

OWLS: Open-world Language-conditioned Speed Fields for Zero-Shot Rover Navigation

Chahyon Ku¹, Erica Tevere², Georgios Georgakis², Rudra Mukherjee², and Bernadette Bucher¹

Abstract—Semantics can provide a proxy for physical traversability. Terrain that looks like loose sand or a rock field is likely to have similar physical properties. Open-vocabulary vision-language models (VLMs) enable this physical reasoning over terrain outside any fixed label set. We present Open-world Language-conditioned Speed Fields (OWLS), which plans directly over the dense feature map of a VLM. Prompted with a description of traversable terrain, the VLM scores every pixel of a single camera image; OWLS projects these scores onto the ground, adapts them to the platform with two morphological operations, and uses the result as the speed field of a minimum-time planner, so terrain appearance and obstacle clearance shape one trajectory. OWLS needs no task-specific training, no terrain annotation, no persistent map, and no exteroceptive sensor beyond the camera. We evaluate it on a physical rover indoors and off-road, on simulated Mars and Moon rovers over 200 procedurally generated terrains, and on real planetary imagery.

I. INTRODUCTION

Semantics can provide a proxy for physical traversability. Terrain that looks like loose sand or a rock field is likely to have similar physical properties. Whether a robot can cross a stretch of terrain depends on information such as the size of obstacles relative to its footprint and clearance, load bearing, and slip, which a camera cannot measure directly. Planetary rovers must reason about these physical properties under severe conditions. Rovers drive without GPS, and communication delays force long stretches of onboard autonomy [1].

Existing systems model physical traversability in several ways. Hand-designed geometric heuristics, such as step-height or slope limits, require substantial per-platform engineering [2]. Learned terrain models [3]–[5] predict a fixed set of classes from annotated data, and each class still needs a hand-assigned planning cost, so neither the classes nor the costs cover terrain the dataset did not anticipate. Self-supervised estimators learn traversability from the robot’s own driving [6], [7], which presumes experience on the target terrain. Open-vocabulary vision-language models (VLMs) remove the fixed label set, but navigation systems built on them take free space from depth or LiDAR and use the VLM to select a goal, rank candidate paths, or annotate that geometric map [8]–[12]. None of these existing approaches

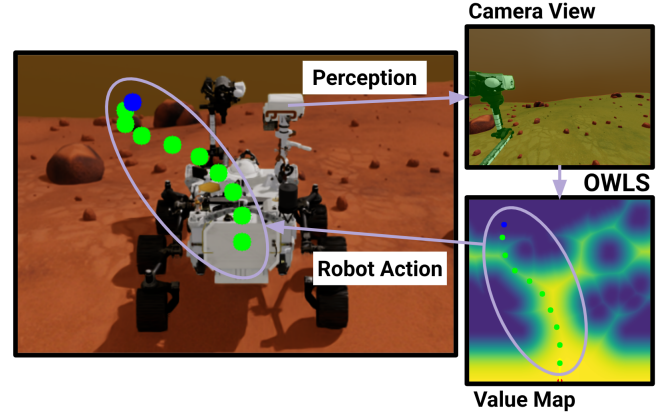


Fig. 1: OWLS plans directly over the dense feature map of a VLM: the VLM scores a single camera image against a description of traversable terrain (green), and the scores become the speed field of a minimum-time planner whose waypoints (green) the rover follows to the goal (blue).

plan over the VLM’s dense output itself, although that output already scores every pixel against a description of drivable terrain.

We present Open-world Language-conditioned Speed Fields (OWLS), a local navigation method that plans directly over the dense feature map of an open-vocabulary VLM (Fig. 1). Prompted with a description of traversable terrain, the VLM scores each pixel of the camera image, and OWLS turns this score map into a trajectory in three steps. Ground projection places each score at a metric position in the robot frame. Two morphological operations then encode the platform, since a semantic score does not say whether a particular robot can drive over a rock or how far it must keep from one: a dilation removes low-scoring regions small enough to drive over, and an erosion surrounds the rest with a clearance buffer scaled to the vehicle. Finally, rather than thresholding the result into occupancy, OWLS uses it as the speed at which the robot moves through each cell and plans the minimum-time trajectory to the goal, which a waypoint controller tracks. Obstacle clearance and the VLM’s judgment of the terrain thus enter a single optimization. OWLS uses the VLM without fine-tuning and needs no terrain annotation, and adapting to a new rover needs at most two radii set from the rover’s dimensions. It keeps no persistent map, replanning from each new image with a single camera and wheel odometry.

Contributions. (i) We show how to **plan directly over the dense feature map of an open-vocabulary VLM**,

A portion of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

¹C. Ku and B. Bucher are with the Department of Robotics, University of Michigan, Ann Arbor, MI 48109, USA {chahyon, bucherb}@umich.edu

²E. Tevere, G. Georgakis, and R. Mukherjee are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA

using ground-projected VLM scores as the speed field of a minimum-time planner, which yields local navigation over previously unseen terrain from a single camera with no task-specific training and no persistent map. (ii) We introduce an **interpretable embodiment adaptation** of the VLM map by two morphological operations with radii set from rover geometry, which carries the same VLM and prompt across three rovers of distinct size and camera placement without retraining. (iii) We **evaluate from terrestrial to planetary settings**: on a physical rover, on a simulated rover on 200 procedurally generated Mars and Moon terrains with prompt and morphology ablations, and qualitatively on real planetary imagery.

II. RELATED WORK

Traversability from Appearance. Traversability is a physical property, and existing methods differ in how they reach it. Geometric methods measure it from range data: Perseverance checks stereo-derived terrain against the vehicle’s clearance and tilt limits [1], and STEP [2] evaluates risk from LiDAR geometry, both with substantial per-platform engineering. Semantic methods use appearance as a proxy for the same properties and learn the mapping from appearance to cost. The common instance trains terrain classifiers over a fixed label set on annotated data [3]–[5] and assigns each class a planning cost; annotation is expensive, per-class costs are hard to design, and the closed class set is often ambiguous or application-specific. Other methods ground the proxy in the robot’s own experience, through vibration-based ground classification [13], wheel-slip distributions with uncertainty for risk-aware planning [14], traversability learned from the robot’s driving over pretrained visual foundation features [6], [7], or a segmentation foundation model distilled with egocentric video [15]. These remove manual labels but still train a platform-specific estimator. OWLS takes the appearance proxy from a pretrained VLM’s response to a text description, with no training, and moves the platform dependence out of the estimator into two morphological radii set from rover geometry.

Mapless Navigation and Planning over Speed Fields. Learned policies act on RGB images without a globally consistent map, trained by imitation [16] or offline reinforcement learning [17]; GNM [18], ViNT [19], and NoMaD [20] train on large cross-platform corpora of human-collected navigation data for image-goal-conditioned navigation and unconditioned exploration. They model traversability only implicitly, are bounded by the diversity of their data, and offer no way to state which terrain counts as drivable. Classical visual servoing also closes the loop on each new image without a map [21], [22], but on hand-engineered geometric features [23], [24]. OWLS instead keeps an explicit, interpretable planner. Eikonal planners compute minimum-time paths under a speed field with fast marching [25] or the Fast Iterative Method [26], and mobile robot planners derive that speed from occupancy or distance to obstacles [27]. OWLS derives it from a VLM feature map, so terrain appearance,

not only geometry, sets how fast the robot can move through each cell.

VLMs for Navigation. Open-vocabulary VLMs supply the appearance proxy without a fixed label set, but navigation systems built on them take free space from other sensors. VLMaps [8] fuses open-vocabulary features into a metric map built from depth and uses language to pick goals; it also derives per-robot obstacle maps by listing obstacle categories in text, a closed list per embodiment. VLFM [9] builds a language-grounded value map from a pretrained VLM, as OWLS does, but uses it to rank the frontiers of an occupancy map built from depth. VLMNav [10] takes traversable regions from perfect simulated depth, samples candidate paths, and asks a VLM to select one. BehAV [11] and Open-Nav [12] plan on occupancy grids from LiDAR and use VLM masks to set a goal or encode operator preferences (e.g. “avoid grass” although grass is traversable), BehAV requiring the operator to enumerate object types and their desirability per scenario. VAMOS [28] trains a planner on diverse data to propose paths in image space and re-ranks them with a per-embodiment affordance model learned in simulation. In each system the VLM selects, annotates, or proposes, while what the robot can traverse comes from depth, LiDAR, or a learned model. OWLS plans over the VLM’s dense output itself: from a single calibrated camera and with no training, the ground-projected feature map, adapted by two radii, is the speed field of the planner.

III. METHOD

OWLS plans a local trajectory directly over the dense feature map of an open-vocabulary VLM and tracks it with a waypoint controller, replanning from each new image without a persistent map (Fig. 2). Planning (Sec. III-A) runs at the perception rate: the VLM scores each pixel against a text prompt, ground projection lifts the scores into a robot-frame value map V , two morphological operations adapt V to the platform, and the adapted map sets the speed field of a minimum-time planner. Tracking (Sec. III-B) runs at the control rate and steers the robot along the planned waypoints.

A. Planning over VLM Features

Language-conditioned Traversability. We prompt an open-vocabulary VLM with a description of traversable terrain and read out its dense, pixel-aligned response $H \in [0, 1]^{h \times w}$, which we call its feature map. For open-vocabulary segmenters such as Grounded SAM 2 [29]–[31] and SAM 3 [32], H is the pixel-wise mask probability before thresholding. We keep the continuous response rather than a binary mask, so the planner can weigh how strongly each pixel matches the prompt. Every simulated experiment uses the single prompt “ground”, held fixed across platforms, VLMs, environments, and episodes, so that per-platform adaptation is carried by the morphological radii below; the hardware trials of Sec. IV-A use “drivable ground”.

Ground Projection. Clearance and travel time are metric quantities in the robot frame, so we lift H onto the ground to form a top-down value map V . We assume flat ground, so

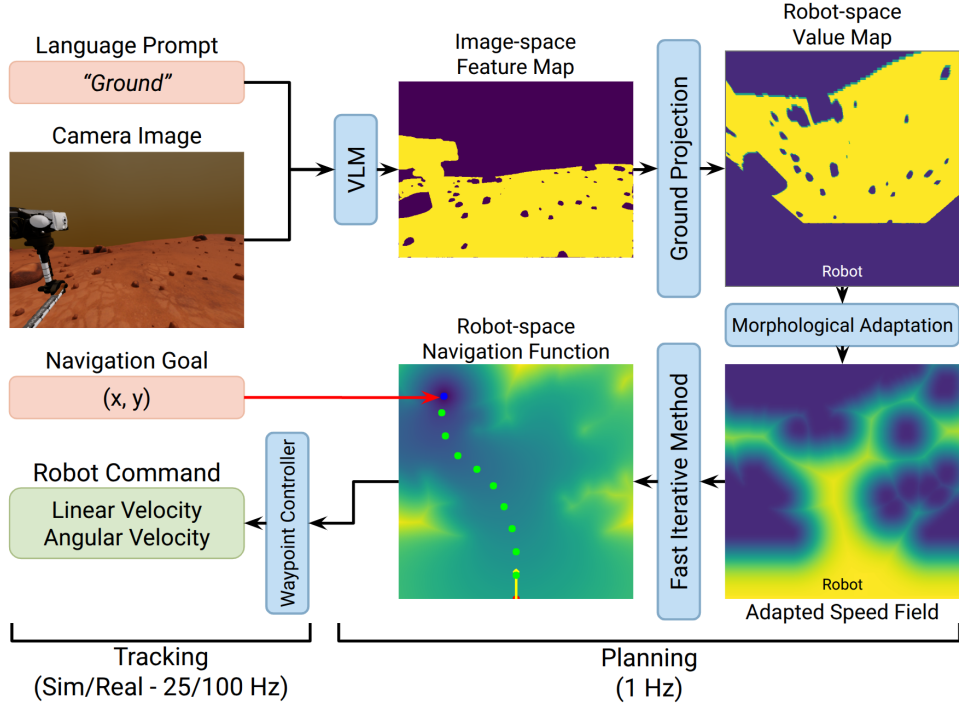


Fig. 2: OWLS plans directly over the dense feature map of a VLM (Sec. III-A). The VLM scores each pixel against the prompt; the scores are ground-projected into the robot frame, adapted to the platform by dilation and erosion, and used as the speed field of a minimum-time planner, whose trajectory a waypoint controller tracks (Sec. III-B).

that terrain within the local window lies on the plane $z = 0$ of the robot frame, and, with intrinsics $\mathbf{K}$ and calibrated robot-to-camera extrinsics, a ground point projects through the planar homography $\mathbf{M}$ obtained by dropping the third column of the extrinsic matrix. Each cell of a regular grid $\Omega \subset \mathbb{R}^2$ takes the value of H at its projection through $\mathbf{M}$, or 0 if it projects outside the image or above the horizon, giving $V : \Omega \rightarrow [0, 1]$.

Morphological Adaptation. The VLM scores how well terrain matches the prompt, not whether a given robot can traverse it, so we adapt V to the platform’s size and capability with two grayscale morphological operations. A *dilation* with a disk of radius ρ_t erases low-traversability regions, including VLM artifacts, smaller than the largest obstacle the platform can drive over. An *erosion* with a Gaussian-tapered disk of radius ρ_r then grows the remaining ones into a clearance buffer whose penalty decays with distance, giving V' . Neither operation thresholds the map, so traversability stays continuous. Both radii follow from rover geometry (detailed later in Sec. III-C): ρ_r is set per platform from its track width and ρ_t is shared across platforms.

Speed Field and Minimum-time Trajectory. Because V' is continuous, we plan over it directly instead of thresholding it into occupancy. We take $c = sV' + \epsilon$ as the speed at which the robot moves through each cell, generalizing the binary navigation function, in which free cells have uniform speed and obstacles have zero speed. The floor $\epsilon > 0$ keeps every cell passable, so the robot plans through low-traversability terrain only where going around takes longer, and unobserved cells ($V = 0$) move at the same floor as

confident obstacles. The navigation function N is the travel time from each cell to the goal under c , the solution of the eikonal equation $\|\nabla N\| = 1/c$, computed with the Fast Iterative Method [26]. It is seeded at the goal cell when the goal lies in Ω and otherwise at the boundary cell along the goal bearing, clipped to a forward cone; the seed is recomputed each cycle, so the effective goal recedes as new terrain enters view. The trajectory $\mathbf{f} = ((0, 0), (x_1, y_1), \dots)$, in the robot frame at image capture, descends the navigation function N from the robot: from each point we take the step within a fixed radius that most reduces the time to goal. It is the minimum-time path under the VLM-derived speed field, so the clearance buffers shape it around obstacles and the VLM scores steer it across terrain, in one optimization. We use a minimal controller (Eq. (2)) to focus on the contribution of our speed field, which suffices for low-speed tracking in our simulated and real-rover experiments, but we can swap it for more advanced control algorithms in deployment scenarios requiring more complex dynamics.

B. Trajectory Tracking

A new trajectory arrives with each image; between images, wheel odometry transforms it into the robot’s current frame. The controller steers toward the first waypoint $\mathbf{p}^* = (x_i, y_i)$ farther than a lookahead distance $d_{\min}$, at bearing $e_{\text{ang}} = \arctan2(y_i, x_i)$. With an alignment threshold α and range r_g to the commanded goal, it switches between turning in place and driving straight,

$$w^* = w_{\max} \text{sign}(e_{\text{ang}}) \mathbf{1}\{|e_{\text{ang}}| \geq \alpha\} \mathbf{1}\{r_g \geq d_{\text{goal}}\}, \quad (1)$$

$$v^* = v_{\max} \mathbf{1}\{|e_{\text{ang}}| < \alpha\} \mathbf{1}\{r_g \geq d_{\text{goal}}\}, \quad (2)$$

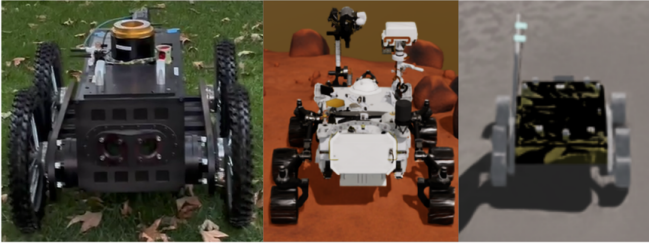


Fig. 3: Real 4-wheeled rover (left), simulated Perseverance (middle), and simulated Pragyan (right) used in the experiments.

and halts within d_{goal} of the goal. On a mobile RTX 4090, a planning cycle with GSAM2 takes 692 ms, 490 ms of it in the VLM and 199 ms in the morphology, consistent with the 1 Hz rate; waypoint and action generation are sub-millisecond.

C. Implementation Details

Blind and Self-occluded Cells. Two kinds of cells show no terrain without being obstacles: the blind strip between the robot and the nearest visible ground, and cells hidden behind fixed vehicle parts in view, such as Perseverance’s stowed arm. Both are set to maximum traversability instead of $V = 0$, so they do not read as obstacles.

Parameters. The grid Ω is 640×640 cells at 0.0125 m, spanning 8 m ahead of the robot and ± 4 m laterally. The speed field uses $\epsilon = 0.1$ and a VLM-dependent gain, $s = 8$ for Grounded SAM 2 and $s = 16$ for SAM 3, which corrects for the different ranges of the two models’ raw mask probabilities H . The morphological radii are $\rho_r = 2.15$ m for Perseverance and 0.50 m for Pragyan and the terrestrial rover, with $\rho_t = 0.19$ m throughout. The controller uses a lookahead d_{min} and stopping radius d_{goal} both of 0.25 m, and $\alpha = 8^\circ$ on all platforms, with linear and angular speeds $(v_{\text{max}}, w_{\text{max}})$ of (0.70, 2.06), (0.35, 2.19), and (0.86, 2.42) in m/s and rad/s for Perseverance, Pragyan, and the terrestrial rover. Planning runs at 1 Hz on all platforms, and control at 25 Hz in simulation and 100 Hz on the physical rover.

IV. EXPERIMENTS

Lacking access to a real space rover, we characterize monocular planetary navigation in four steps: real-rover trials (Sec. IV-A) establish feasibility under real compute and sensor constraints; a simulated sweep over 100 terrains per platform (Sec. IV-B) tests transfer to space embodiments and appearance; ablations over the same terrains (Sec. IV-C) vary the prompt and the morphology, the only inputs set per deployment; and inference on real space-rover imagery (Sec. IV-D) tests the VLM under flight lighting and sensor noise.

A. Physical Rover Experiments on Earth

To test our method under real compute and sensor constraints, we run it on a 4-wheeled rover in two Earth scenarios: an indoor hallway and an off-road field mixing grass, dirt, bushes, trees, and pavement.

Setup. The rover carries a single front-facing RGB pinhole camera at 640×480 , and our method runs on a laptop

Environment	Obstacles	GSAM2	DINOV3 patch	DINOV3 lang
Indoor	Static	100	40	0
Indoor	Dynamic	80	20	0
Outdoor	Static	100	40	0
Outdoor	Dynamic	40	0	0

TABLE I: Navigation success rate (%) of a real 4-wheeled rover in two Earth scenarios with dynamic / static obstacles over 5 randomized trials per configuration (Sec. IV-A).

with a mobile NVIDIA RTX 4090, generating a trajectory from the front-view image and publishing closed-loop wheel velocity commands at the rates of Sec. III-C. Wheel encoders provide odometry at the control rate, closing the velocity loop and carrying the trajectory forward between perception updates (Sec. III-B). All trials ran the prompt “drivable ground”. Because the planner accepts any dense, pixel-aligned VLM output, we compare three feature maps to plan over: **Grounded SAM 2** (GSAM2) [29]–[31], **DINOV3 patch embeddings** (DINOV3-patch) [33], and **DINOV3 language embeddings** (DINOV3-lang) [33], [34]. For both DINOV3 variants the feature map is the similarity of patch features to a query embedding: DINOV3-patch queries with the image patch ahead of the rover and takes no text prompt, serving as a language-free control; DINOV3-lang queries with a text embedding, applying the DINOV2-txt [34] recipe to DINOV3, as released.

Metric. We report navigation success rate over 5 randomized trials per configuration, a trial succeeding if the robot traverses the environment end to end without collision. This count establishes feasibility rather than fine-grained comparison, so we treat differences of one or two trials as indicative only; it measures how consistently the method follows traversable area up to a local dead-end.

Is OWLS capable of driving a real mobile robot in various environments? With GSAM2, OWLS avoided all static obstacles indoors and outdoors in all 5 trials (Table I), and dynamic ones such as humans and boxes indoors 80% of the time (Fig. 4); in dynamic outdoor scenes only 2 of 5 succeeded. We attribute these to under-segmentation in unconstrained outdoor lighting plus phase lag from the two-rate architecture. DINOV3-patch succeeded less often (Table I): its smoother feature map made it less reactive, so it avoided only obstacles that small steering corrections could clear. DINOV3-lang traversed nothing: on our trial footage its text query spans roughly a seventh of the patch query’s raw similarity range, which per-frame renormalization then amplifies.

B. Simulated Rover Experiments on Mars and Moon

To verify whether performance on a 4-wheeled Earth rover transfers to space rovers (different embodiment) in space environments (different appearance and obstacles), we evaluate in the Space Robotics Bench (SRB) [35].

Setup. SRB provides robot definitions and procedurally generated Mars and Moon terrain over IsaacLab [36] and the IsaacSim [37] renderer. We use the provided Perseverance and Pragyan definitions (Fig. 3) and terrains as is, adding

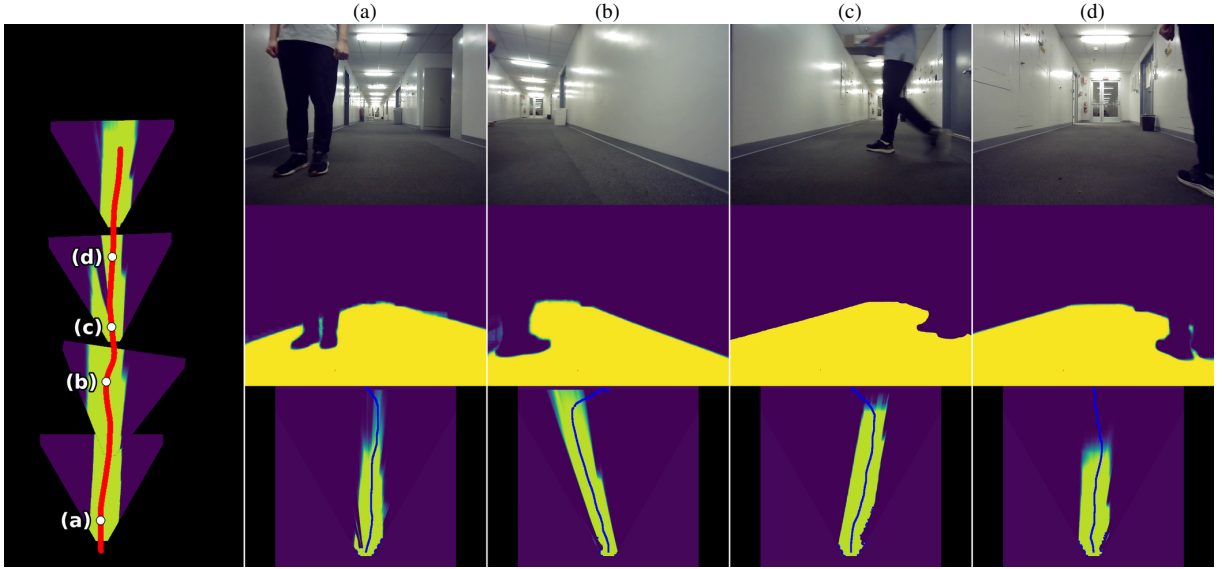


Fig. 4: Physical rover navigating an indoor hallway while a person walks across its path (Sec. IV-A). Left: executed trajectory (red) over the top-down traversability of successive perception cycles. (a–d) The four timesteps, each with the camera view (top), the VLM’s image-space heatmap (middle), and the top-down speed field with the planned trajectory (blue, bottom). The heatmap excludes each person as they cross, and since OWLS keeps no map, each cycle replans over the ground left unoccupied at that moment.

a stationary target 10 m away at a randomly drawn bearing. Only Perseverance needs the self-occlusion mask of Sec. III-C, for its stowed arm. We run OWLS with two VLMs: **GSAM2** [29]–[31], as in the real-rover study, and the more recent **SAM 3** [32]. They share every geometric, morphological, and control parameter; only the speed gain is set per VLM (Sec. III-C), so the comparison isolates the VLM with its calibration. We do not carry the DINOv3 variants into simulation, since DINOv3-patch proved insufficiently reactive and DINOv3-lang failed outright during the hardware experiments. All simulated runs use the prompt “ground” and the radii of Sec. III-C.

Baseline. We compare against NoMaD [20], a navigation foundation model trained on “over 100 hours” of heterogeneous navigation data from varied robots and environments [18], [38]. Where OWLS plans one trajectory over the VLM feature map, NoMaD samples many unconditioned trajectories; following the same conventions used in VAMOS [28], we take the one whose waypoint is closest to the goal. Neither method is fine-tuned in domain, as both are meant to run out of the box. NoMaD has no interface for a metric goal or platform calibration, so its network runs unconditioned, the goal entering only through the selection above, where OWLS receives the goal, calibration, odometry, and its two morphological radii. We read it as a reference point for a learned navigation prior rather than a matched baseline.

Metric. We report *navigation success rate*: success is holding within 0.25 m of the target for one second inside a 60 s episode, and a no-progress guard ends an episode early if the rover gains less than 0.1 m on the goal over a 20 s window, both applied identically to all methods. The criterion scores goal-reaching alone and carries no collision term, so contact with a rock is penalized only

Method	Success Rate (%)	STE
<i>Perseverance (Mars)</i>		
NoMaD [20]	23.0	0.119
OWLS + GSAM2 (ours)	58.0	0.319
OWLS + SAM 3 (ours)	60.0	0.319
<i>Pragyan (Moon)</i>		
NoMaD [20]	35.0	0.217
OWLS + GSAM2 (ours)	75.0	0.561
OWLS + SAM 3 (ours)	77.0	0.597

TABLE II: Simulated space experiments (Sec. IV-B) over the same 100 terrains per platform. On terrains both solve, NoMaD takes 30.4 vs. 27.3 s against OWLS with GSAM2 on Perseverance and 46.4 vs. 38.6 s on Pragyan.

indirectly, on the episodes where it stalls the rover into the no-progress guard. Mean completion time over successes conditions each method on a different terrain subset, so for efficiency we use success-weighted time efficiency, $STE = \frac{1}{N} \sum_i S_i t_{\text{opt}} / \max(t_i, t_{\text{opt}})$, with S_i success on terrain i , t_i its time, and $t_{\text{opt}} = 10 \text{ m} / v_{\text{max}}$ (14.3 and 28.6 s for Perseverance and Pragyan). Averaging over all N terrains keeps STE comparable across methods, in the spirit of success weighted by path length (SPL) [39]. SRB fixes its procedural terrain per process, so to sample the obstacle distribution we launch one process per terrain, 100 per platform with one episode each, with the start pose and goal direction fixed across terrains and conditions so that terrain is the only quantity that varies. Every condition runs on the same 100 terrains.

Is OWLS capable of driving a space rover in a space environment? OWLS reaches the goal on the majority of unseen terrains on both platforms and with both VLMs: 58.0 and 60.0 on Perseverance, 75.0 and 77.0 on Pragyan (Table II). Fig. 5(a, b) shows planning over the VLM feature

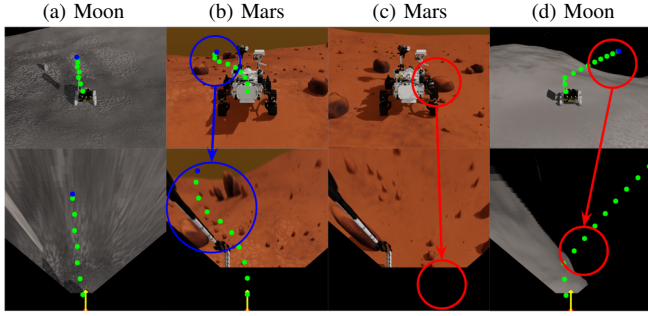


Fig. 5: Simulation results: third-person view (top, visualization only) and top-down RGB (bottom, visualization only) with waypoints (green) generated from goal (blue). Successes: (a) a valid path under out-of-distribution lunar appearance, (b) threading two rocks with the erosion clearance. Failures: (c) an obstacle forgotten after leaving the field of view, (d) a slope read as a cliff by the flat-ground projection.

map handling both open-world appearance and embodiment: on the Moon it plans a valid path over terrain whose appearance lies far outside the VLM’s training distribution, and on Mars it threads between two rocks with the clearance the erosion buffer adds. NoMaD reaches 23.0 and 35.0 on the same terrains, behind every OWLS variant and slower on the terrains both solve (Table II). OWLS fails by stalling, where the no-progress guard fires, more often than by timing out (36 against 6 of 42 Perseverance failures with GSAM2, 16 against 9 of 25 on Pragyan). NoMaD’s failures also end in the no-progress guard (77 of 77 on Perseverance, 59 of 65 on Pragyan): its trajectory samples are not conditioned on the goal and rarely cover the traversable region toward it, so the selected waypoint makes no progress.

How much does the choice of VLM matter? Under the deployed prompt it makes no measurable difference: the variants land within two points of each other on both platforms (58.0 vs. 60.0 and 75.0 vs. 77.0), despite differing in architecture and training data. Under other wordings it does: replacing “ground” by “area” or “region” takes the same SAM 3 pipeline to 0.0 on both platforms (Table III), which Sec. IV-C takes up.

C. Prompt and Morphology Ablations

The prompt and the two morphological operations are the only inputs an operator sets per deployment, so we vary each over the terrain set of Sec. IV-B with everything else held fixed (Tables III and IV).

Setup. The prompt rows replace the single word “ground” by five other nouns for free space, or by “rocks” to name the hazard instead; Table IV scans each morphological radius under “ground” on Perseverance with GSAM2, the other radius held at its deployed value.

Metric. Alongside success rate over the same 100 terrains we log two diagnostics per perception cycle: the maximum VLM score, which is near zero when the VLM cannot ground the prompt, and the fraction of the image the VLM scores above zero, the traversable fraction. The “ground” row repeats Table II and is the reference for every other row.

Condition	Perseverance		Pragyan	
	GSAM2	SAM 3	GSAM2	SAM 3
<i>Free-space noun</i>				
ground (deployed)	58.0	60.0	75.0	77.0
area	58.0	0.0	74.0	0.0
region	54.0	0.0	83.0	0.0
path	60.0	15.0	73.0	66.0
terrain	60.0	56.0	71.0	83.0
surface	59.0	59.0	73.0	90.0
range over nouns	54–60	0–60	71–83	0–90
<i>Hazard noun</i>				
rocks	10.0	34.0	71.0	10.0

TABLE III: Prompt ablations (Sec. IV-C): success rate (%) over the 100 terrains of Table II.

Does the prompt determine the trajectory? It does, except on lunar terrain sparse enough that the VLM grounds the hazard noun onto the surface itself. Because OWLS plans over the VLM’s response to the prompt, prompting for “rocks” instead of “ground” marks hazards as traversable and drops success to 10.0 (GSAM2) and 34.0 (SAM 3) on Mars and to 10.0 for SAM 3 on the Moon. The exception is GSAM2 on the Moon, 75.0 $\rightarrow$ 71.0: with no rock in view, which is most lunar frames, GSAM2 grounds “rocks” onto the regolith and its feature map barely changes.

Does the wording of the prompt matter? For GSAM2 it does not; for SAM 3 it decides the outcome. Replacing “ground” by another single free-space noun leaves GSAM2 within four points of its deployed value on Perseverance and eight on Pragyan. The same replacement takes SAM 3 to 0.0 on both platforms under “area” and “region”. The diagnostics identify the cause: under those two words SAM 3 returns an empty mask, maximum score zero, on 99 and 100 % of Perseverance cycles (97 and 93 % on Pragyan) and proposes no waypoint, whereas GSAM2 scores all six nouns above 0.97 with a traversable fraction between 0.43 and 0.64. Success falls as the empty-mask rate rises: “path” returns an empty mask on 41 % of Perseverance cycles and 21 % of Pragyan ones, and scores 15.0 and 66.0. Nor does a higher score identify a better wording: SAM 3 under “surface” reaches 90.0 on Pragyan, above every deployed configuration, with a median peak score of 0.54 against about 0.1 for the other nouns, but it does so by calling over 97 % of the frame traversable, and the same wording leaves Perseverance at 59.0; we keep “ground”, fixed before this evaluation. Language is therefore a working but brittle specification interface: a wording must be validated against the VLM before deployment.

Are the geometry-set radii effective? They are: the radii derived from rover geometry (Sec. III-A) match or beat every other value scanned in Table IV, so adapting to a new platform takes geometric measurements rather than parameter tuning. Erosion success peaks at the deployed 2.15 m and falls on either side, by 7 points with erosion off and 6 at 3.0 m. Dilation is flat between off and the deployed 0.19 m, and a 1.0 m radius costs 13 points, as it erases obstacles the rover cannot clear. In summary, success falls or plateaus as either radius moves away from its deployed

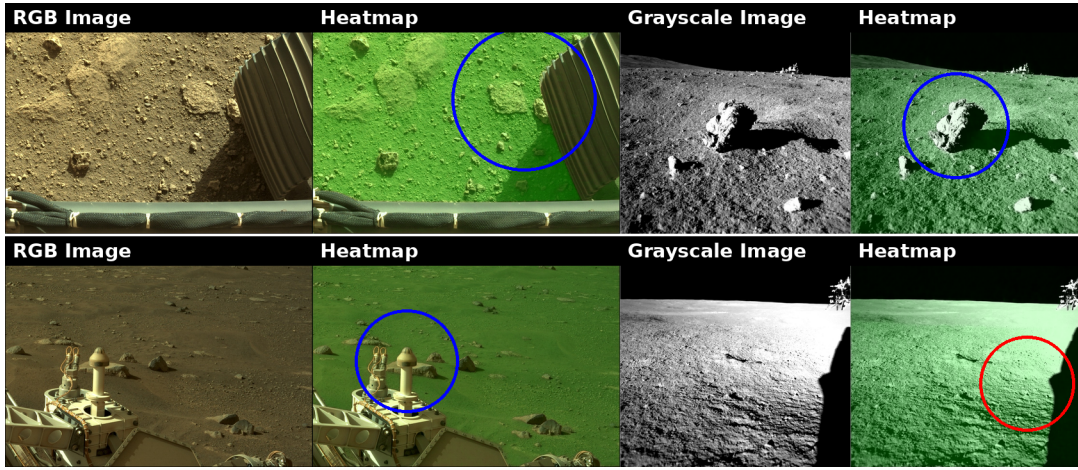


Fig. 6: Qualitative results on real planetary rover imagery (Sec. IV-D). Left: Mars [40] Right: Moon [41]. Each pair shows the input and the VLM’s image-space traversability heatmap (green, brighter is more traversable). Blue circles mark hazards correctly excluded from ground; the red circle marks a shadowed region misclassified as obstacle under extreme lighting.

Radius	Success (%)
<i>Erosion</i> ρ_r	
off	51.0
0.50 m	54.0
2.15 m (deployed)	58.0
3.0 m	52.0
<i>Dilation</i> ρ_t	
off	59.0
0.19 m (deployed)	58.0
1.0 m	45.0

TABLE IV: Morphology scan (Sec. IV-C): GSAM2 success (%) on Perseverance under ground over the terrains of Table III, one radius scanned with the other at its deployed value ($\rho_r = 2.15$ m, $\rho_t = 0.19$ m); bold is best per group.

value; as the metric scores no contact, a plateau bounds goal-reaching, not clearance.

D. Ground Detection on Images from Real Space Rovers

The VLM feature map is the most failure-sensitive input to the planner, and our modular design lets it be tested on flight imagery without deploying end to end.

Setup. We run SAM 3 [32], without fine-tuning, under the deployed “ground” wording on frames from the real Perseverance and Pragyan rovers. The Mars frames come from Perseverance [40] and the monochrome Moon frames from the Chandrayaan-3 Pragyan navigation camera [41]; none were used to develop or tune the method.

Metric. Labeled flight imagery is too scarce for a quantitative score, so the evidence is visual inspection of the feature maps (Fig. 6).

Is the VLM capable of adapting to sensor noise and extreme lighting from real space rovers? Partly: the VLM separates ground from hazards across the domain shift, but misclassifies under extreme lighting (Fig. 6); the rocks, robots, and slopes appear as low-scoring regions in the complement of the ground query, not as named concepts. We hypothesize that extreme lighting is scarce in these models’ training data, and a lightweight visibility estimator could mitigate it.

V. CONCLUSION

Summary. We presented OWLS, which plans local navigation directly over the dense feature map of an open-vocabulary VLM: the VLM’s per-pixel response to a text prompt is ground-projected into an egocentric value map, adapted to the platform by two morphological operations, and used as the speed field of a minimum-time planner whose trajectory a waypoint controller tracks, with no task-specific training, terrain annotation, or persistent map. Real-rover trials establish feasibility. In simulation OWLS reaches the goal on most of 100 unseen terrains per platform with either of two VLMs, ahead of a learned navigation foundation model in all four settings, and naming the hazard instead of free space cuts success in three of four, so the trajectory follows the VLM’s language-conditioned output, though rewording the prompt alone can zero it out. Real planetary imagery shows the VLM transferring across the domain gap while remaining sensitive to extreme lighting.

Limitations and Future Work. We discuss five limitations in our method and evaluation. First, *the memory-less design* forgets any obstacle leaving the field of view, the most common failure we observe (Fig. 5(c)), which additional cameras or a bounded local memory could mitigate without sacrificing the mapless property. Second, the method has *no 3D understanding*: the flat-ground assumption behind ground projection (Sec. III-A) permits planning from one RGB image without depth sensing, but projects inaccurately on steep slopes (Fig. 5(d)) and likely fails to model overhead clearance and tall obstacles unless augmented with monocular depth or stereo, which we leave for future work. Third, our method runs an *off-the-shelf VLM* on a compute platform beyond what flight hardware offers today, though a radiation-hardened multicore processor is in development [42] and a commercial smartphone SoC has already flown on Ingenuity [43], [44]. Fourth, *language can be a brittle specification interface*: swapping one free-space noun for another can take SAM 3 to zero success (Sec. IV-C), and current VLMs perform best with prompts representing appearance rather

than the physical properties that decide traversal. Finally, we tested the feature map on flight imagery (Sec. IV-D) but never closed the loop under *planetary optical effects* in our simulated experiments due to limited modeling of such effects (e.g. the opposition effect washing out texture when the camera faces away from the sun, and the low sun near the lunar poles casting sharp and long shadows).

REFERENCES

- [1] O. Toupet, M. Ono, T. del Sesto, M. Maimone, and M. McHenry, “Enhanced autonomous navigation on the Perseverance Mars rover,” *IEEE Transactions on Field Robotics (T-FR)*, vol. 3, pp. 83–124, 2026.
- [2] A. Dixit, A.-a. Agha-mohammadi, D. D. Fan, J. W. Burdick, Y. Kubo, and K. Otsu, “STEP: Stochastic traversability evaluation and planning for risk-aware off-road navigation,” in *Robotics: Science and Systems (RSS)*, 2021.
- [3] A. Shaban, X. Meng, J. Lee, B. Boots, and D. Fox, “Semantic terrain classification for off-road autonomous driving,” in *Conference on Robot Learning (CoRL)*, 2022.
- [4] X. Meng, *et al.*, “TerrainNet: Visual modeling of complex terrain for high-speed, off-road navigation,” in *Robotics: Science and Systems (RSS)*, 2023.
- [5] A. Li, C. Yang, J. Frey, J. Lee, C. Cadena, and M. Hutter, “Seeing through the grass: Semantic pointcloud filter for support surface learning,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 11, pp. 7687–7694, 2023.
- [6] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, “Fast traversability estimation for wild visual navigation,” in *Robotics: Science and Systems (RSS)*, 2023.
- [7] S. Triest, M. Sivaprakasam, S. Aich, D. Fan, W. Wang, and S. Scherer, “Velociraptor: Leveraging visual foundation models for label-free, risk-aware off-road navigation,” in *Conference on Robot Learning (CoRL)*, 2024.
- [8] C. Huang, O. Mees, A. Zeng, and W. Burgard, “Visual language maps for robot navigation,” in *International Conference on Robotics and Automation (ICRA)*, 2023.
- [9] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, “VLFM: Vision-language frontier maps for zero-shot semantic navigation,” in *International Conference on Robotics and Automation (ICRA)*, 2024.
- [10] D. Goetting, H. G. Singh, and A. Loquercio, “End-to-end navigation with vision-language models: Transforming spatial reasoning into question-answering,” in *International Conference on Neuro-symbolic Systems (NeuS)*, 2025.
- [11] K. Weerakoon, *et al.*, “BehAV: Behavioral rule guided autonomy using VLMs for robot navigation in outdoor scenes,” in *International Conference on Robotics and Automation (ICRA)*, 2025.
- [12] M. Yuan, L. Wang, and S. L. Waslander, “OpenNav: Open-world navigation with multimodal large language models,” in *International Conference on Intelligent Robots and Systems (IROS)*, 2025.
- [13] S. Otte, C. Weiss, T. Scherer, and A. Zell, “Recurrent neural networks for fast and robust vibration-based ground classification on mobile robots,” in *International Conference on Robotics and Automation (ICRA)*, 2016.
- [14] X. Cai, *et al.*, “EVORA: Deep evidential traversability learning for risk-aware off-road autonomy,” *IEEE Transactions on Robotics (T-RO)*, vol. 40, pp. 3756–3777, 2024.
- [15] Y. Kim, *et al.*, “Learning semantic traversability with egocentric video and automated annotation strategy,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, no. 11, pp. 10 423–10 430, 2024.
- [16] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-end driving via conditional imitation learning,” in *International Conference on Robotics and Automation (ICRA)*, 2018.
- [17] D. Shah, A. Bhorkar, H. Leen, I. Kostrikov, N. Rhinehart, and S. Levine, “Offline reinforcement learning for visual navigation,” in *Conference on Robot Learning (CoRL)*, 2022.
- [18] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, “GNM: A General Navigation Model to Drive Any Robot,” in *International Conference on Robotics and Automation (ICRA)*, 2023.
- [19] D. Shah, *et al.*, “ViNT: A foundation model for visual navigation,” in *Conference on Robot Learning (CoRL)*, 2023.
- [20] A. Sridhar, D. Shah, C. Glossop, and S. Levine, “NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration,” in *International Conference on Robotics and Automation (ICRA)*, 2024.
- [21] S. Hutchinson, G. Hager, and P. Corke, “A tutorial on visual servo control,” *IEEE Transactions on Robotics and Automation (T-RA)*, vol. 12, no. 5, pp. 651–670, 1996.
- [22] Y. Ma, J. Kosecka, and S. Sastry, “Vision guided navigation for a nonholonomic mobile robot,” in *Conference on Decision and Control (CDC)*, 1997.
- [23] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Alvey Vision Conference*, 1988.
- [24] E. Marchand, F. Spindler, and F. Chaumette, “ViSP for visual servoing: a generic software platform with a wide class of robot control skills,” *IEEE Robotics & Automation Magazine (RAM)*, vol. 12, no. 4, pp. 40–52, 2005.
- [25] J. A. Sethian, “A fast marching level set method for monotonically advancing fronts,” *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.
- [26] W.-K. Jeong and R. T. Whitaker, “A fast iterative method for eikonal equations,” *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2512–2534, 2008.
- [27] A. Valero-Gomez, J. V. Gomez, S. Garrido, and L. Moreno, “The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories,” *IEEE Robotics & Automation Magazine (RAM)*, vol. 20, no. 4, pp. 111–120, 2013.
- [28] M. G. Castro, *et al.*, “VAMOS: A hierarchical vision-language-action model for capability-modulated and steerable navigation,” arXiv:2510.20818, 2025.
- [29] S. Liu, *et al.*, “Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection,” in *European Conference on Computer Vision (ECCV)*, 2024.
- [30] T. Ren, *et al.*, “Grounded SAM: Assembling open-world models for diverse visual tasks,” arXiv:2401.14159, 2024.
- [31] N. Ravi, *et al.*, “SAM 2: Segment anything in images and videos,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [32] N. Carion, *et al.*, “SAM 3: Segment anything with concepts,” in *International Conference on Learning Representations (ICLR)*, 2026.
- [33] O. Siméoni, *et al.*, “DINOv3,” arXiv:2508.10104, 2025.
- [34] C. Jose, *et al.*, “DINOv2 meets text: A unified framework for image- and pixel-level vision-language alignment,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [35] A. Orsula, M. Geist, M. Olivares-Mendez, and C. Martinez, “Space Robotics Bench: Robot Learning Beyond Earth,” arXiv:2509.23328, 2025.
- [36] M. Mittal, *et al.*, “Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning,” arXiv:2511.04831, 2025.
- [37] NVIDIA, “Isaac Sim,” <https://github.com/isaac-sim/IsaacSim>, 2025, version 6.0.0.
- [38] N. Hirose, D. Shah, A. Sridhar, and S. Levine, “SACSoN: Scalable autonomous control for social navigation,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, no. 1, pp. 49–56, 2024.
- [39] P. Anderson, *et al.*, “On Evaluation of Embodied Navigation Agents,” arXiv:1807.06757, 2018.
- [40] R. M. Swan, *et al.*, “AI4MARS: A dataset for terrain-aware autonomous driving on Mars,” in *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 1982–1991.
- [41] Indian Space Research Organisation, “Chandrayaan-3 Pragyan rover navigation camera (NavCam) imagery,” ISRO Science Data Archive, PRADAN, <https://pradan.issdc.gov.in/ch3/>, Aug. 2024.
- [42] NASA Jet Propulsion Laboratory, “NASA Awards Next-Generation Spaceflight Computing Processor Contract,” <https://www.jpl.nasa.gov/news/nasa-awards-next-generation-spaceflight-computing-processor-contract/>, Aug. 2022.
- [43] J. Balaram, *et al.*, “Mars helicopter technology demonstrator,” in *AIAA Atmospheric Flight Mechanics Conference, AIAA SciTech Forum*, 2018.
- [44] Z. Towfic, *et al.*, “Benchmarking and testing of Qualcomm Snapdragon system-on-chip for JPL space applications and missions,” in *IEEE Aerospace Conference*, 2022.