception [9], [10], traversability estimation [11], [12], [13], [14], or planning [15], [16].

However, traditional modularity introduces significant limitations. First, these systems are typically heavily tuned for a specific robot embodiment and a bounded set of operating scenarios, making them brittle when deployed in new environments. Second, the intermediate representations, such as 2.5D costmaps, can abstract away valuable information and create performance bottlenecks between modules. Most importantly for our work, these systems lack cross-embodiment generalizability; transferring them to a new robot often requires re-training learned components and extensive re-tuning of the entire stack [11], [16]. Our work aims to achieve the robustness of these systems while overcoming their reliance on hand-tuning and their inability to generalize across embodiments.

End-to-End Learned Navigation and Foundation Models. To address the limitations of modular stacks, a dominant paradigm in recent years has been end-to-end learned navigation. This approach seeks to learn a direct mapping from sensor inputs to control actions, shifting the burden from manual system design to large-scale data provision. The success of foundation models in other domains has inspired similar efforts in robotics [1], [2], [3], [4], [17], which have demonstrated that policy performance scales effectively with the size and diversity of the training dataset. However, without any additional structure, these methods can be brittle during real-world deployment, e.g., they often struggle to train across widely heterogeneous datasets due to individual dataset variations in the action space.

Hierarchical Architectures and Vision-Language Models. To achieve a better balance, our work builds upon the paradigm of hierarchical models, which separate high-level planning from low-level control, the latter of which is often treated as an open-loop black box. This structure is well-established in both manipulation [18], [19] and navigation [20], [4], [3]. However, the choice of representation and the division of responsibility between the modules are critical. As our experiments later demonstrate, many prior hierarchical models underperform even traditional modular baselines in complex settings. Bidirectional influence between the VLM planner and the affordance module is necessary for robust performance.

One line of work [20], [4], [3] uses a generalist model that

takes a goal image as input and outputs a sequence of low-level velocity commands. This approach places an immense burden on a single model to both learn high-level navigation semantics and infer the specific low-level capabilities of the robot directly from observations. This conflation of tasks compromises performance on anything beyond simple, flat terrain. Moreover, it introduces a practical limitation by requiring a prior demonstration to obtain the goal image and often relies on a pre-built map for long-range navigation, limiting its applicability in unseen environments.

More recently, these hierarchical systems have been instantiated as VLAs, leveraging the semantic reasoning of pre-trained VLMs [21], [18], [22]. The method most relevant to ours is NaVILA [21], which finetunes a VLM to map a natural language command to a sequence of textual low-level actions (e.g., "Move forward 25 cm"). This approach has two key drawbacks. First, specifying precise goals via text can be tedious and ambiguous for non-object-centric navigation. Second, discrete, short-horizon textual output commands are not well-suited for long-range planning and, crucially, do not provide a natural interface for downstream modulation by an embodiment-aware module.

We designed VAMOS to overcome these limitations. By predicting a continuous 2D path as our interface, we (1) enable precise, long-range spatial reasoning, (2) do not require prior demonstrations or maps, and (3) create a representation that can be explicitly modulated by our per-embodiment affordance model. This lets our high-level planner focus solely on generalizable navigation strategy, while the affordance model assumes sole responsibility for grounding the plan in the specific robot's physical capabilities.

III. VAMOS: VLA FOR HIERARCHICAL NAVIGATION, AFFORDANCE-MODULATED AND STEERABLE

We propose a learning-based navigation algorithm, VAMOS, that can learn from large, heterogeneous datasets while maintaining awareness of embodiment-specific capabilities. To do this, we combine a high-level VLM planner with embodiment-specific, low-level locomotion affordance models, which re-rank the high-level predictions to align with robot capabilities at test time (Fig. 2). In the following subsections, we outline our high-level generalist model architecture and training paradigm (Section III-A) and describe the low-level affordance modulation (Section III-C).

A. High-Level VLM Planners from Large-Scale Datasets

A high-level generalist navigation model must be able to incorporate a variety of large-scale data sources, benefiting from their union. To this end, we build on recent advances in vision-language modeling by parameterizing our high-level generalist navigation model as a VLM. Our key design decision then became: *What choice of interface between the high- and low-level models facilitates generic training across heterogeneous datasets while effectively interfacing with embodiment-specific, low-level control?*

We **cast high-level navigation as a trajectory prediction problem**, leveraging 2D point prediction as a unifying interface for general-purpose navigation. Specifically, we train a VLM planner $P_\phi(\tau|I, g_l)$ to go from a monocular RGB image $I \in \mathscr{I}$ and target goal coordinates encoded in text g_l to predict a coarse 2D path $\tau \in \mathscr{T}$ in pixel space. The 2D path τ is a sequence of points that describes a trajectory of where the robot should move in future time-steps, projected onto the image plane for simplicity. Formally, the 2D path is defined as $\tau : (x,y)_t$, where (x,y) are normalized pixel locations of the robot's position in the frame at step t.

Our choice of parameterization has several advantages. First, it facilitates general-purpose training from a variety of data sources, with variable action spaces, unified via point prediction. Second, as noted in prior work [18], [23], training on point-level predictions helps VLMs retain much of their pre-trained generalization capabilities. The high-level VLM navigation module interfaces with a *low-level* controller π *bidirectionally* (see Section III-C); it provides waypoints for the low-level controller to track, while the low-level controller modulates the high-level predictions via its affordance function F_π.

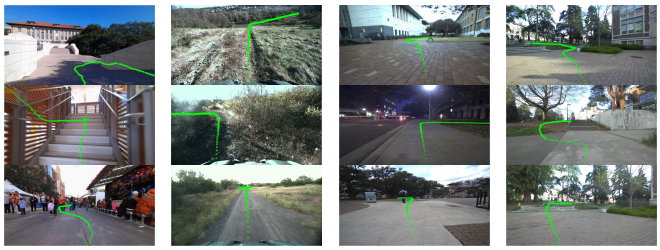
To train our steerable VLM planner, we first assemble a diverse navigation dataset mix that spans 29.8 hours and contains odometry-labeled data from 4 different robotic navigation datasets taken from 3 different embodiments (Fig. 3). We perform a series of data processing and filtering operations (Section III-B) that let us obtain higher-quality data for training our navigation generalist. From this dataset, we easily extract labeled data in the form of tuples of images and corresponding navigation paths, represented as 2D points in pixel space. We additionally annotate and augment this data with text descriptions from a state-of-the-art VLM to improve model steerability.

Given this training data, we finetune high-level VLMs to perform path predictions given input images and target goal coordinates. We perform supervised finetuning over a pre-trained `PaliGemma 2 3B` model at 224px^2 resolution [24]. We use low-rank adapters (LoRAs) since training our models using full-parameter fine-tuning vs LoRA [25] yields similar performance.

B. Training Data and Preprocessing

a) High-Level Generalist Training Data: We obtain training data for the high-level navigation module from diverse robotic navigation datasets. Since different robots may not share the same low-level action space, we align predictions across these datasets using pixel-point prediction as a unifying interface. For all data sources, we label trajectories in hindsight using camera poses at a horizon H into the future. Importantly, we use poses of the robot *on the ground* for all training data; this lets us specify goals in image space behind occluded points. We use known or estimated intrinsic and extrinsic matrices to project the 3D poses recorded in the datasets into 2D image trajectories.

We curate a diverse mix of datasets for navigation that spans different robot embodiments, camera perspectives, timing and weather conditions, and, importantly, different navigation capabilities and affordances. We perform several



(a) SCAND **(b)** TartanDrive 2 **(c)** CODa **(d)** Spot

Fig. 3: We fine-tune a VLM with navigation-specific, real-world datasets, with heterogeneous embodiments and capabilities, to obtain a general-purpose high-level planner. We use a filtered data mix from SCAND [26], TartanDrive 2 [27], CODa [28], and a small, 0.3 hour in-domain dataset collected on Spot.

data pre-processing operations on our data that are crucial for improving model performance to the point of deployability, i.e., combining both short- and long-horizon trajectories, filtering data based on curvature, and empirically determining the right data mix.

b) Steerability Recipe: The textual interface of our generalist VLM lets us provide preferences expressed as text-based instructions to steer the model's predictions at test time. To train a steerable model, we augment 10% of the data with state-of-the-art VLM annotations and co-train with two text-only visual question datasets. First, we generate 4 temporally correlated noisy versions of the ground-truth 2D trajectory τ plus a mirrored version of τ. Then, we overlay all paths onto the image I and use chain-of-thought prompting to ask `GPT-5-mini` to (1) describe the obstacles and terrain in the scene, (2) describe the paths, and (3) rank them based on their quality and diversity. We take the top three 2D paths and their respective descriptions, and we add them to our dataset. Finally, we co-train with data from the COCO-QA [29] and Localized Narratives [30] datasets to prevent forgetting.

C. Affordance Conditional Modulation

Formulation. The high-level VLM predictions are modulated by a low-level, capability-aware affordance function, which ensures that only achievable behavior is executed on hardware. The high-level navigation policy generates a set of candidate trajectories that the robot can follow to reach the goal. To pick the trajectory candidate best suited to the specific low-level locomotion policy running on the robot, we predict an affordance score $F_\pi : M \times X \times Y \times A \to [0,1]$ that jointly maps from an elevation map $M : \{1,2,\dots,W\} \times \{1,2,\dots,H\} \to \mathbb{R}$, normalized query point $x,y \in [0,1]$ position in Euclidean space around the robot, and heading angle $a \in \{0°,45°,\dots,315°\}$ to the probability that the policy π can actually traverse (x,y) in the map M when heading in direction a. This setup is inspired by the traversability estimation literature, both in simulation [13], [14] and from real-world data [11], [12]. An affordance score of 1 indicates that the point is fully traversable, while 0 indicates that the point is not traversable.

This affordance function F_π is learned via supervised learning fully in simulation by rolling out the embodiment-specific locomotion policy across a diversity of terrains. F_π enables test-time modulation of predictions from the VLM and is of benefit in two situations. First, it helps to find the candidate trajectory predicted by the VLM that is best aligned with the actual capabilities of the robot. Second, it assists with filtering out potentially noisy or infeasible predictions from the VLM, e.g., if it incorrectly predicts a path through an obstacle.

Training. Training data for learning affordance function F_π is made available by executing trajectories in *simulation* over a large variety of procedurally generated terrains using the chosen low-level policy. To collect each data point, a random elevation map M is spawned; following this, the agent is reset to a particular position (x,y) in the simulator, the policy is executed over a short horizon in a particular direction a, and binary traversal success (or failure) of the low-level policy is noted. This results in a set of data points $\mathscr{D} = \{M^{(n)}, x^{(n)}, y^{(n)}, a^{(n)}, s^{(n)}\}_{n=1}^{N}$, where $M^{(n)} \in \mathbb{R}^{W \times H}$ is a local elevation map, $(x^{(n)}, y^{(n)})$ is the queried agent position, $a^{(n)} \in \{0°,45°,\dots,315°\}$ is the heading direction, and $s^{(n)} \in \{0,1\}$ is a label representing success or failure of the trajectory. Given this training data $\mathscr{D}$, we train an affordance function F_π, represented as an MLP by minimizing a standard binary cross-entropy loss ℓ – $\mathscr{L} = \min_{F_\pi} \mathbb{E}_{M,x,y,a,s\sim\mathscr{D}} [\ell(F_\pi(M,x,y,a),s)]$.

D. Deployment

The navigation missions are defined given a series of GPS waypoints or 3D coordinates in the world frame, which are converted to 2D points in the image to be passed as input to the high-level VLM. During deployment, the VLM is first queried on the current image I and a text-encoded 2D goal coordinate g_t to obtain a set of viable paths $p_1, p_2, \dots, p_K$ in pixel space. Each pixel-space path p_i is then projected into world positions of the robot in the ground plane along each path: $\tau_i^w = \left[(x_0,y_0)^i, \dots, (x_H,y_H)^i\right]_{i=1}^{K}$ to query affordances. The affordance of each candidate path is then computed using this sequence of points along with the local elevation map M to query F_π, thereby obtaining a pointwise affordance score for each path: $\left[F_\pi(M,x_0,y_0,a_0)^i, \dots, F_\pi(M,x_H,y_H,a_H)^i\right]_{i=1}^{K}$. Finally, since a path is blocked if even one of its elements is blocked, a cumulative affordance is computed as the minimum affordance score along each path: $F^c(p_i^w) = \min\left[F_\pi(M,x_0,y_0,a_0)^i, \dots, F_\pi(M,x_H,y_H,a_H)^i\right]$. Intuitively, paths τ_i^w with higher affordances are better, while low-affordance paths are unlikely to be successfully navigated using the low-level policy π. Given this per-path measure of cumulative affordance $F^c(p_i^w)$, we can select a single trajectory to execute on the robot greedily by choosing the trajectory with the highest affordance, or we can sample with soft sampling to allow for some stochasticity in path selection: $\hat{\tau}^w \sim \text{Softmax}\left(\frac{F(\tau_1^w)}{\beta}, \frac{F(\tau_2^w)}{\beta}, \dots, \frac{F(\tau_K^w)}{\beta}\right)$.

This modulation results in a sample path $\hat{\tau}^w$ that can then be executed on the robotic hardware by commanding waypoints to the low-level policy. During deployment, we assume access to a low-level, velocity- or position-conditioned

(a) Hallway **(b)** Atrium **(c)** Lab **(d)** Campus **(e)** Forest **(f)** Down Ramp

Fig. 4: We run real-world navigation experiments indoors and outdoors in unseen scenes with challenging terrain, lighting, and vegetation. Our results show that VAMOS outperforms state-of-the-art navigation foundation models and model-based baselines.

	Indoor									**Outdoor**									
	Hallway			Atrium			Lab			Campus			Forest			Down Ramp			**Avg. SR**
Method	SR	NI	T	SR	NI	T	SR	NI	T	SR	NI	T	SR	NI	T	SR	NI	T	
Modular Stack	**100**	0	0	**100**	0	0	**100**	0.2	0	0	–	2	0	–	0	20	1	0	53
ViPlanner	**100**	0	0	**100**	0	0	0	–	0	**100**	0	0	**100**	0	0	0	–	0	67
NoMaD	60	1.3	1	0	–	3	40	2	0	0	0	5	0	–	2	60	0.7	0	27
NaVILA	20	–	1	0	–	1	40	–	0	0	–	0	0	–	1	0	–	5	10
VAMOS (Ours)	**100**	0.2	0	80	0.25	1	**100**	0	0	80	0	0	**100**	0.4	0	**80**	0.25	0	**90**

SR: Success Rate over 5 trials (%) ↑, NI: Avg. number of interventions on successful runs [0-2] ↓, T: 3 min. timeouts [0-5] ↓

TABLE I: VAMOS outperforms model-based (above the horizontal line) and end-to-end generalist navigation baselines (below the line) across a wide variety of conditions. Success Rate (SR) is computed over 5 trials for each robot-environment pair. Notably, we show that prior navigation generalists struggle to match the performance of a traditional modular stack, while VAMOS outperforms all baselines.

locomotion controller for our real-world platforms. We use the predictions of the high-level VLM in a receding horizon control fashion, where it predicts $k = 5$ waypoints but uses only the first m waypoints predicted by the high-level controller before replanning, where $m < k$ is a tunable parameter. If the goal coordinate is not in the image frame, the robot rotates in place until the goal is back in the image before replanning.

IV. Experiment Results

Out experiments evaluate the following research questions. (1) Is our hierarchical navigation method competitive with other navigation baselines in unseen environments? (2) Does our navigation method support cross-embodiment navigation? (3) Is VAMOS steerable? (4) Do we benefit from having a high-level generalist VLM compared to having a robot-specific navigator? (5) Do we benefit from low-level affordance modulation for single-robot navigation? We first describe the setup of our experiments and then walk through results pertaining to each question.

A. Experiment Setup

To validate the claims in this work, we test the methodology on two robotic platforms:

1. Legged: Boston Dynamics Spot. We evaluate performance on the BD Spot Robot using the built-in locomotion controller (capable of traversing ramps, stairs, and other terrains) as the low-level policy.

2. Wheeled: UW Hound Robot. To test transfer across embodiments, we also consider a second robot, the UW Hound [31]. Importantly, the Hound uses the same high-level VLM planner, but we simply vary the low-level affordance function and controller.

Simulation Environment. We build our simulation environment to learn the affordance function on Isaac Lab. We use a perceptive RL policy trained with reinforcement learning in simulation [32] as a proxy for the built-in BD Spot policy. To learn perceptive affordance functions that transfer well to real world, we must provide a wide diversity of terrains in simulation; during real-world deployment, there are often more distractors in the environment, such as furniture or vegetation, that must be modeled for proper sim-to-real transfer. To add diversity to our simulation environments, we generated inter-connected structures with stairs and ramps using wave function collapse. Additionally, to model irregular patterns, we used cellular automata to generate smooth, uneven terrains.

B. *Is* VAMOS *a capable navigation system in the real world?*

We compare performance between our method and other state-of-the-art baselines in terms of navigation capabilities in real-world, unseen, indoor and outdoor environments (Fig. 4). The chosen baselines are (1) a geometric model-based modular navigation stack similar to [7], (2) ViPlanner [15], a learned geometric and semantic planner, (3) NoMaD [3], a navigation foundation model, and (4) NaVILA [21], a navigation VLA. We focus on a short- to medium-horizon range for goal navigation, where the goal position is specified in 3D global coordinates. To reach long-range goals, we generate waypoints to the goal every ~ 10 meters (Fig. 5).

The "Hallway" course ($\sim 20m$) tests the ability to navigate down narrow corridors with tight turns. The "Atrium" course ($\sim 20m$) measures the ability to navigate cluttered open scenes in low light. The "Lab" ($\sim 5m$) course tests the ability to navigate to a point occluded by a large irregular obstacle. The "Campus" ($\sim 40m$) course tests the ability to navigate long distances, including going up a 7-step staircase. The "Forest" ($\sim 20m$) course tests the ability to navigate in vegetated environments that including stairs; rooted and vegetation-covered terrain; irregular concrete paths; and paths with overhanging vegetation. Finally, the

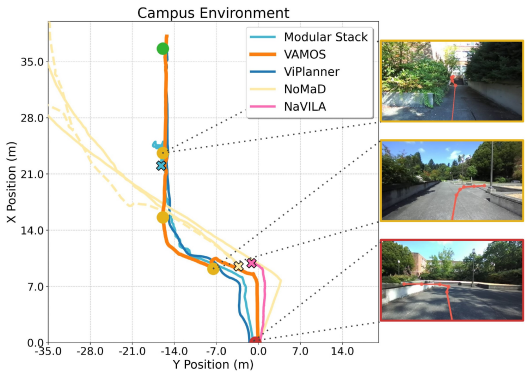


Fig. 5: Top-down map showing paths taken by different methods from start (red) to goal (green) through waypoints (yellow). VAMOS achieves long-horizon, precise navigation. Right: predicted and selected paths when replanning after reaching a waypoint. Dotted lines show returns to the last completed waypoint after interventions; X's mark baseline failures or timeouts.

"Down Ramp" ($\sim 15m$) course tests the ability to navigate to a point below the start pose, evading foot-snaring vines.

We present the results in Table I. VAMOS *achieves higher average success rate across all courses, performing well across all conditions, which no other baseline does.*

In indoor environments, VAMOS performs on par with the modular stack and ViPlanner, with the exception being the more challenging "Lab" course, where it outperforms all baselines. This is because the inferred geometric costmaps indoors are clean and easy to plan against. However, two generalist baselines, NoMaD and NaVILA, struggle to generalize out-of-distribution, even though they were both trained using indoor data similar to our data mix, and mainly navigate in straight lines or bounce off walls. We credit VAMOS's superior performance to our usage of 2D trajectories, which have been shown to maintain more of the pre-trained VLM's generalization capabilities [18].

VAMOS also excels in outdoor urban and off-road environments. Neither the modular stack nor the generalist baselines perform well outdoors. The geometric modular stack fails at the interface of perception and planning, where inaccurate perception leads to downstream failures. The generalist baselines fail because in more open environments, they mainly walk in straight lines. ViPlanner performs well due to its well-tuned geometric and semantic perception integration. However, in both the "Lab" and "Down Ramp" environments, which are challenging due to large geometric obstacles that require long-term planning, ViPlanner fails to reason about long-term outcomes. *These experiments highlight* VAMOS*'s rich geometric and semantic reasoning capabilities, resulting in a significantly higher overall average success rate (90%) compared to the baselines.*

C. *Does* VAMOS *support cross-embodiment navigation?*

We evaluate the cross-embodiment capabilities of our method on a simple test environment consisting of a staircase and a ramp, side-by-side, leading to an elevated floor, as shown in Figure 6. We use the same high-level planner for both Spot and Hound robots, and we swap only the embodiment-specific affordance module. First, we show that

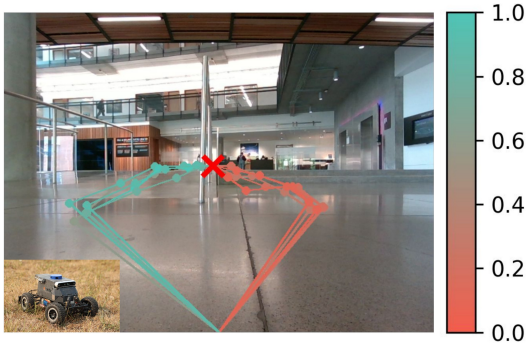


Fig. 6: We evaluate the cross-embodiment capabilities of VAMOS on a wheeled robot, Hound, where the goal (red X) is to reach an elevated floor through either a ramp or stairs. We show 10 candidates predicted by the VLM and their corresponding affordance score. Reranking with the affordance function enables higher success rates in cross-embodied navigation as shown in Table II.

	Spot			Hound		
	Stairs	Ramps	SR	Stairs	Ramps	SR
No Modulation	4/10	6/10	100%	4/10	6/10	60%
With Modulation	8/10	2/10	100%	1/10	9/10	**90%**

TABLE II: Affordance modulation maintains Spot's perfect performance and improves Hound's overall success rate from 60% to 90%, enabling cross-embodied navigation. Counts show per-terrain path choices (teal = success, red = failure) over 10 runs.

Method	**Spot**	**Hound**
ViPlanner	100%	0%
VAMOS	100%	**90%**

TABLE III: VAMOS outperforms the best navigation baseline in cross-embodiment tasks, selecting ramps vs. stairs aligned with robot capabilities via its affordance model (N=10).

affordance modulation lets the same VLM predictor be used effectively with two different robot embodiments, enabling navigation for both platforms. As we show in Table II, the same VLM *with* affordance modulation enables accurate navigation for both legged and wheeled platforms, taking specific robot capabilities into account. In this case, the wheeled robot can only take the ramp, while the legged robot can succeed on both stairs and ramps. In contrast, executing VLM predictions without affordance modulation often results in predictions that are not achievable under the current low-level embodiment. [1]

Compared to the best performing method in Table I, ViPlanner, we show that our method achieves almost perfect success rates on both embodiments, while ViPlanner fails when deployed on Hound, as shown in Table III. By swapping affordance models that are cheap to train and run, we obtain performant cross-embodiment navigation.

[1]To improve multimodal generation in this experiment, we collected 50 static images with slight pose variations from each robot in that environment, labeled each with a path going up stairs and a path going up ramps, and then generated 10 noisy samples per hand-drawn trajectory to generate the dataset that we used to finetune the base VAMOSVLM planner. This helped more clearly illustrate the differentiation provided by the affordance function.

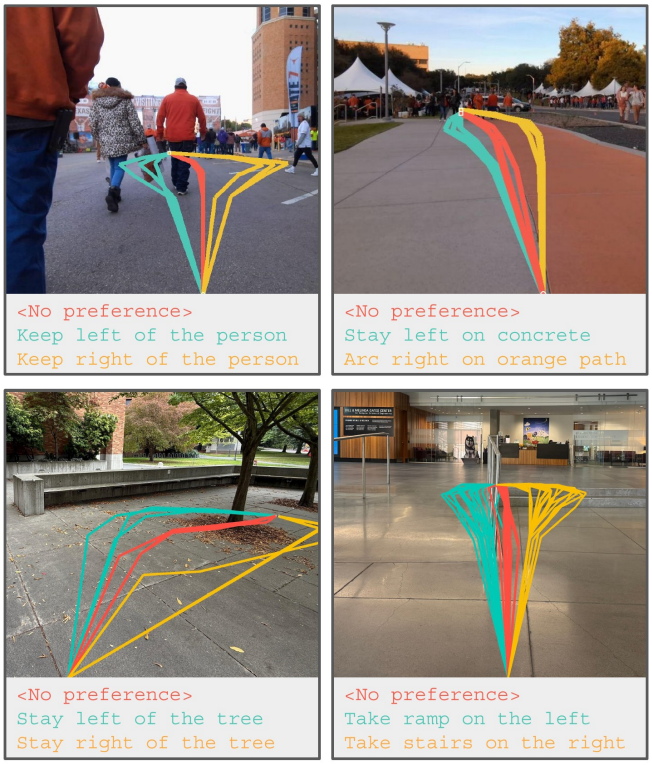


Fig. 7: VAMOS is steerable through natural language preferences appended to its goal coordinate specification. Different preferences are indicated by the shown natural language prompts and depicted using different colors.

D. Is VAMOS *steerable via natural language?*

We evaluate the steerability of our model qualitatively and quantitatively. In Figure 7, we show examples of the 2D paths predicted by VAMOS with and without preferences appended to the text input that encodes the goal coordinate. As shown in Figure 7, we can adapt the output trajectories to follow a particular direction (left or right) or to take a particular terrain (stairs, ramps, or grass planters). Using VLM-as-a-judge (ChatGPT 5) on the bottom-right image in Figure 7 , we obtain 20/20 preference alignment when specifying which path to take for both the ramps and the stairs compared to the original trajectories without pre-specified preferences.

E. Does the high-level VLM generalist provide benefits over a robot-specific navigator?

To understand whether training a generalist VLM policy is actually beneficial, we perform an analysis of offline model performance. Specifically, we aim to answer whether pooling data from the heterogeneous datasets in Figure 3 is beneficial compared to simply training the model on single, robot-specific datasets. We compare the performance of the high-level VLM predictor on path prediction across mean L2 prediction error as a metric. Specifically, we compare the performance of a model trained on a pooled dataset across all the datasets mentioned in Figure 3 to the performance of a model trained on each individual dataset. The results in Figure 8 indicate that pooling data results in better performance than training on specific datasets.

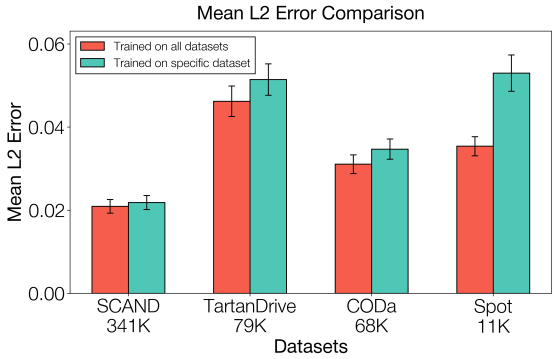


Fig. 8: Pooling data across all robot datasets (red) improves model performance compared to training specialist navigation models on individual robot datasets (teal). We evaluate over the entire validation set. Error bars represent 95% CI.

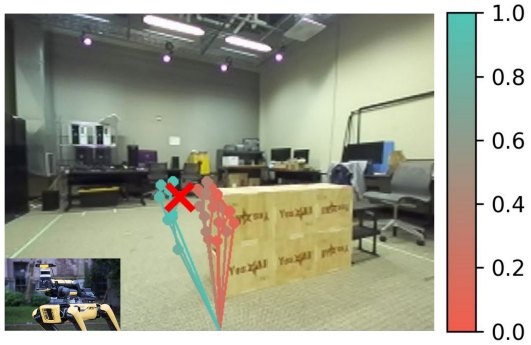


Fig. 9: The affordance function also helps with filtering out noisy VLM predictions in single-embodiment OOD settings. In this example, it filters out the paths predicted by the VLM that go through obstacles to reach the goal (red X), leading to higher success rate (Table IV).

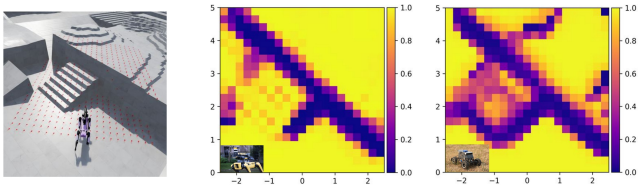
Condition	Success Rate
No Modulation	20.0%
With Modulation	**60.0%**

TABLE IV: Affordance modulation reduces high-level VLM prediction errors in cases where the VLM predicts paths that violate the robot's capabilities or physics. Success rate is over 10 runs.

F. Do we benefit from low-level affordance modulation for single-robot navigation?

Next, we evaluate whether modulation with the affordance function can improve model performance with a single embodiment by correcting for VLM errors. We show quantitatively in Table IV that the VLM performance without modulation can make mistakes in OOD settings, such as going through obstacles, that are corrected by the affordance function modulation. The same can be seen qualitatively in Figure 9, where affordance modulation prevents the execution of catastrophic paths suggested by the VLM.

Finally, we visualize the affordance function in Figure 10. We see that it naturally captures the geometry of the environment and the particular agent's capabilities. Projecting this affordance function onto the VLM predictions prevents mistakes like navigating directly into obstacles.



(a) Scene Geometry **(b)** Spot Affordance **(c)** Hound Affordance

Fig. 10: The affordance function indicates that the Spot robot can ascend stairs, but the wheeled Hound cannot (yellow signifies high-affordance score). Both robots cannot traverse tall obstacles (e.g., the wall has a low-affordance score).

V. Conclusion

We presented VAMOS, a technique for general-purpose navigation using vision-language models. The central idea in this work is to combine diverse, heterogeneous datasets for training a hierarchical VLA model. The high-level VLM planner predicts candidate navigation paths as 2D pixel paths. This output is modulated by a low-level affordance model that enables capability- and embodiment-aware navigation on deployment. We show significantly improved performance over both model- and learning-based baselines in our extensive real-world navigation experiments. The resulting methodology provides a step towards open-world, general-purpose navigation agents that can reason both geometrically and semantically about how to act in the world.

Acknowledgements

This research was partly funded by the Army Research Lab (ARL) and compute provided by the University of Washington Hyak program. The authors would like to thank Xiangyun Meng and Yi Li for early discussions and feedback. The authors would also like to thank Khimya Khetarpal, Daphne Chen, Brady Moon, Gokul Swamy, Alex Stephens, and Swapnil Pande for presentation feedback, as well as other members of Robot Learning Lab and WEIRD Lab at University of Washington for feedback and support. Finally, the authors would like to thank the authors of the baselines we compared against for providing their code.

References

[1] Physical Intelligence. π_0: A vision-language-action flow model for general robot control, 2024.

[2] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[3] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey 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] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. In *Conference on Robot Learning*, pages 711–733. PMLR, 2023.

[5] Charles Thorpe, Martial H Hebert, Takeo Kanade, and Steven A Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988.

[6] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 25(8):425–466, 2008.

[7] Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, JoonHo Lee, Wentao Yuan, Zoey Chen, Samuel Deng, et al. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. *arXiv preprint arXiv:2303.15771*, 2023.

[8] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66):eabp9742, 2022.

[9] Amirreza Shaban, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox. Semantic terrain classification for off-road autonomous driving. In *Conference on Robot Learning*, pages 619–629. PMLR, 2022.

[10] Gian Erni, Jonas Frey, Takahiro Miki, Matias Mattamala, and Marco Hutter. Mem: Multi-modal elevation mapping for robotics and learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11011–11018. IEEE, 2023.

[11] Mateo Guaman Castro, Samuel Triest, Wenshan Wang, Jason M Gregory, Felix Sanchez, John G Rogers, and Sebastian Scherer. How does it feel? self-supervised costmap learning for off-road vehicle traversability. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 931–938. IEEE, 2023.

[12] Matías Mattamala, Jonas Frey, Piotr Libera, Nived Chebrolu, Georg Martius, Cesar Cadena, Marco Hutter, and Maurice Fallon. Wild visual navigation: Fast traversability learning via pre-trained models and online self-supervision. *arXiv preprint arXiv:2404.07110*, 2024.

[13] Jonas Frey, David Hoeller, Shehryar Khattak, and Marco Hutter. Locomotion policy guided traversability learning using volumetric representations of complex environments. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5722–5729, 2022.

[14] Pascal Roth, Jonas Frey, Cesar Cadena, and Marco Hutter. Learned perceptive forward dynamics model for safe and platform-aware robotic navigation. *arXiv preprint arXiv:2504.19322*, 2025.

[15] Pascal Roth, Julian Nubert, Fan Yang, Mayank Mittal, and Marco Hutter. Viplanner: Visual semantic imperative learning for local navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5243–5249. IEEE, 2024.

[16] Matt Schmittle, Rohan Baijal, Nathan Hatch, Rosario Scalise, Mateo Guaman Castro, Sidharth Talia, Khimya Khetarpal, Byron Boots, and Siddhartha Srinivasa. Long range navigator (lrn): Extending robot planning horizons beyond metric maps, 2025.

[17] Google Robotics. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint*, 2023.

[18] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, et al. Hamster: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025.

[19] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[20] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. *arXiv preprint arXiv:2210.03370*, 2022.

[21] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. In *RSS*, 2025.

[22] Catherine Glossop, William Chen, Arjun Bhorkar, Dhruv Shah, and Sergey Levine. Cast: Counterfactual labels improve instruction following in vision-language-action models. *arXiv preprint arXiv:2508.13446*, 2025.

[23] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[24] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[26] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter 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.

[27] Matthew Sivaprakasam, Parv Maheshwari, Mateo Guaman Castro, Samuel Triest, Micah Nye, Steve Willits, Andrew Saba, Wenshan Wang, and Sebastian Scherer. Tartandrive 2.0: More modalities and better infrastructure to further self-supervised learning research in off-road driving tasks. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12606–12606. IEEE, 2024.

[28] Arthur Zhang, Chaitanya Eranki, Christina Zhang, Ji-Hwan Park, Raymond Hong, Pranav Kalyani, Lochana Kalyanaraman, Arsh Gamare, Arnav Bagad, Maria Esteva, et al. Towards robust robot 3d perception in urban environments: The ut campus object dataset. *IEEE Transactions on Robotics*, 2024.

[29] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[30] Jordi Pont-Tuset, Jasper Uijlings, Soravit Changpinyo, Radu Soricut, and Vittorio Ferrari. Connecting vision and language with localized narratives, 2020.

[31] Sidharth Talia, Matt Schmittle, Alexander Lambert, Alexander Spitzer, C Mavrogiannis, and Siddhartha S Srinivasa. Hound: An open-source, low-cost research platform for high-speed off-road underactuated nonholonomic driving. *arXiv preprint arXiv:2311.11199*, 2023.

[32] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[33] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

[34] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

[35] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *European Conference on Computer Vision*, pages 18–35. Springer, 2024.

[36] Riku Murai, Eric Dexheimer, and Andrew J Davison. Mast3r-slam: Real-time dense slam with 3d reconstruction priors. *arXiv preprint arXiv:2412.12392*, 2024.

[37] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. *arXiv preprint arXiv:2409.19152*, 2024.

[38] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. *arXiv preprint arXiv:2407.10353*, 2024.

[39] Joey Hejna, Chethan Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

[40] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.

[41] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[44] Takahiro Miki, Lorenz Wellhausen, Ruben Grandia, Fabian Jenelten, Timon Homberger, and Marco Hutter. Elevation mapping for locomotion and navigation using gpu. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2273–2280. IEEE, 2022.

[45] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. *Advances in neural information processing systems*, 16, 2003.

APPENDIX I
HIGH-LEVEL TRAINING DETAILS

A. Hyperparameters and Compute

We present all our training hyperparameters for the high-level VLM in Table V. We find that training for multiple epochs lends to rapid overfitting, so we train our model for 1 epoch using an Nvidia L40 node of 8 GPUs with a per-device batch size of 8 for about 5 hours. Notably, we find that it is possible to fine-tune our model with LoRA on a consumer-grade Nvidia RTX 4090 GPU, albeit with a much smaller per-GPU batch size of 2. We take advantage of state-of-the-art training infrastructure for large language models (LLMs) by integrating our training with the HuggingFace ecosystem, using the TRL library [33] with data-parallelism implemented by the `accelerate` library [34].

TABLE V: Key Training Hyperparameters

Hyperparameter	Value
Base Model	`google/paligemma2-3b-pt-224`
Seed	42
Optimizer	`adamw`
Learning Rate	1e-4
Adam β_1	0.9
Adam β_2	0.999
Adam ε	1e-8
Weight Decay	1e-5
Max Grad Norm	1.0
LR Scheduler	Cosine
Warmup Ratio	0.1
Num Train Epochs	1
Batch Size (per device)	8
Gradient Accumulation Steps	1
Num GPUs	8
Effective Batch Size	64
Precision	`bfloat16`
Max Sequence Length	2048
Data Packing	True
LoRA Specific Parameters (PEFT)	
LoRA R (Rank)	16
LoRA Alpha	16
LoRA Dropout	0.05
LoRA Target Modules	`q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`

B. Dataset Preparation and Mixtures

a) Dataset Processing: We perform several data pre-processing operations on our data that allows us to obtain higher-quality data for training. Notably, scaling up navigation datasets naively leads to a lot of data where the navigator mostly walks or drive straight. We balance short and long-range trajectories by sampling from two different horizons at a 50% ratio, which increases the diversity in paths while maintaining effective short-range navigation. Given that much of the data in these datasets is highly-correlated, we also filter the number of trajectories to maintain the most salient examples. To do this, we rank trajectories based on curvature, defined as the ratio between the ground-truth trajectory length and the straight-line distance to the goal, i.e., $c = \frac{\sum_{i=1}^{k-1} \|w_t^{i+1} - w_t^i\|}{\|w_t^k - w_t^1\|}$, where $\tau_t = \{w_t^1, \ldots, w_t^k\}$, and we select the top n data points based on curvature, where n varies based on dataset. The odometry in these datasets can be noisy, so we also filter out the top 3% of trajectories based on this curvature metric to reject noisy samples. Finally, we align the 2D image coordinate representation of the goal to the tokenization scheme of the pre-trained PaliGemma 2 model – in particular, location tokens are represented using 1024 discrete location tokens (`<loc0000>` to `<loc1023>`) corresponding to binned normalized image coordinates. We convert the goal to a text instruction of the form `"Navigate to x=<locXXXX>, y=<locYYYY>"`, which then gets tokenized and passed into the model alongside the image. If a natural-language preference is specified, we append this preference to this string, e.g., `"Navigate to x=<locXXXX>, y=<locYYYY>.Stay on the right of the people."`

Dataset	Hours	Used (%)
SCAND [26]	19.5	351.2K (50%)
CODa [28]	7.8	70.5K (25%)
TartanDrive 2 [27]	2.2	79.1K (100%)
Spot	0.3	11.2K (100%)
Human Sketch (FT)	–	2K (100%)
Total	29.8	514K

TABLE VI: Dataset mix used for training high-level navigation. All datasets include odometry annotations. Human sketch annotations enable few-shot adaptation (Section IV-C).

b) Dataset Mixtures:: In early experiments, we found training with all the data, or training with a uniformly subsampled percentage of all the data performed worse than our data mix detailed in Table VI. We arrived to this mix heuristically: we found that the SCAND dataset [26] contains high-quality, diverse data, so we keep a high proportion of it, whereas the CODa dataset [28] is very repetitive, covering very similar scenes throughout the dataset, although at higher variations of weather and lighting conditions, so we down-weight it. We keep the full datasets for the TartanDrive [27] and in-domain Spot datasets given their relatively-small size, although we filter out trajectories with noisy odometry from the TartanDrive dataset. Finally, we consider only the Spot subset of data from the SCAND dataset [26] given that we could not obtain accurate camera parameters for the Jackal subset.

We also experimented with two additional sources of non-robot data: videos processed with monocular tracking [35] or structure-from-motion algorithms [36], [37], and data collected with paired odometry using an iPhone and its built-in odometry estimation through ARKit, similar to [38]. For unlabeled egocentric videos, we obtain estimated camera poses using the CoTracker video tracker model [35], similar to [16], which tracks a grid of 2D points on a video. To obtain trajectories using CoTracker, we run egocentric videos in reverse and track a subset of the grid of points sampled on the ground in front of the camera to obtain a sequence of traversed points. We collected a dataset of 3.7 hours,

with 133K data points, of in-domain walking data collected with an Insta360 fisheye camera. We found that adding this data to our data mix hurt performance. CoTracker struggles with maintaining trajectories behind occlusions, which leads to shorter ground-truth trajectories, noisy data, and mostly straight paths. We also experimented with Mast3r-SLAM [36], and while it handled occlusions better, processing long-horizon trajectories was too computationally inefficient.

The second source of data we experimented with was odometry-labeled videos collected with an iPhone and labeled with ARKit. This allowed us to collect data at a much higher speed than through robot data collection. Most of our efforts focused on collecting data to improve the multi-modal capabilities of the robot, by starting at similar positions but taking different paths to reach the goal. We focused on collecting data going up stairs and ramps to support our experiment in Section IV-C. We collected a dataset of 81K data points (about 2.3 hours). However, we found that using the entirety of this data lead to more twisty predicted trajectories throughout, and hurt quantitative metrics. We believe this data collection approach to be quite promising, both due to its ease of scalability and potential for collecting targeted data beyond what is usually found in internet and existing robotic datasets. We leave finding better ways to select data mixes from diverse sources such as the ones described in this section [39] for future work.

APPENDIX II
QUALITATIVE RESULTS

We show a visualization and top-down map ofall real-world navigation courses in Figure 11. We also show some examples of the predicted and ground truth trajectories in each dataset in Figure 12. Our model is good at following paths and trails, going behind obstacles and occlusions, and reaching the goal. Given that the ground-truth trajectories are long-horizon, sometimes these paths take roundabout ways to reach the goal. Our high-level VLM often takes direct paths to the goal. We show some of the VLM's failure modes in Figure 13. Two salient failure modes are dynamic obstacles and over/under-shooting turns. Given the staticness of the training data, the model does not capture close-by dynamic obstacles, such as walking people, very well. Additionally, the model sometimes overshoots or undershoots turns behind obstacles and occlusions. Sometimes this occurs due to the way in which we subsample trajectories uniformly, causing clipping at important points of the trajectories. However, our experiments with other subsampling methods that aim to capture salient points, such as the Ramer-Douglas-Peucker algorithm [40], [41] (as is done in [18]) showed that this type of sampling hurt performance.

APPENDIX III
TRAVERSABILITY FUNCTION DETAILS

A. Terrain Generation

For adequate sim to real transfer we found that it was important to generate a varied set of terrains(Figure 14) to simulate the diversity of the real world. We chose to use 5 different terrain types: irregular stairs, smooth mounds, procedurally generates stair and ramp environments, simple ramp, and simple stairs.

Simple Stairs: For the simple stairs terrain we have a 2m long by 10m wide flat area on both sides of the terrain, then a 6m long by 10m wide staircase connecting the two flat areas. The step width is set to 0.4m and the step height is drawn from a uniform distribution from 0.05m to 0.15m.

Simple Ramp: The simple ramp is similar to the simple stairs except the two flat regions are connected by a ramp. The slope of this ramp is drawn from a uniform distribution from 0.01m to 0.3m.

Procedural Terrain: The procedural terrain is composed of 25 two by two square tiles. Each tile in the terrain can either be a box, ramp, stairs, or flat. Then we use wave function collapse to populate all of the tiles and ensure they adhere to certain rules. We want stairs to either connect to other stairs or a flat area, and we want the area at the top of the stairs to be at the same height as the top of the stairs. Additionally, we randomize the heights of each stair or ramp and the sizes and heights of the boxes.

Smooth Mounds: To generate smooth mounds we use cellular automaton. We first choose n random cells in our height map to serve as seeds. Each of these positions is set to some height drawn from a uniform distribution between 1m and 3m. Then we set the value of every cell in the heightmap to the value of the closest seed. Finally, to smooth everything out, for each cell in the height map if the difference between the minimum and maximum neighbor is greater than some threshold, then we set the height at the cell to the mean of those two neighbors. In practice, we have found that this algorithm is able to generate irregular terrains similar to uneven ground outdoors.

Irregular Stairs: To generate a irregular terraced pattern, we first use the cellular automaton to generate the smooth mounds. Then given some step height, we round the heights of each cell to the nearest whole number multiple of the step height. This terrain is meant to make our value function more robust to sharp local changes in elevation.

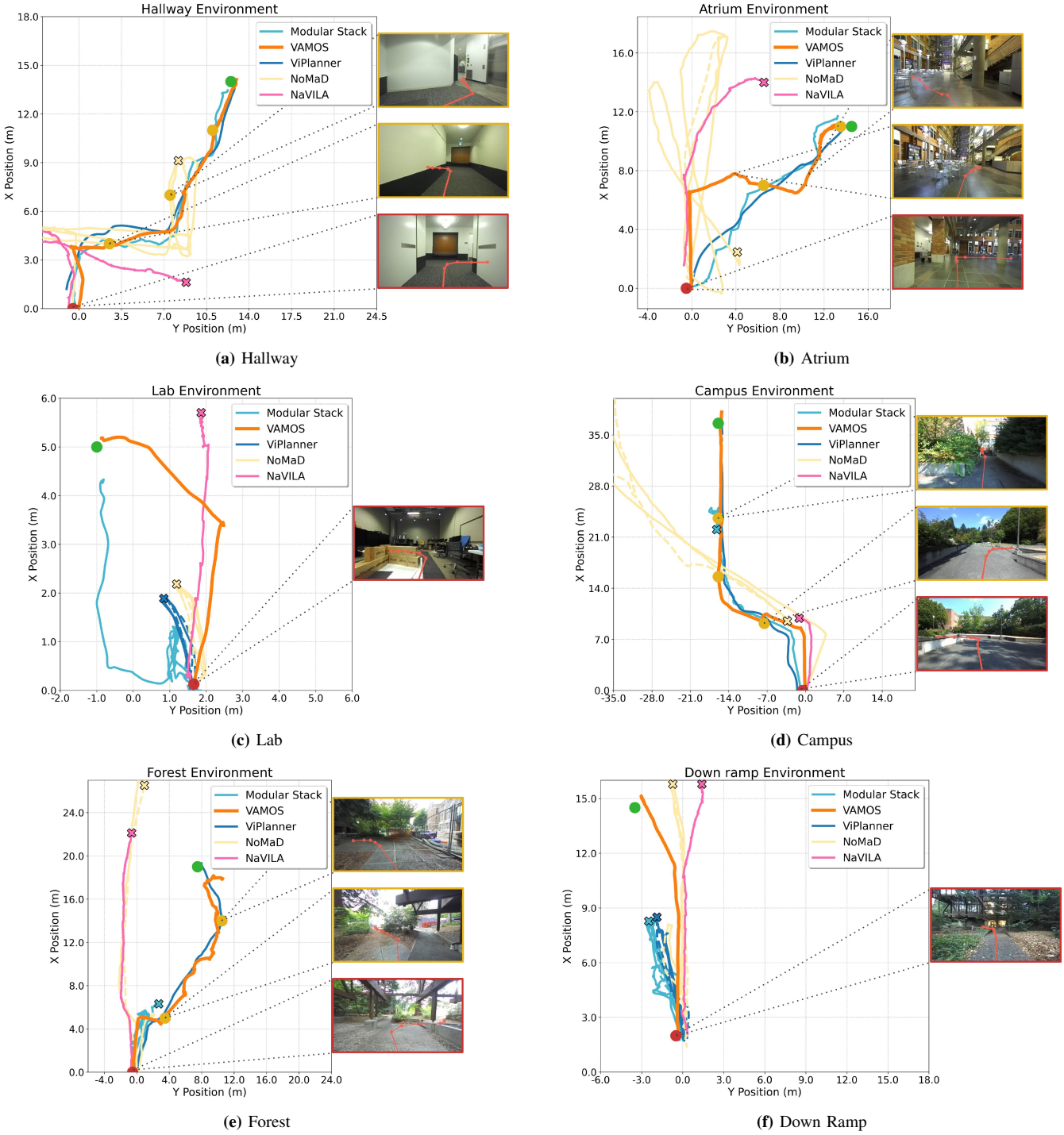
B. Dataset Generation

We first generate 1000 different 10m by 10m terrains using the methods outlined above, with all of the terrain types being equally represented. Then we depending on the robot type we have slightly different methods for data collection.

Spot: For spot we select a uniformly random unit vector as our velocity target. Then we roll out the policy until it terminates or times out. We terminate the roll-outs when the robot hits the wall which we compute by Equation 1 where v_r is the robot velocity vector, v_c is the command velocity vector, and τ is some threshold.

$$\mathbf{1}\left\{\frac{v_r \cdot v_c}{\|v_c\|} < \tau\right\} \tag{1}$$

In practice, we use $\tau = 0.3$. Additionally, we only compute this termination after the first 0.5 seconds to allow the robot to initially accelerate. The other termination that we use is a



(a) Hallway **(b)** Atrium

(c) Lab **(d)** Campus

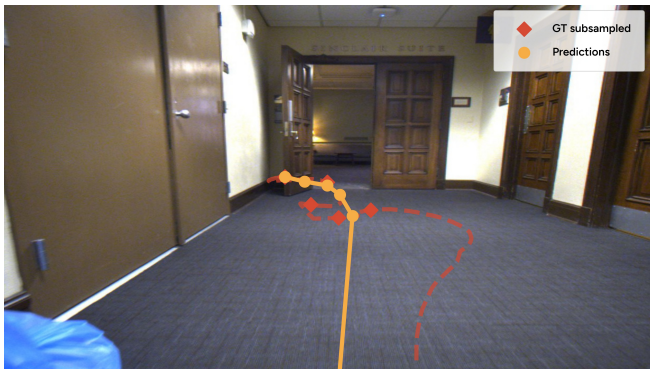
(e) Forest **(f)** Down Ramp

Fig. 11: We show a top-down map for all navigation courses of the paths taken by different methods to navigate from the start of the course (red circle) to the goal (green circle), through each waypoint (yellow circles). VAMOS is capable of long-horizon, precise navigation. To the right, we visualize the paths predicted and selected by VAMOS when replanning after reaching a waypoint. Dotted lines correspond to taking the robot back to the last previously-completed waypoint after interventions, and X's correspond to the positions where baselines failed or timed-out.

penalty for falling when either the velocity in the z direction is less than -1 or the robot is tilted more than 45° degrees on the roll or pitch axes. The rewards for each timestep correspond to the terminations. We have a reward of -1 when we terminate.

The policy we roll out in simulation is trained with PPO [42] in rough terrains using Isaac Lab [32] with proprioceptive and perceptive observations consisting of geometric height samples, following a terrain curriculum, similar to [43]. Even though this is a different policy than the built-in

(a) SCAND



(b) CODa

(c) TartanDrive

(d) Spot

Fig. 12: Examples of high-level trajectory predictor. The high-level navigator consistently gets to the goal, is good at following paths, going around obstacles, and taking turns behind occlusions such as walls, people, poles, etc.

(a) Dynamic Obstacles

(b) Overshooting

Fig. 13: Examples of failure modes. The high-level navigator sometimes struggles with dynamic obstacles, as in the training data dynamic obstacles usually move out of the way and their motion is not captured in the training data. It also sometimes overshoots or undershoots turns.

Spot locomotion policy we use during deployment, it acts as a good surrogate for learning the capabilities of a performant all-terrain navigation policy.

Hound: For Hound we collect all trajectories by driving in a straight line because the affordances of the car over a small distance tend to be the same while driving strait and turning. We use the same terminations and rewards as Spot.

Instead of using a learned or default low-level controller, we use a pure-pursuit based controller with a kinematic bicycle model to reach various waypoints.

C. Qualitative Comparison to Regression-based Value Function

Rather than training the model to classify local elevation maps as failures or successes, we can compute the discounted sum of rewards for each rollout and train the model to regress this value given the local elevation map (see Figure 15). However, in practice we observe that classifying faliure or success of a trajectory works better than regressing reward to go. We believe that the reason for this is that we can easily balance the classification dataset to include an equal

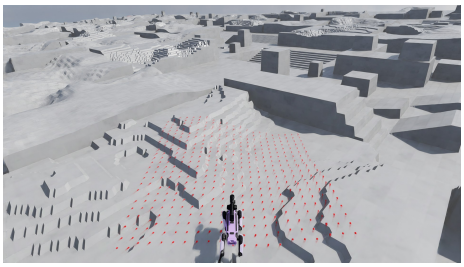
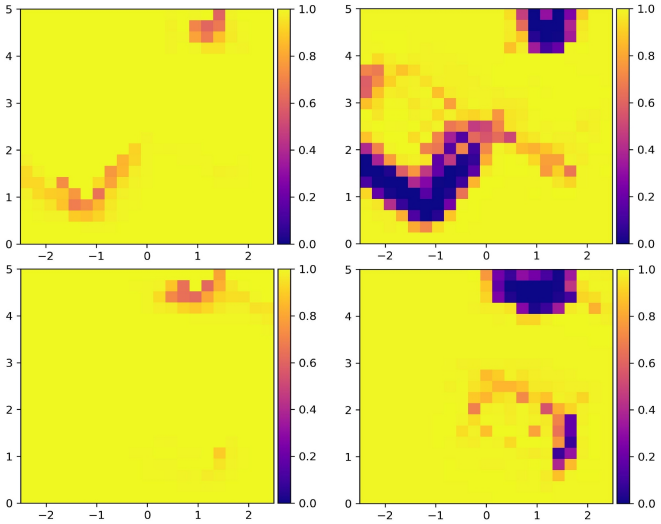


Fig. 14: We generate terrains in simulation to train the affordance function using a combination of procedurally generated stairs- and ramp-like terrains with different parameterizations.



(a) Regression on Returns **(b)** Classification on Labels

Fig. 15: Computing the affordance function as a classification task rather than a regression on returns, as is common in reinforcement learning, yields more discriminative affordance scores. Here, we show two examples from two different scenes, where each row represents a scene.

proportion of successes and failures. Additionally, we believe the classification problem better represents the task because avoids coupling the labels of unrelated observations.

APPENDIX IV
DEPLOYMENT

A. Compute and Sensors

We deploy our fine-tuned PaliGemma 2 high-level navigation generalist on an external laptop with an Nvidia RTX 3080 Laptop GPU, while the low-level low-level traversability function runs onboard a Jetson Orin AGX. High-level inference runs at 1 Hz when the laptop is connected to external power, and around 0.5 Hz otherwise. For Spot experiments, the laptop is connected to the robot's network through ethernet for increased reliabilty. For HOUND experiments, the laptop is connected to the robot through a 5 GHz Wi-Fi hotspot running on the laptop. We found 5GHz Wi-Fi to provide much better capacity and latency over 2.4 GHz Wi-Fi, albeit less reliable outdoors.

As sensor readings for the traversability function, we use an Ouster OS-1 LiDAR and Spot's built-in depth cameras on all sides of the robot to construct a square 16x16 meter elevation map using [44], from which we crop smaller local grids for the traversability function observations. We use a Zed2i camera for the Spot robot and a Realsense D455 camera for the HOUND robot.

B. VLM Sampling

For the navigation experiments outlined in Section IV-B, we sample the VLM with `temperature` $= 0.1$, `num_beams` $= 1$, and no `top_k` nor `top_p` sampling. For the traversability function experiments outlined in Section IV-F, the only difference is that we sample the VLM with `temperature` $= 1.0$ for the obstacle avoidance experiments and `temperature` $= 0.3$ for the embodiment experiments.

C. State Machine

We deploy VAMOS in the real world within a simple state machine. First, the robot either plans with the VLM and then tracks the first m out of 5 predicted waypoints, or it rotates in place until the goal is within the image frame and then plans with the high-level VLM. Then, after either reaching the first m waypoints, or after a timeout set to 20 seconds, whichever happens first, we repeat the loop and re-plan or rotate to put the goal within the image frame. We find that $m = 3$ works well for shorter courses, and $m = 4$ works well for longer courses.

APPENDIX V
MODULAR STACK BASELINE DETAILS

For completeness, we provide a detailed description of our "Modular stack" baseline, which is highly performant and serves as a comparison point in our experiments. This baseline includes robust state estimation, global and local path planning, terrain analysis, and a strong low-level control module:

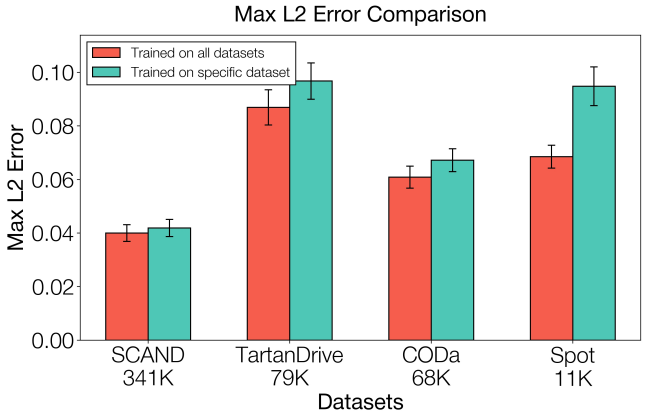
- State Estimation: We use Spot's built-in visual odometry, a production-level odometry system deployed across all Spot robots.
- Traversability Analysis: The geometric costmap from Multi-Modal Elevation Mapping (MEM, [10]) is employed for terrain assessment. This is the same costmap utilized in prior works such as [12].
- Global Planning: Using the MEM costmap, we employ ARA* [45], an incremental, anytime variant of A*, for efficient global path planning.
- Local Planning: A pure-pursuit controller is used locally. This approach achieves performance comparable to MPPI with a kinematic bicycle model while being simpler and computationally cheaper.
- Low-Level Control: Spot's built-in RL-MPC locomotion controller handles low-level control, providing a robust, production-ready policy for navigating challenging terrains.

APPENDIX VI
ADDITIONAL QUANTITATIVE RESULTS

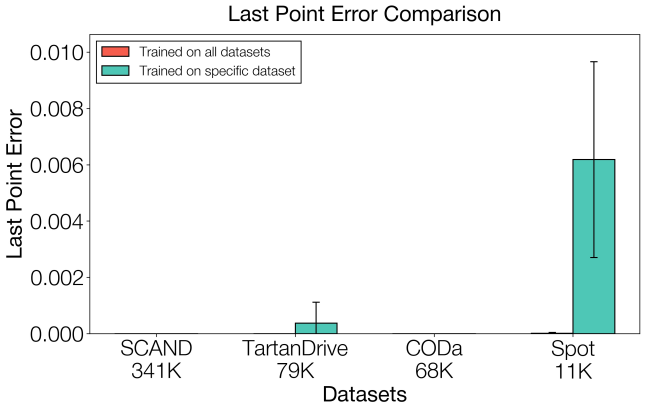
We compare our high-level generalist with robot-specific models using additional metrics to measure offline performance as mentioned in Section IV-E. We consider the following metrics, as shown in Figure 17:

- **Mean L2 Error:** (Fig. 8) Measures the error for all 5 points in a predicted trajectory, averaged across each trajectory, across all trajectories in the validation dataset.
- **Max L2 Error:** (Fig. 16a) Measures the maximum error between all 5 points in a predicted trajectory, averaged across all trajectories in the validation dataset.
- **Fréchet Distance on Subsampled Trajectories:** (Fig. 16c) Measures the Fréchet distance between the subsampled ground-truth trajectory (i.e. the 5 points used as labels during training) and the predictions. Similar to a `max` function.
- **Fréchet Distance on Full Trajectories:** (Fig. 16d) Measures the Fréchet distance between the full, dense, ground-truth trajectory (i.e. the original trajectories of hundreds of datapoints, depending on the horizon, subsampled at 10 Hz) and the 5-point predictions. Similar to a `max` function.
- **Dynamic Time Warping Distance on Subsampled Trajectories:** (Fig. 16e) Measures the normalized Dynamic Time Warping distance between the subsampled ground-truth trajectory (i.e. the 5 points used as labels during training) and the predictions. Similar to a `mean` function.
- **Dynamic Time Warping Distance on Full Trajectories:** (Fig. 16f) Measures the normalized Dynamic Time Warping distance between the full, dense, ground-truth trajectory (i.e. the original trajectories of hundreds of datapoints, depending on the horizon, subsampled at 10 Hz) and the 5-point predictions. Similar to a `mean` function.

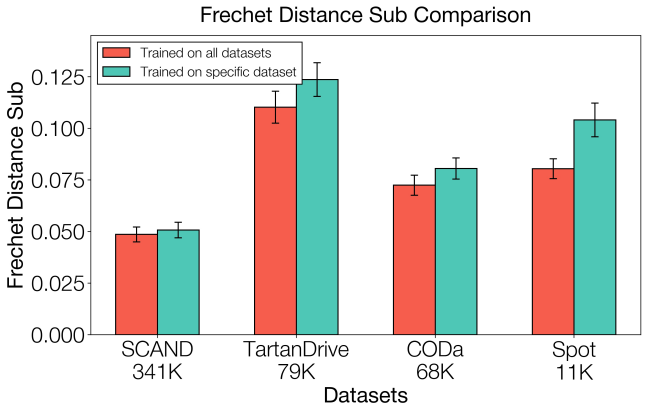
Additionally, we run statistical significance tests for all metrics on a per-dataset basis. Across these four datasets, the generalist model consistently yields small statistically significant improvements in trajectory accuracy. In Tartan-Drive, mean L2 and sub-sampled Fréchet distances improve at $p < 0.05$, while full-trajectory Fréchet and normalized DTW show highly significant gains (***). CODA exhibits highly significant reductions ($p < 10^{-5}$) in mean and max L2, sub-sampled Fréchet, and sub-sampled DTW, with endpoint and full-trajectory Fréchet errors remaining comparable. In SCAND, the general model outperforms on mean L2 (*), max L2 (**), sub-sampled Fréchet (**), and full-trajectory DTW (*), with other metrics not significant. On Spot, nearly all metrics except endpoint error and full-Fréchet errors achieve *** significance. Overall, these results suggest the general model produces smoother, more accurate paths across varied environments, even when terminal or peak deviations remain similar.



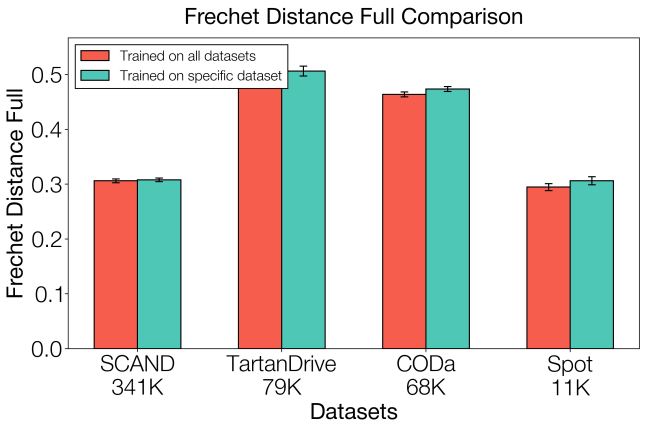
(a) Max L2 Error



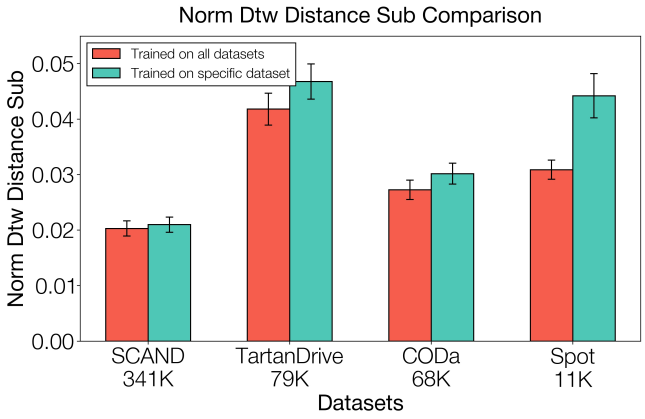
(b) Last Point Error



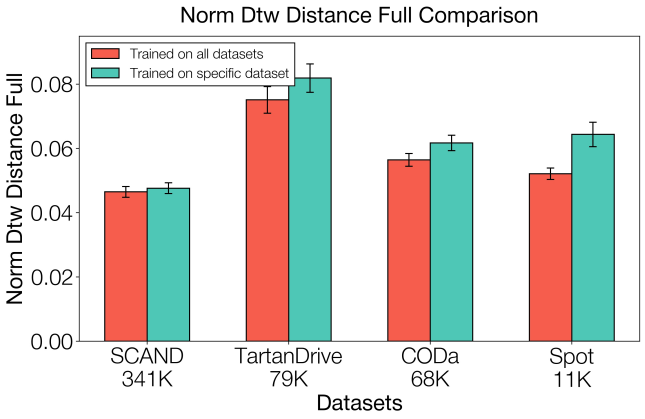
(c) Fréchet Distance on Subsampled Trajectories



(d) Fréchet Distance on Full Trajectories

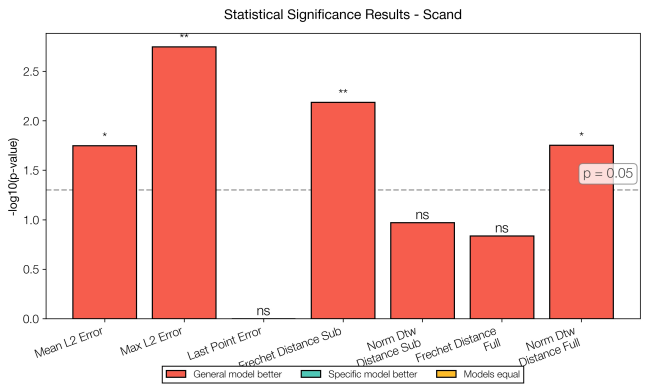


(e) DTW Distance on Subsampled Trajectories

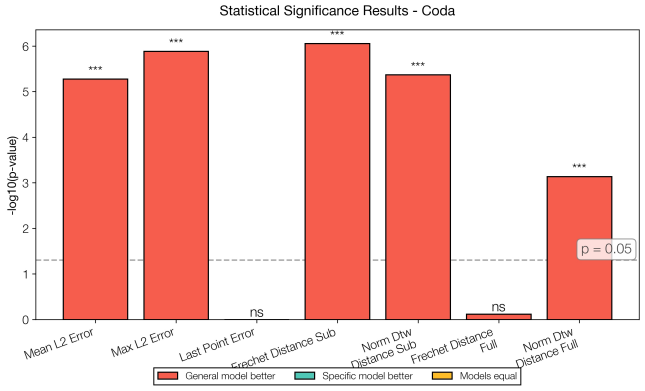


(f) DTW Distance on Full Trajectories

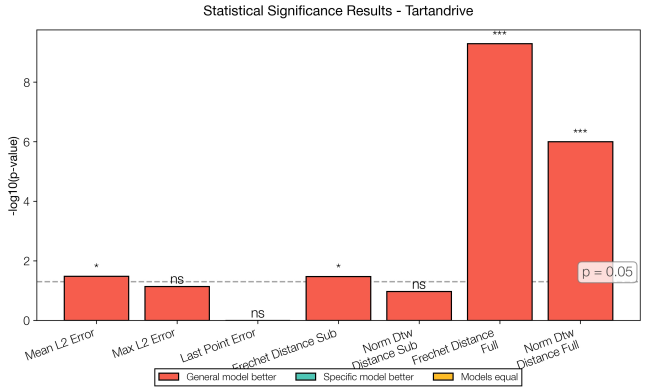
Fig. 16: Offline metrics comparing high-level generalist vs robot-specific navigators. In all metrics, we benefit from training on the pooled data compared to robot-specific datasets.



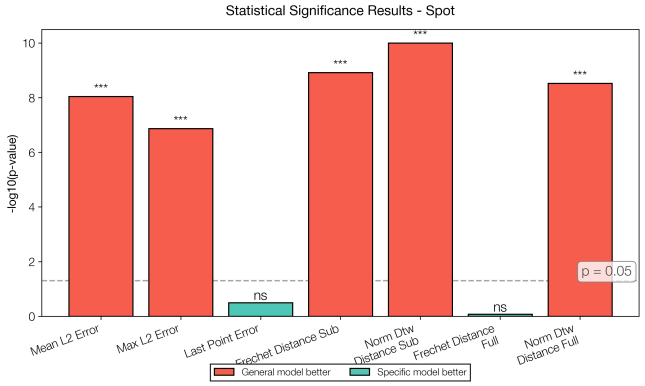
(a) SCAND



(b) CODa



(c) TartanDrive



(d) Spot

Fig. 17: Statistical significance of paired t-tests comparing generalist and robot-specific models on four datasets (Tartandrive, CODA, SCAND, Spot) across seven trajectory-error metrics. Bars show $-\log_{10}(\text{p-value})$; green indicates the general model is better, orange the specific model, and gray cases where means are equal. '*' means $p < 0.05$, '**' means $p < 0.01$, '***' means $p < 0.001$, and "ns" means $p \geq 0.05$; the dashed line marks $p = 0.05$ ($-\log_{10}(0.05) \simeq 1.3$).