DTRAP: Dynamic Speed Profiling-based Traversability-aware Planning for Planetary Navigation

Suchetan Saravanan and Damien Vivet

***Abstract*— Autonomous navigation on unstructured and uneven terrain remains a critical challenge for planetary rovers, requiring both safe path planning and adaptive speed control. Traditional occupancy or elevation maps often fail to capture the full complexity of terrain hazards and rover mobility constraints. We present a global planning framework that integrates traversability analysis with velocity profile estimation. Local terrain features such as slope, step height, and roughness are extracted from elevation data and normalized by rover-specific parameters including maximum admissible slope and ground clearance. These form a costmap used in graph-based global planning, while the planner simultaneously adjusts target speeds to safely negotiate terrain hazards. Our method produces dynamically feasible trajectories that improve autonomous exploration capabilities in challenging planetary environments. Experimental results demonstrate that our approach improves safety by 44%, reduces maximum terrain hazard by over 10%, and decreases traversal time by nearly 10% compared to state-of-the-art planners.**

I. INTRODUCTION

Autonomous navigation in unstructured and uneven terrain is a fundamental challenge for planetary rovers, where long-term safety, energy efficiency, and mission reliability are critical. In such environments, conventional occupancy-based representations are insufficient: the robot must reason not only about obstacle presence, but about how traversable a terrain is, given its own mechanical limitations.

Environment modelling for autonomous exploration typically relies on sensor data such as LiDAR or stereo vision [1], [2], [3]. Volumetric mapping frameworks like OctoMap [4], [5], UFOMap [6], and Voxblox [7] offer efficient occupancy modeling but remain limited to binary or probabilistic classifications (free/occupied/unknown). These lack the geometric richness and semantics required for fine-grained traversability assessment—especially in rough terrain where mobility is highly robot-specific.

To address this limitation, recent work has introduced traversability as an additional layer in the planning stack. Some approaches compute local traversability costs on-the-fly to support reactive motion planning [8], while others generate global traversability maps from elevation data [9], [10] using metrics such as slope, roughness, and step height. However, these methods often decouple mapping from localization, leading to drift or inconsistencies during long-range or low-feature exploration. More critically, they frequently fail to normalize terrain difficulty with respect to the robot's physical characteristics—such as wheel radius, ground clearance, or stability thresholds—limiting their generalizability and reliability.

Recent trends have focused on incorporating terrain features directly into global planners, adapting classical methods like Hybrid A* or RRT* to include traversability-aware heuristics. For example, Traversability Hybrid A* [11] augments the heuristic with terrain costs to avoid risky areas, while terrain-weighted graphs have been used with A* to account for directional slope or step interactions [12]. Sampling-based methods assess terrain safety during node expansion using local surface metrics [13] or interpolate sparse terrain estimates via Gaussian Processes.

Despite these advances, most planners treat velocity control as a separate or post hoc module that is handled by the path-following controller. In planetary navigation, safety is the top priority, and a cautious controller will naturally slow down in proportion to the terrain's traversal cost. However, if velocity considerations are left out of the planning stage, the resulting path may be safe but unnecessarily slow and may not respect the robot's velocity constraints leading to sudden braking upon encountering an obstacle on the path or inability to slow-down in time resulting in unsafe movement. Local dynamic planners like the Dynamic Window Approach [14] or the Model-predictive path-integral controller [15] integrate velocity and acceleration constraints, but operate over short horizons and lack access to global terrain context.

While prior work addresses parts of this problem space, no existing method integrates both global path planning and terrain-aware speed control in a unified framework. Some systems build traversability-aware graphs for safe path generation (e.g., TRG-planner [16]); others learn terrain-conditioned speed distributions [17], or use MPC to optimize over terrain and dynamics jointly (e.g., STEP [18]). However, these typically separate path generation and speed control, or limit velocity reasoning to local refinements. In contrast, our approach integrates both cell-wise terrain cost and admissible velocity profiles directly into the structure of a global planner, enabling coherent, risk-aware motion planning from the outset.

In this work, we propose a traversability-aware global planning framework that jointly estimates terrain-safe trajectories and terrain-adaptive velocity profiles while respect-

*This work was supported by the French "Agence Innovation Defence" AID under Grant 2023-65-0083.

University of Toulouse, ISAE-SUPAERO, France damien.vivet@isae-supaero.fr

ing the robot's velocity constraints. Our approach extracts local geometric descriptors—such as slope magnitude and direction, step height, and surface roughness—from elevation maps, and normalizes them according to rover-specific parameters (e.g., maximum slope tolerance, wheel radius, center of gravity, and ground clearance). These normalized hazard metrics are then fused into a global traversability costmap that guides graph construction and global path planning. This formulation enables the seamless integration of diverse robotic platforms without extensive parameter tuning, and directly produces a velocity profile usable by the robot during traversal of the planned path.

Uniquely, our method encodes both traversability and velocity constraints during graph construction, assigning edge costs that reflect terrain difficulty and speed feasibility simultaneously. This allows the planner to proactively slow down in high-risk regions—such as when approaching steps—where velocity must be reduced relative to wheel size and terrain height or on slopes—where velocity must be reduced relative to the robot's 3D orientation and the surface normal to maintain stability. The result is a globally consistent, terrain-aware planner that produces dynamically safe trajectories suited for long-range, autonomous planetary navigation, where energy efficiency and operational robustness are paramount.

II. METHODOLOGY

Our proposed system comprises two main components: (1) a terrain-aware hazard estimation pipeline that constructs a traversability costmap from local 3D perception, and (2) a global planner that leverages this map to compute both a safe trajectory and a dynamically admissible velocity profile. The core idea is to normalize terrain hazards according to the rover's physical parameters (e.g., wheel radius, center of gravity, ground clearance), and to integrate this representation directly into the planner's cost and speed computation. This enables the planner to make globally consistent decisions that reflect both geometric difficulty and dynamic feasibility.

A. Environment modeling and traversability estimation

We introduce a terrain-aware traversability mapping approach that generates a 2D global costmap from dense 3D point cloud data. This map serves as the foundation for global path planning by encoding both terrain-dependent cost and velocity constraints.

The input point cloud, typically acquired from LiDAR, stereo, or RGB-D sensors, is processed locally to extract three key geometric descriptors: *slope* (magnitude and direction), *step height*, and *surface roughness*. These features are normalized using the rover's mechanical specifications, including maximum slope tolerance, wheel radius, ground clearance, and center of gravity. The result is a hazard vector for each cell, capturing terrain difficulty relative to the specific robot.

To ensure robustness and temporal consistency, we maintain a persistent global hazard grid G. Each cell accumulates hazard vectors from overlapping local observations. Upon receiving a new scan, the point cloud is transformed to the world frame and segmented into grid cells. In each populated cell, geometric descriptors are recomputed, normalized, and appended to the cell's history. The median value of each hazard component is retained to reduce the influence of noise and outliers.

A traversability cost is then assigned to each cell in the global costmap $\mathcal{M}$ as the maximum among the median-normalized slope, step height, and roughness values. This robot-specific and spatially consistent map encodes both hazard severity and traversability limitations. An example is shown in Figure 1, and the mapping process is detailed in Algorithm 1. More implementation details are available in our prior work [19].

Algorithm 1 Global traversability mapping algorithm.

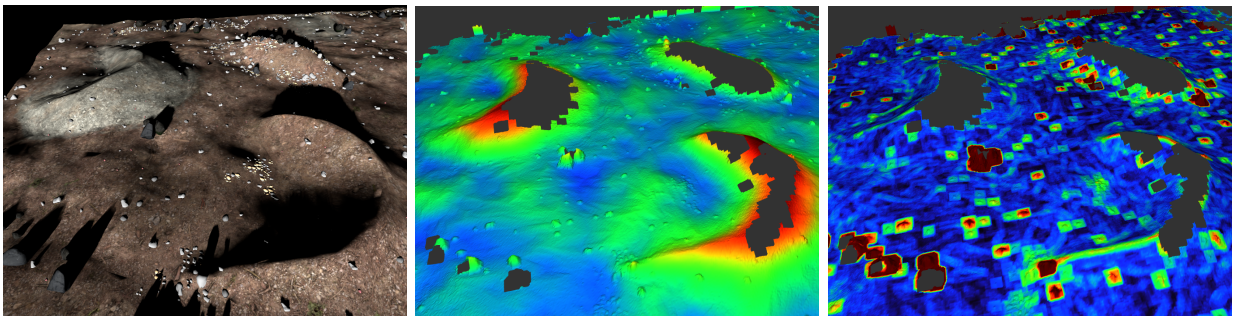
Require: A 2D global costmap $\mathcal{M}$, a 2D global hazard grid G, a point cloud in the sensor frame $\mathcal{P}_s$, a robot pose ${}^{w}T_r$, an extrinsic ${}^{r}T_s$, the maximum step the robot can cross h_{max}, the maximum slope the robot can cross α_{max}

Transform $\mathcal{P}_s$ in the world frame using ${}^{w}T_r$ and ${}^{r}T_s$ into $\mathcal{P}_w$
for Each cell c in G **do**
  if c contains enough point from $\mathcal{P}_w$ **then**
    $\mathcal{P}_c$ is the subset of $\mathcal{P}_w$ contained in c
    Compute the barycentre $\mathbf{b}_c$ and the covariance Σ_c of $\mathcal{P}_c$
    Compute $SVD(\Sigma_c)$ and get the main direction of the subset $\mathbf{d}_c$ and its normal $\mathbf{n}_c$
    Compute the normalized step height $h = (z_{\text{max}} - z_{\text{min}})/h_{\text{max}}$
    Compute the normalized slope $\alpha = \arccos(\mathbf{n}_c \cdot [0,0,1])/\alpha_{\text{max}}$
    Compute the normalized roughness $r = \frac{\sum_{\mathbf{p} \text{ in } \mathcal{P}_c} (\mathbf{p}-\mathbf{b}_c)\cdot\mathbf{n}_c}{n h_{\text{max}}}$
    Add the hazard triplet $\{h, \alpha, r\}$ in c
    Compute the median values of each hazard in c as $\{h_m, \alpha_m, r_m\}$
    Update the global costmap $\mathcal{M}(c) = \max\{h_m, \alpha_m, r_m\}$
  end if
end for

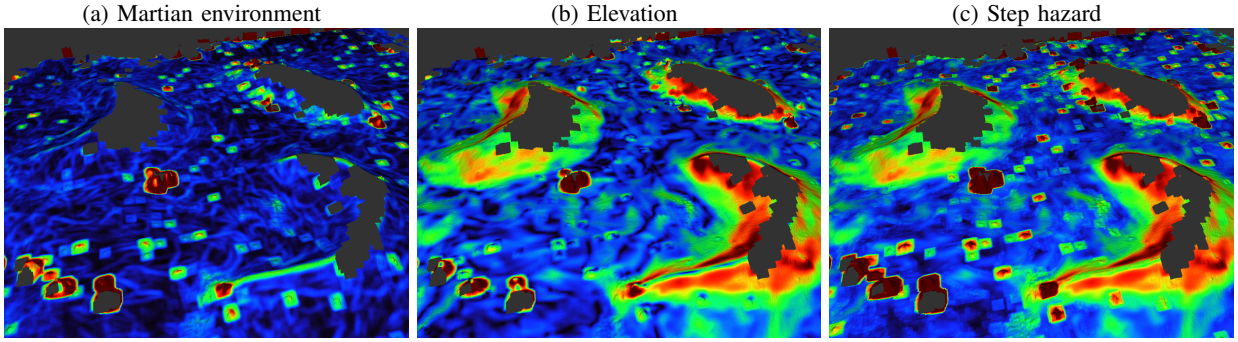
The resulting traversability map $\mathcal{M}$ is used in downstream planning to evaluate both path cost and speed feasibility. For any candidate trajectory, the traversal cost and safe velocity profile are derived directly from $\mathcal{M}$, enabling both collision-free and terrain-compliant navigation. The next section details how this velocity-aware planning is implemented.

B. Traversability-aware Velocity constraints

While the traversability costmap $\mathcal{M}$ encodes the geometric difficulty of the terrain, it remains necessary to adjust the robot's velocity according to the severity of the upcoming



(a) Martian environment (b) Elevation (c) Step hazard



(d) Roughness hazard (e) Slope hazard (f) Maximum hazard value

Fig. 1: The simulated environment (a) along with the extracted hazard maps. (b) shows the mean elevation computed from accumulated LiDAR scans. (c), (d) and (e) represent the step, the roughness and slope hazard layers derived from the elevation data. Finally, (f) shows the maximum hazard value per cell, used to identify the most critical areas in the environment.

hazards. In particular, sharp vertical discontinuities such as steps impose mechanical and dynamic constraints that require the rover to reduce its speed to avoid loss of stability or damage from impacts.

1) Step-Based Speed Regulation: To ensure safe step traversal, a velocity profile is generated such that the robot decelerates smoothly to a predefined safe speed v_s^{step} before reaching the step, and accelerates symmetrically after crossing it. The profile is modelled as a smoothed step function centered on the step location, using a sigmoid transition:

$$v^{\text{step}}(x) = v_n - (v_n - v_s^{\text{step}}).\left(\frac{1}{1+e^{\gamma(|x|-\delta)}}\right) \quad (1)$$

where:

- v_n is the nominal (maximum) cruising speed,
- v_s^{step} is the desired step-crossing speed,
- γ controls the sharpness of the deceleration profile,
- x is the signed distance to the step (zero at step front),
- δ defines the point at which $v(x) \approx v_s^{\text{step}}$

The value of δ is computed using:

$$\delta = d + \frac{1}{\gamma}\ln\left(\frac{v_n - v_s^{\text{step}}}{\epsilon} - 1\right) \quad (2)$$

Here, d represents the distance from the step at which the velocity v_s^{step} is expected to be reached, and ϵ defines the acceptable deviation from that target speed v_s^{step} given by:

$$v_s^{\text{step}} = (1-h)\cdot v_n \quad (3)$$

where $h \in [0,1]$ is the normalized step hazard.

The resulting velocity profile is symmetric around the step and ensures a smooth transition in both approach and departure phases. An illustration is provided in Figure 2.

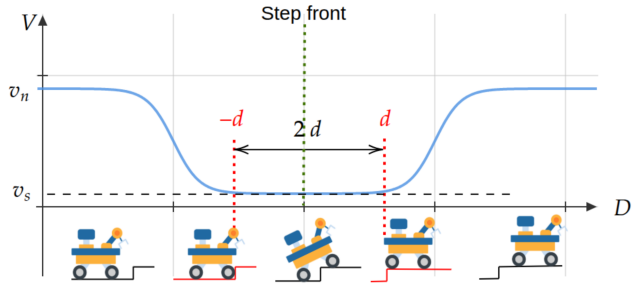


Fig. 2: Step climbing velocity profile. When approaching a step, the robot decelerates to reach the desired speed v_s at distance d from the step front. Once passed, a symmetric acceleration profile allows recovery to the nominal speed v_n.

2) Slope-Based Velocity Regulation: To ensure stability on sloped terrain, the local surface gradient is decomposed into two directional components: one aligned with the direction of travel (longitudinal slope), and one orthogonal to it (lateral slope). These components are used to assess the risk of tipping or slippage, and to scale the admissible velocity accordingly.

At each cell (x, y), the terrain slope magnitude $\theta(x, y)$ and gradient direction $\phi(x, y)$ are first extracted from elevation

data. Given a planned heading ψ, the slope components are defined as:

$$\begin{cases} \theta_{\parallel}(x, y, \psi) = \theta(x, y) \cdot \cos\left(\phi(x, y) - \psi\right) \\ \theta_{\perp}(x, y, \psi) = \theta(x, y) \cdot \sin\left(\phi(x, y) - \psi\right) \end{cases} \tag{4}$$

where:

- $\theta_{\parallel}$ represents the longitudinal slope (along the path),
- $\theta_{\perp}$ represents the lateral slope (across the path),
- ψ is the robot's heading in the map frame.

Stability constraints are then derived from the position of the rover's center of gravity Z_{cg}, and the horizontal distances between the CoG and the vehicle's support points in both the lateral (X_{cg}^f, X_{cg}^r) and longitudinal (Y_{cg}^f, Y_{cg}^r) directions. An illustration is provided in Figure 3.

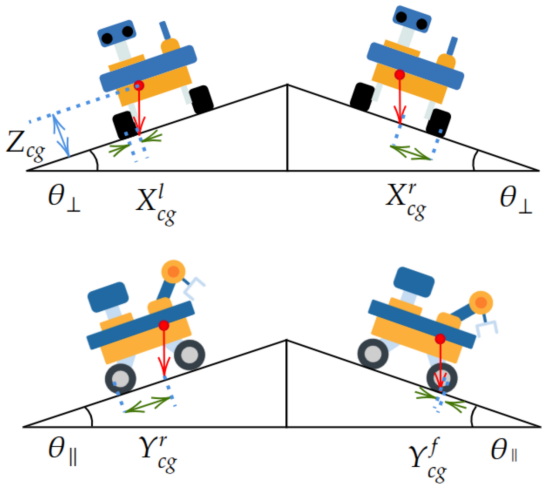


Fig. 3: Lateral and longitudinal stability margins

The maximal admissible pitch P and roll R values can then be used to estimate the maximum admissible slopes in each directions such as:

$$\begin{cases} \theta_{\perp max} = \; min \left\{ \underbrace{atan\left(\frac{X_{cg}^f}{Z_{cg}}\right)}_{R_{cw}}, \underbrace{atan\left(\frac{X_{cg}^r}{Z_{cg}}\right)}_{R_{ccw}} \right\} \\ \theta_{\parallel max} = \; min \left\{ \underbrace{atan\left(\frac{Y_{cg}^f}{Z_{cg}}\right)}_{P_{up}}, \underbrace{atan\left(\frac{Y_{cg}^r}{Z_{cg}}\right)}_{P_{dw}} \right\} \end{cases} \tag{5}$$

These limits define the slope angles at which the rover would become statically unstable (e.g., prone to tipping or sliding). A binary traversability mask is then derived to determine whether the slope is acceptable:

$$\text{Traversable}(x, y, \psi) = \left(|\theta_{\parallel}| \leq \theta_{\parallel}^{\max}\right) \wedge \left(|\theta_{\perp}| \leq \theta_{\perp}^{\max}\right) \tag{6}$$

In practice, rather than relying on a hard constraint, a continuous directional slope cost is defined to penalize near-limit slopes:

$$\text{cost}(x, y, \phi) = w_1 \frac{|\theta_{\parallel}|}{\theta_{\parallel}^{\max}} + w_2 \frac{|\theta_{\perp}|}{\theta_{\perp}^{\max}} \tag{7}$$

where w_1 and w_2 are tunable weights to prioritize forward vs. lateral stability.

To maintain stability and control, the rover's speed is scaled as a function of slope severity. A smooth attenuation function $f(\theta_{\text{min}})$ is applied to reduce the velocity as the slope approaches critical values:

$$v^{\text{slope}}(x, y, \theta_{\text{min}}) = v_n \min\left\{ f\left(\frac{|\theta_{\parallel}|}{\theta_{\parallel}^{\max}}\right), f\left(\frac{|\theta_{\perp}|}{\theta_{\perp}^{\max}}\right) \right\} \tag{8}$$

where v_n is the nominal cruise speed, and the attenuation function is defined as:

$$f(\theta_{\text{min}}) = \max\left\{ \frac{v_s^{\text{slope}}}{v_n}, e^{-k\theta_{\text{min}}} \right\} \tag{9}$$

with:

- k a user-defined attenuation gain,
- θ_{min} the most constraining slope parameter given by eq. 10.

$$\theta_{\text{min}} = \min\left\{ \frac{|\theta_{\parallel}|}{\theta_{\parallel}^{\max}}, \frac{|\theta_{\perp}|}{\theta_{\perp}^{\max}} \right\} \tag{10}$$

- v_s^{slope} the minimum admissible velocity in steep terrain defined by:

$$v_s^{\text{slope}} = (1 - \alpha) \cdot v_n \tag{11}$$

with $\alpha \in [0, 1]$ the normalized slope hazard.

This profile enables smooth velocity modulation across varying inclinations, ensuring that the rover slows down in potentially unstable configurations without relying on abrupt or binary decision rules.

3) Roughness Speed Regulation: While slope and step height are typically used to constrain stability, terrain roughness directly impacts the mechanical stress and vibration experienced by the rover's suspension and onboard sensors. To account for this, a roughness-based speed limit is used in highly irregular areas such as:

$$v_s^{\text{rough}} = (1 - r) \cdot v_n \tag{12}$$

where $r \in [0, 1]$ is the normalized roughness score at a given cell.

Each terrain hazard imposes an independent constraint on the maximum allowable speed: slope affects stability, step height constrains safe traversal of vertical discontinuities, and roughness limits vibrations. The integration and use of these different velocity profile is detailed in section II-C.2.

C. Traversability and velocity constrained planner

The goal of the planner is to compute a safe and dynamically feasible trajectory from a start to a goal location in a partially known terrain, using the traversability map $\mathcal{M}$, the robot's motion and stability constraints.

Our approach combines a graph-based representation of the environment with edge-wise risk and velocity estimation, enabling efficient pathfinding that takes into account both geometry-induced hazards and speed constraints.

1) Graph Construction from Traversability Map: Planning directly over the full traversability map $\mathcal{M}$ using grid-based search can quickly become computationally expensive, especially in large-scale environments or when considering rich cost functions. To address this, we precompute a sparse graph-based roadmap that abstracts the terrain while preserving its key geometric and traversability features. This static graph, built once from the prior map, provides a lightweight planning structure that enables fast and efficient queries during mission execution.

We generate a sparse graph-based roadmap $\mathcal{G} = (V, E)$ over the traversability map $\mathcal{M}$ using a wave-front expansion method. The generation process begins with a set of seed nodes (sampled or on a grid) in a queue. We iteratively deque a node, sample new candidate nodes at a fixed radius r, and validate them. An edge is created if the parent and children nodes are on traversable terrain and the line-of-sight path between the two nodes is also traversable. Each edge is directed independently from the parent node to the child and vice-versa, establishing $\mathcal{G}$ as a directed graph. Validated new nodes are added to the graph and the queue, continuing the expansion until all reachable areas are connected.

For each edge $e \in E$ connecting nodes A and B, we define:

- $\mathcal{L}_e$: the physical length of the edge
- $\mathcal{C}_e$: the cumulative traversability cost
- $t_e = \frac{\mathcal{L}_e}{\bar{v}_e}$: the time to cross that edge with $\bar{v}_e$, the mean admissible velocity profile along e.

2) Velocity Profile Integration: For each candidate edge e, we evaluate its velocity profile based on the underlying traversability cells. Each cell contributes a local velocity bound derived from either step geometry (Eq. 1), slope stability (Eq. 8) or roughness (Eq. 12).

The global velocity profile along the edge is then computed by taking into account the constraints of the N traversed traversability map cells. For each cell i, a local velocity bounds v_i is processed from terrain features. Finally, the minimal admissible speed for all the obtained profiles is kept to obtain the final profile:

$$v_e(x) = \min_i \{v_i(x)\} \tag{13}$$

An example of this process is illustrated in Fig. 4, where the global edge velocity profile results from the minimum of all individual terrain-induced constraints.

The mean admissible speed $\bar{v}_e$ is then computed as:

$$\bar{v}_e = \frac{1}{\mathcal{L}_e} \int_0^{\mathcal{L}_e} v_e(x)\, dx \tag{14}$$

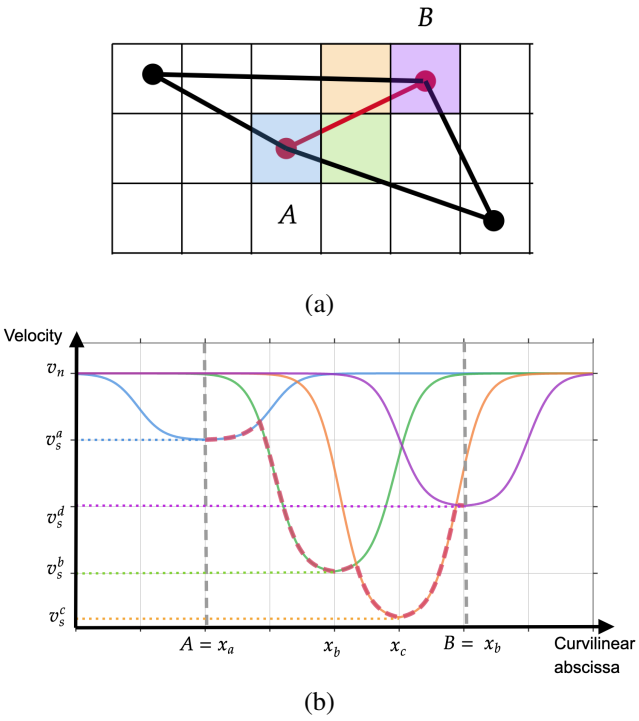


Fig. 4: Velocity profile estimation for a given edge. (a) Given a edge $A \rightarrow B$ of the roadmap $\mathcal{G}$ for which the velocity profile has to be determined, each traversed cell of the traversability map is considered to contribute individually resulting in N velocity profiles (b) based on equations (1) and (8). The minimal admissible value of each velocity profile is the kept to define the global edge velocity profile in dash red.

This value is used in Eq. 15 to balance between fast and safe motion. Note that, at this step, discontinuities in the velocity profile will occur at edge transition. Such discontinuities will be mitigated once the best path found and the final profile is computed in a single pass.

3) Path Planning Algorithm: Given the graph $\mathcal{G}$ with edge costs $\mathcal{C}_e$ and edge traversal time t_e, we apply standard Dijkstra search to find the minimum-cost path from start to goal. Since each edge already incorporates terrain risk and velocity feasibility, the planning heuristic is changed to reduce the overall navigation time along the path. The final cost to be optimized is then only the time of traversal as velocities already account for traversability risk (a non traversable area has a null velocity).

$$\mathcal{C}_g = t_e = \frac{\mathcal{L}_e}{\bar{v}_e} \tag{15}$$

Once the best path obtained, a complete velocity profile is reprocessed in a single pass to assure a smooth velocity transition between successive edges all along the trajectory.

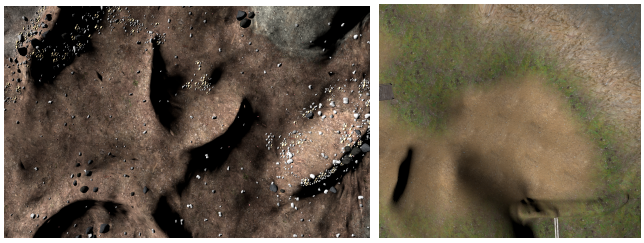


Fig. 5: The simulation worlds used in the experiments. **Left**: Martian world. **Right**: Clearpath inspection world.

III. EXPERIMENTS AND DISCUSSION

A. Simulation Setup

Experiments were carried out in two virtual environments using Gazebo Harmonic as the simulation back-end, integrated with ROS 2 Jazzy (Figure 5).

The first testbed is a Martian analogue terrain derived from the European Rover Challenge (ERC) 2022 benchmark, containing representative planetary features such as rocky outcrops, irregular slopes, and impact craters. The second environment is a modified version of the Clearpath Inspection World, in which unstructured regions with varying slope gradients were introduced to evaluate the planner's performance in slope-constrained and high-risk areas.

The simulated rover is a four-wheel differential-drive platform equipped with a Velodyne VLP-16 3D LiDAR, publishing scans at 15 Hz. It has a ground clearance of 20 cm, a wheel radius of 15 cm, and can tolerate slopes of up to 28° laterally and 45° longitudinally. The nominal cruise speed is 1.2 m/s.

For the experiments, the traversability costmaps are generated offline from LiDAR data at a grid resolution of 0.05 m. The proposed velocity-aware global planner operates directly on these maps, enabling simultaneous computation of safe trajectories and admissible speed profiles.

B. Directionality property

The first set of experiments illustrates the *directionality property* of the proposed planner: its ability to adapt global paths according to asymmetric limits in admissible pitch and roll. Figure 6 shows six example paths computed under different stability constraints.

Here, P_{up} and P_{down} denote the maximum admissible longitudinal slopes for uphill and downhill motion, respectively. Likewise, R_{cw} and R_{ccw} correspond to the maximum admissible lateral slopes for clockwise and counter-clockwise roll (as defined in Eq. 5). These values are normalized to $[0, 1]$ relative to the platform's physical limits.

The evaluated parameter sets for each path are summarized in Table I. Variations in these constraints bias the planner toward different strategies for reaching the goal, selectively avoiding certain orientations or slope directions as dictated by the permitted stability envelope. The first checkered flag at the bottom of the image corresponds to the start position and the second one corresponds to the goal position. Paths A and B have an upper bound on P_{down} which results in paths

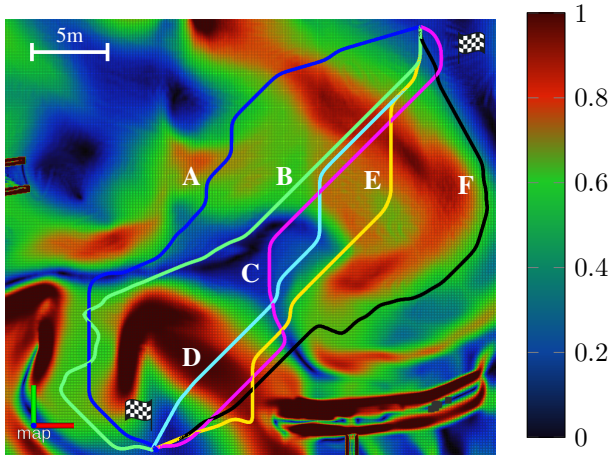


Fig. 6: Planned trajectories (A to F) in the inspection environment under different directional constraints. Each path is generated by imposing specific limits on longitudinal pitch (up and down) and lateral roll (clockwise and counterclockwise) angles. These constraints influence the planner's route selection to ensure compliance with the robot's stability and mobility capabilities. The background colormap represents the maximum hazard value across the terrain, highlighting areas of increased traversal risk.

that pass through the gentler downward slope on the left. In contrast, C-F are allowed to pass through the steeper slope near label D. A similar behaviour can be observed for paths A and F which have a limit on the upward pitch parameter (P_{up}). Path C has the highest constraint on the admissable roll values due to which it prioritizes straight paths up to the goal. Finally, paths A and F have constraints on the R_{ccw} and R_{cw} parameters due to which penalizing of paths in the corresponding roll direction can be observed.

Path	R_{cw}	R_{ccw}	P_{up}	P_{down}
A	0.80	0.05	0.40	0.40
B	0.33	0.33	0.80	0.30
C	0.10	0.10	0.80	0.80
D	0.33	0.33	0.80	0.80
E	0.50	0.50	0.50	0.50
F	0.20	0.33	0.30	0.80

TABLE I: Directionality constraints applied in each path scenario. Values are normalized with respect to the rover's maximum physical limits.

C. Comparison with other planners

We evaluate the proposed traversability-aware path-planning approach against a diverse set of planners representing different planning paradigms. For graph/tree-based methods that rely on random sampling, we compare against **RRT*** and **PRM***. For grid-based search methods operating directly on the occupancy grid, we evaluate against **A*** and **Dijkstra**. To account for planners that explicitly handle kinematic constraints and generate smooth, drivable paths, we include the recent **Hybrid A*** algorithm [11].

In all cases, the same traversability costmap generated by our framework is used as the planning input to ensure a fair comparison.

Figure 8 shows the resulting paths overlaid on the traversability map, while Figure 7 presents the corresponding velocity profiles post-processed with our method on the planned path. We assess performance using the following metrics: (i) *Path length* P_l (m), (ii) *Maximum hazard value* along the path H_{max}, (iii) *Mean hazard value* H_{av}, and (iv) *Traversal time* P_t (s), computed by applying our velocity estimation model to each planned path.

Table II summarizes the results of the plans shown in 8, with the best score for each metric highlighted in **bold**. Notably, our method consistently achieves the shortest traversal time and the lowest hazard exposure while maintaining competitive path length.

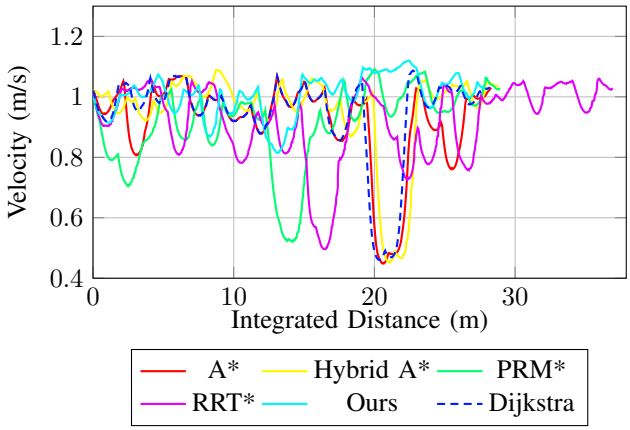


Fig. 7: A comparison of velocity profiles generated on the path planned by different algorithms. The velocity of the agent is plotted against the cumulative distance travelled along its path.

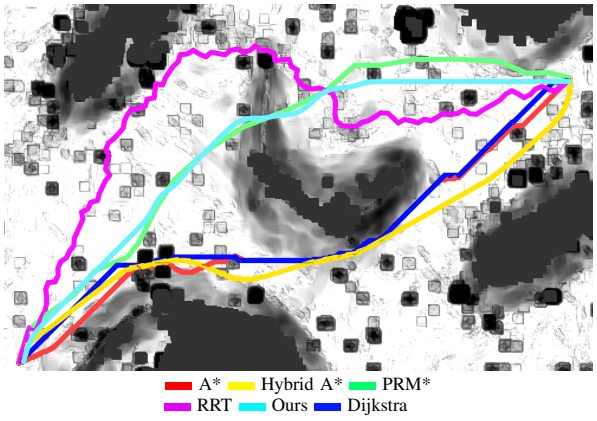


Fig. 8: A figure showing different paths planned by different planning algorithms in the martian world.

Overall, our approach outperforms all other methods across all metrics. While A* and Dijkstra produce paths of similar length to ours, their traversal times are longer due to post hoc velocity assignment rather than joint path–velocity optimization. This difference is particularly evident around the 20 m mark in Figure 7, where all planners traverse a high-cost area (bottom left in Figure 8). Our planner mitigates speed loss in such zones by integrating velocity constraints directly into the planning stage, resulting in both safer and faster navigation.

Our experiments demonstrate that integrating terrain-aware traversability costs with velocity constraints in a unified planning step yields safer, faster, and more efficient paths than traditional planners. By accounting for both geometric hazards and rover dynamics during planning rather than as an afterthought our approach consistently minimizes hazard exposure and travel time across challenging terrains.

D. Planning performance

For planetary rovers, computational efficiency is of utmost importance, as onboard processing resources are often limited and timely decision-making is critical for mission safety. To assess the computational performance of our approach, we benchmarked the planning time of our method against several commonly used planners. Each planner was tasked with completing 1000 independent planning queries between the start and goal points shown in Fig. 8. For multi-query planners that require an initial graph construction phase (e.g., PRM*), this graph was generated once prior to the planning iterations. We report: (i) the graph generation time (T_g), (ii) the mean planning time per query (T_p), and (iii) the total time required to complete all 1000 queries (T_{1000}).

The results are summarized in Table III. Our method significantly outperforms traditional grid-based search algorithms (A*, Dijkstra) and optimization-based planners (RRT*, Hybrid A*) in terms of planning speed, achieving a mean planning time of only 5.5 ms. Notably, RRT* and Hybrid A* are several orders of magnitude slower, making them impractical for repeated replanning in resource-constrained environments.

IV. CONCLUSIONS

We presented a unified global planning framework that jointly computes terrain-aware paths and feasible velocity profiles, accounting for both geometric hazards and robot-specific mobility constraints. By using local terrain features such as slope, step height, and roughness, our approach generates safe, smooth, and dynamically feasible trajectories through unstructured environments. Simulations in two planetary analogue worlds confirm that the system effectively avoids hazardous regions and adapts speed to maintain stability, outperforming a diverse set of planners in both safety and efficiency.

Despite these promising results, the current framework relies solely on low-level geometric descriptors, which limits its ability to anticipate more complex terrain phenomena such as slippage, soil deformability, or traction loss. In future work, we aim to incorporate semantic terrain understanding

Metric	Ours	A*	Dijkstra	Hybrid A*	RRT*	PRM*
P_l (m)	**28.08**	28.29	28.21	28.83	37.23±5.67	29.12±0.67
H_{max}	**0.35**	0.68	0.67	0.68	0.69±0.07	0.69±0.05
H_{av}	**0.16**	0.23	0.21	0.20	0.27±0.04	0.25±0.03
P_t (s)	**28.11**	32.13	31.13	31.52	37.22±0.66	29.12±5.66

TABLE II: Quantitative comparison of planners on traversability-aware metrics. Best values in **bold**. For statistical methods results are presented as mean±std (based on 30 runs).

Planner	T_g (ms)	T_p (ms)	T_{1000} (ms)
Ours	**440**	**5.5**	**5100**
A*	–	57.57	5717
Dijkstra	–	61.0	6103
RRT*	–	2176.3	NA
PRM*	6800	14.6	7810
Hybrid A*	–	3020.5	NA

TABLE III: Planning performance comparison across different planners. T_g: graph generation time, T_p: mean planning time per query, T_{1000}: total planning time for 1000 queries. NA indicates that the measurement was not completed due to excessive computation time.

using visual and proprioceptive sensing to estimate non-geometric risk factors, and to integrate these cues more tightly into the planning process.

REFERENCES

[1] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," 2023.

[2] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual slam," *Expert Systems with Applications*, vol. 205, p. 117734, 2022.

[3] C. Debeunne and D. Vivet, "A review of visual-lidar fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, 2020.

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. Software available at `https://octomap.github.io`.

[5] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger, "Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1144–1151, April 2018.

[6] D. Duberg and P. Jensfelt, "UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6411–6418, 2020.

[7] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1366–1373, 2017.

[8] A. Patel, M. A. V. Saucedo, C. Kanellakis, and G. Nikolakopoulos, "Stage: Scalable and traversability-aware graph based exploration planner for dynamically varying environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5949–5955, 2024.

[9] Y. Jia, W. Tang, H. Sun, J. Yang, B. Liu, and C. Wang, "Portable planner for enhancing ground robots exploration performance in unstructured environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9295–9302, 2024.

[10] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation mapping for locomotion and navigation using gpu," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2273–2280, IEEE, 2022.

[11] M. Thoresen, N. H. Nielsen, K. Mathiassen, and K. Y. Pettersen, "Path planning for ugvs based on traversability hybrid a*," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1216–1223, 2021.

[12] Y. Tang, Q. Li, H. Geng, Y. Xie, H. Shi, and Y. Yang, "Path planning on multi-level point cloud with a weighted traversability graph," *arXiv preprint arXiv:2504.21622*, 2025.

[13] Z. Jian, Z. Lu, X. Zhou, B. Lan, A. Xiao, X. Wang, and B. Liang, "Putn: A plane-fitting based uneven terrain navigation framework," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7160–7166, 2022.

[14] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[15] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *ArXiv*, vol. abs/1509.01149, 2015.

[16] D. Lee, I. M. A. Nahrendra, M. Oh, B. Yu, and H. Myung, "Trg-planner: Traversal risk graph-based path planning in unstructured environments for safe and efficient navigation," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1736–1743, 2025.

[17] X. Cai, M. Everett, J. Fink, and J. P. How, "Risk-aware off-road navigation via a learned speed distribution map," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2931–2937, 2022.

[18] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation," *arXiv preprint arXiv:2103.02828*, 2021.

[19] S. Saravanan, A. Bains, C. P.C. Chanel, and D. Vivet, "Fit-slam 2: Efficient 3d exploration with fisher information and traversability-based adaptive roadmap," *Robotics and Autonomous Systems*, vol. 194, p. 105188, 2025.