IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Real-Time Multi-Level Terrain-Aware Path Planning for Ground Mobile Robots in Large-Scale Rough Terrains

Yuxiang Li, Kun Chen, Yifei Wang, Weifan Zhang, Jiancheng Wang, Haoyao Chen, *Senior Member, IEEE*, and Yunhui Liu, *Fellow, IEEE*

***Abstract*—Autonomous ground mobile robots rely on their configuration characteristics to prevent tip-overs and collisions, ensuring safe navigation in complex environments. However, complex configurations with specially designed links and joints produce a higher-dimensional workspace and bring significant challenges for path planning, especially in large-scale rough terrains. To address this, we propose a real-time multi-level terrain-aware path planning framework that integrates different levels of terrain awareness into the global and local layers. An implicit map representation is introduced at the global layer to enable efficient terrain analysis and path planning, while an iterative geometric evaluation is designed at the local layer to estimate configuration stability and improve path smoothness. By sharing the global layer information with the local layer, the framework enhances path planning efficiency and adaptability in complex environments. Its modular design supports diverse robot configurations and pathfinding algorithms, enabling effective autonomous navigation in large-scale 3D terrains with online or offline maps. Simulations and real-world experiments demonstrated that our approach outperforms state-of-the-arts across diverse environments, including uneven terrains, multi-layered structures, and complex debris fields. The results highlighted that our approach provides faster and safer path planning, more accurate and robust configuration-stability estimation, and higher success rates in traversing complex 3D environments.**

***Index Terms*—Terrain analysis, configuration-stability estimation, path planning, rough terrain.**

I. Introduction

AUTONOMOUS ground mobile robots are critical in a wide range of applications, from exploration and transportation to search and rescue missions [1]. These robots often operate in complex and unpredictable environments, such as rugged landscapes, disaster zones, and urban rubble fields. The terrain in these environments can vary greatly in terms of slope, texture, and structure. Ensuring safe, stable, and efficient navigation is a key challenge. Therefore, ground mobile robots are designed with different chassis configurations, including wheeled [2], tracked [3], and legged [4] types, each using specific mechanical structures to enhance mobility. However, path planning that accounts for a robot's complex physical constraints in traversing challenging environments remains an open problem in robotics.

Traditional path planning approaches [5], [6], [7] primarily focus on obstacle avoidance and trajectory smoothness, generally assuming a 2D workspace. However, advancements in robotic mechanical structures [2], [3], [4] feature robots with complex configurations, including auxiliary moving parts, such as flippers, resulting in a higher-dimensional workspace. Hierarchical approaches [8], [9] decompose path planning into two stages: global planning and local planning. Global planning provides a coarse navigation path over a large area [10], but global paths often overlook detailed terrain changes, potentially introducing navigation hazards. Local planning addresses this issue by using high-resolution maps for precise motion planning, generating collision-free trajectories that adhere to kinematic constraints and minimize driving risks [11]. Despite some improvements, these approaches [8], [9], [10], [11] still struggle in complex environments due to limited terrain awareness.

Recent studies [12], [13], [14], [15] integrate terrain awareness into path planning to improve adaptability on uneven terrains using techniques like terrain analysis and configuration-stability estimation. Terrain analysis [12] evaluates geometric metrics, such as slope and roughness, and weights them to measure traversability based on the robot's capabilities. Configuration-stability estimation [14] determines whether a robot's physical configuration—such as its orientation, position, and joint states—maintains stable contact with the terrain. This helps identify the risk of tipping over or losing balance in challenging environments. Although these researchers [12], [13], [14], [15] have provided interesting solutions for improving robots' traversal capability, their approaches still face limitations. Accurate terrain analysis depends on high-resolution map representations, but large-scale environments make accurate terrain mapping and real-time assessment difficult [16]. Moreover, terrain analysis provides only a coarse assessment, which is less effective on rough terrains with complex robot configurations [17]. Additionally, the robot's high-dimensional workspace poses challenges for motion planning, especially in maintaining kinematic feasibility and adhering to physical constraints [18]. It also complicates the reliable estimation of

This work was supported in part by the National Natural Science Foundation of China under Grants U21A20119 and U1713206, and in part by the Shenzhen Science and Innovation Committee Funds under Grant RCJC20231211090050082, Grant SZXJP20230703093206015, and Grant JCYJ20241202123714019. (Corresponding author: Haoyao Chen)

Yuxiang Li, Kun Chen, Yifei Wang, Weifan Zhang, Jiancheng Wang, and Haoyao Chen are with the School of Robotics and Advanced Manufacture, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China, e-mail: ({19B953004, 210330122, 21B353013, 23S153018, 22B353018}@stu.hit.edu.cn, hychen5@hit.edu.cn).

Yunhui Liu is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hongkong 999077, China, e-mail: (yhliu@cuhk.edu.hk).

configuration stability, which further hinders efficient motion planning.

This paper proposes a real-time multi-level terrain-aware path planning framework to address the above challenges for mobile robots' navigation on large-scale rough terrains. Unlike traditional hierarchical frameworks [8], [9] that focus on global topology guidance and local elevation analysis, our framework incorporates terrain information at both global and local layers. First, an implicit map, which efficiently balances detailed and large-scale terrain representation, is introduced in the global layer to facilitate terrain analysis and global path planning. Terrain information is computed and stored online during map integration. Next, an iterative geometric approach is incorporated in the local layer to estimate robot configuration stability and produce smooth paths. By sharing the global layer information with the local layer, the framework enhances path planning efficiency and adaptability in complex environments. Finally, the overall framework enhances path planning efficiency and adaptability in complex environments. The contributions of this paper are as follows:

1) An implicit map for terrain analysis is proposed, incorporating terrain metrics such as roughness, slope, and sparsity. Its sub-resolution accuracy ensures reliable terrain awareness in large-scale complex environments.
2) Terrain analysis is presented with the terrain metrics derived from the implicit map, followed by traversal risk assessment and traversability estimation. The resulting traversable voxel set enables more effective global planning.
3) A novel iterative geometric approach is designed in local planning for more robust configuration-stability estimation, resulting in higher success rates than state-of-the-art approaches when navigating challenging terrains.
4) The framework is highly modular and tailored for path planning in large-scale rough terrains. It supports different robot configurations, various pathfinding algorithms, and both online and offline maps. The code will be open-sourced on GitHub[1].

II. Related Work

A. Terrain Analysis

Explicit map-based approaches, such as elevation map [19], volumetric map [20], and mesh [21], are commonly used in terrain-aware path planning. Elevation map-based approaches [19], [22] assess terrain traversability with geometric metrics and incorporate pathfinding methods, such as rapidly-exploring random tree (RRT) [23]. However, elevation maps fail to model multi-layered structures like tunnels and stairs, restricting their applicability in complex terrains. The plane-fitting-based navigation framework [13] improves 3D path planning by constructing a volumetric map and analyzing terrain metrics for sampled nodes during RRT expansion. However, the voxelized map inevitably decreases the accuracy of terrain representation, and simplifying the sampling to SE(2) space limits its applicability to multi-layered environments. Meshes are utilized in [12], [21] to represent 3D rough terrains and assess traversability, using a graph-based algorithm [24] for pathfinding. However, the computational expense of mesh triangulating makes it suitable only for small-scale terrains and local path planning. In contrast, our approach integrates an implicit representation within a volumetric map to balance accuracy and efficiency.

Implicit map-based approaches—such as surfel-based map [16], normal distribution transform (NDT)-based map [25], and Gaussian-based map [26], [27]—offer significant advantages in capturing fine terrain details. Surfel-based navigation [16] utilizes surfel maps constructed from dense point clouds to evaluate traversability for uneven terrains and demonstrates superior terrain representation. The normal distribution transform (NDT)-based method [25] offers a promising solution for terrain representation through incremental map integration, along with a support vector machine trained to classify voxels as binary traversability. Gaussian-based map representations, such as Gaussian process regression [26] and Gaussian mixture models [27], are popular for terrain analysis. However, they require supervised training for traversability estimation and introduce additional computational overhead, which limits their effectiveness in real-time, large-scale mapping and planning. Inspired by the implicit map-based approach [25], we take a step further by fusing neighboring distributions to formulate terrain metrics. Instead of training a classifier for traversability estimation, we apply carefully designed weighting and thresholding of terrain metrics, explicitly accounting for robot capabilities while preserving the interpretability of terrain analysis.

Collision detection and traversal risk assessment are critical components of safe navigation. As for 2.5D elevation maps [19], [22], collision detection can be straightforwardly achieved by setting a step height threshold. In terms of traversal risk estimation, Dixit *et al.* [22] propose a probabilistic model incorporating conditional risk values to assess risk levels. However, such approaches are inherently limited by the dimensional constraint of elevation maps and are not applicable in complex 3D environments. The workspace of ground mobile robots is a constrained 3D space, where a critical constraint is that the robot's configuration must lie on the terrain surface. This requirement makes the Euclidean signed distance field (ESDF) map [28], commonly used for unmanned aerial vehicles, unsuitable for ground mobile robots in 3D terrains. As for 3D traversability maps introduced by [13], [16], [21], collision detection can be performed by applying a traversability threshold to determine potential traversal risks. However, this manner is simplistic and struggles to handle diverse conditions, such as positive and negative obstacles, overhangs, and multi-layered structures. To address these challenges, we propose calculating terrain complexity based on the implicit map, followed by a comprehensive analysis of traversal risks, including terrain risk, collision risk, and falling risk. Our method supports online computation during map integration, thereby facilitating real-time performance of path planning.

[1]https://github.com/HITSZ-NRSL/terrain-aware-planning.git

B. Configuration-Stability Estimation

Learning-based approaches [3], [4], [29], [30] have shown great potential for various robotic tasks, benefiting from minimal reliance on task-specific assumptions. However, reinforcement learning-based approaches [3], [29] often require carefully crafted rewards and extensive training to mitigate undesirable behaviors. Wellhausen *et al.* [4] and Guzzi *et al.* [30] employ labeled elevation maps to train deep convolutional neural networks that infer collisions during local motion and estimate the traversability of sampled nodes. By contrast, we propose a geometry-based approach that intuitively analyzes the contact between a robot's chassis and the terrain, offering greater interpretability and generalizability in understanding how these interactions influence the robot's actions.

Geometry-based approaches [31], [32], [33], [34] are practical for estimating configuration stability, where stability margins [31] depend on two key factors: the robot's center of mass (CoM) and the support polygon. Kanoulas *et al.* [32] and Kita *et al.* [33] analyze the contact between the robot's feet and the ground, utilizing the zero-moment point (ZMP) criterion [34] to evaluate configuration stability for legged robots. This criterion is also applicable to wheeled and tracked robots [35], [36]. The key difference is that legged robots typically focus on planning footholds and gaits, while wheeled and tracked robots tend to optimize terrain contact areas to maintain stability. Additionally, Norouzi *et al.* [31] use the open dynamics engine to simulate reconfigurable robots and model their behavior when falling from a given point. Fabian *et al.* [36] and Yuan *et al.* [37] simplify the robot model to skeletons by abstracting the vehicle's morphology, thereby alleviating the computational overhead. Similarly, Yuan *et al.* [14] estimate configuration stability using forward simulations to analyze contact points. Chen *et al.* [15] further abstract the robot model into three segmented lines and formulate a mixed-integer nonlinear programming problem with a derived analytical solution to speed up the problem-solving process. Although these geometry-based approaches [14], [15], [36], [37] significantly improve the computational efficiency, the simplifications often reduce reliability. Incorrectly extracted contact points can introduce uncertainty, impacting the overall accuracy of the configuration-stability estimation. To address these limitations, we propose a novel configuration-stability estimation technique that combines carefully abstracted robot models with rational iterative steps for geometric evaluation. Our approach enables more accurate modeling of robot-terrain interaction while maintaining computational efficiency.

III. System Overview

The proposed approach addresses path planning for ground mobile robots navigating large-scale complex terrains. The overall framework consists of two primary components: the global and local layers (see Fig. 1). Detailed pipelines of the global and local layers are shown in Figs. 2 and 6, respectively. When no prior map is provided, the robot uses onboard sensors to capture sensing data for real-time localization and mapping. The robot's estimated position and pose in the world coordinate system and the undistorted point clouds are used to update both the local and global maps online. When a prior map is available, it is loaded into the global map. The robot then performs frame-to-map registration with the current sensing data for localization and extracts a relevant local area from the prior map to update the local map.

The global layer incrementally integrates frames from the localization and mapping module into the implicit map, where terrain analysis, traversal risk assessment, and traversability estimation are performed (see Sec. IV). The traversable voxel set is maintained to speed up global path planning. Once a user-defined goal point is set, the nearest traversable voxel to the goal is selected as the navigation target, triggering global path planning. A global path is then generated using the traversability map via a graph- or sampling-based pathfinding algorithm. Although the global path accounts for traversability and avoids potential risks, the global map is too coarse to ensure kinematic feasibility. Therefore, the local layer must concurrently maintain a high-resolution map to support local motion planning guided by the global path.

The local layer integrates the iterative geometric approach designed to estimate the robot's configuration stability by simulating the robot's falling behavior under gravity (see Sec. V). The configuration-stability estimation is then incorporated into a pathfinding algorithm for local path planning. Terrain information from the global layer is shared with the local layer to enhance calculation efficiency. Consequently, the local path planning is able to generate smooth paths online, adhering to the kinematic constraints of the ground mobile robot. The generated local path is then exported to the base controller for trajectory tracking. As the robot moves and gathers new observations, global and local paths are continuously re-planned based on the updated map until the goal is reached.

IV. Global Map Representation and Path Planning

The global layer comprises global map representation and path planning (see Fig. 2). Inspired by mainstream approaches for navigating rough terrains [13], [17], we integrate several terrain metrics into the global map for terrain analysis. To balance real-time map integration with detailed terrain representation, an implicit representation is introduced into the volumetric map, along with new formulations for terrain metrics under the representation. These metrics are combined to represent terrain complexity and assess terrain risk. Based on the calculated terrain risk, collision and falling risks are further involved in estimating traversal risk and traversability (i.e., traversal cost). Finally, connected traversable voxels are grouped into a set, facilitating efficient global path planning in large-scale and complex environments.

A. Global Implicit Mapping

The global map representation is based on the octree structure [20] to construct a 3D volumetric map, with the leaf node storing information within the corresponding voxel. While low resolution is typically employed to cover large-scale areas and ensure real-time map updates and path planning, it inevitably sacrifices terrain details. Inspired by [25], we introduce NDT, a 3D Gaussian distribution, as the implicit

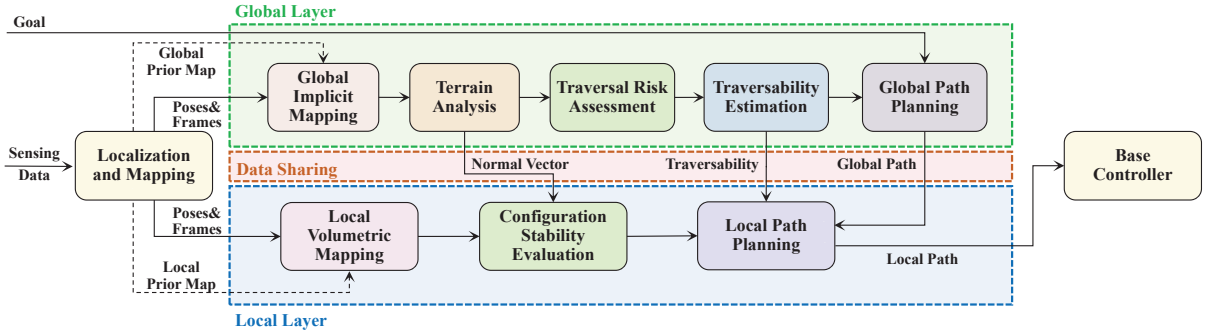


Fig. 1. The proposed multi-level terrain-aware path planning framework for large-scale 3D rough terrains. The framework consists of two main components: a layer focused on terrain analysis and global path planning in large-scale environments, and a local layer dedicated to estimating configuration stability and generating feasible paths to navigate rough terrains.

representation within the volumetric map. This novel approach not only classifies voxels as *free*, *occupied*, or *unknown* but also leverages 3D Gaussian distributions to encode geometric information. The key insight is to estimate the Gaussian distribution—specifically, the mean and covariance—for each voxel based on accumulated measurement points. By doing this, the implicit map provides a richer and more detailed terrain representation than traditional volumetric or elevation maps at the same resolution. Rather than classifying terrain into binary traversability [25], we take a step further by leveraging the sub-resolution representation of implicit mapping to achieve comparable accuracy in terrain metrics at a lower resolution. This enables the implicit map to provide more efficient terrain analysis and global planning, especially in large-scale and complex environments.

When updating the implicit map, we use the ray-casting technique to mark voxels traversed by the rays as *free* and those hit by the rays as *occupied*. Then, the Gaussian distribution parameters of the updated *occupied* voxels are recalculated. When a new measurement point $\boldsymbol{p} = [x_i, y_i, z_i]^T$ falls in the voxel v, the mean vector $\boldsymbol{\mu}_v$ and the covariance matrix $\boldsymbol{\Sigma}_v$ are incrementally updated as follows:

$$\begin{cases} \boldsymbol{\mu}_v = \boldsymbol{\mu}_{v_{old}} + \frac{1}{1+k_{v_{old}}}\left(\boldsymbol{p} - \boldsymbol{\mu}_{v_{old}}\right) \\ \boldsymbol{\Sigma}_v = \frac{k_{v_{old}}}{1+k_{v_{old}}}\left(\boldsymbol{\Sigma}_{old} + \frac{1}{1+k_{v_{old}}}\left(\boldsymbol{p} - \boldsymbol{\mu}_{v_{old}}\right)\left(\boldsymbol{p} - \boldsymbol{\mu}_{v_{old}}\right)^T\right), \end{cases} \tag{1}$$

where k_v represents the number of points that have fallen into the current voxel v , and the subscript old indicates the parameters before inserting the new point. The distribution parameters are continuously updated with new observations without storing all measurement points. Hence, the proposed map representation preserves the octree's efficiency while capturing more terrain details. Furthermore, a saturation threshold K_{crit} is employed in the incremental map integration by restricting updates to voxels with point count exceeding it, such that terrain analysis can focus only on voxels where the distribution parameters have changed.

Due to measurement noise and voxel discretization, a single voxel's distribution often cannot accurately represent local terrain. More comprehensive terrain analysis requires merging distributions of neighboring voxels. Based on the property of

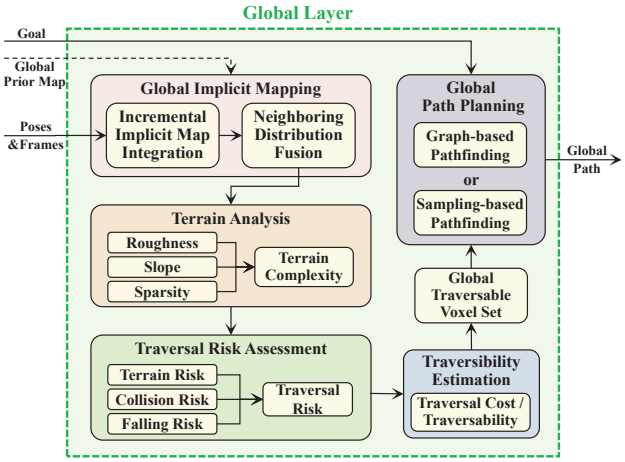


Fig. 2. The global layer pipeline involves implicit mapping, terrain analysis, traversal risk assessment, traversability estimation, and global path planning.

joint Gaussian distributions [38], the distributions of multiple voxels can be fused as follows:

$$\begin{cases} \boldsymbol{u}_{v_1 \oplus v_2} = \frac{1}{k_{v_1}+k_{v_2}}(k_{v_1} \cdot \boldsymbol{u}_{v_1} + k_{v_2} \cdot \boldsymbol{u}_{v_2}) \\ \boldsymbol{\Sigma}_{v_1 \oplus v_2} = \boldsymbol{\Sigma}_{v_1} + \boldsymbol{\Sigma}_{v_2} + \frac{k_{v_1} \cdot k_{v_2}}{k_{v_1}+k_{v_2}}(\boldsymbol{\mu}_{v_1} - \boldsymbol{\mu}_{v_2})(\boldsymbol{\mu}_{v_1} - \boldsymbol{\mu}_{v_2})^T. \end{cases} \tag{2}$$

B. Terrain Analysis

Terrain analysis is crucial for ground mobile robot navigation, impacting terrain traversability estimation and the robot's reactivity to terrain changes. Existing methods use geometric metrics like roughness and slope for terrain analysis, relying on dense point clouds or volumetric maps. Using dense point clouds to assess terrain is computationally intensive and impractical for large-scale environments. Low-resolution volumetric maps lead to efficient assessment but lose detailed terrain information and cause traversal risks. Herein, the implicit representation is employed to enhance terrain analysis by computing geometric metrics such as roughness, slope, and sparsity, which are then combined to estimate terrain complexity, risks, and traversability (see Fig. 3).

First, singular value decomposition (SVD) is performed on the fused covariance matrix $\boldsymbol{\Sigma}$ in formula (2) to obtain the

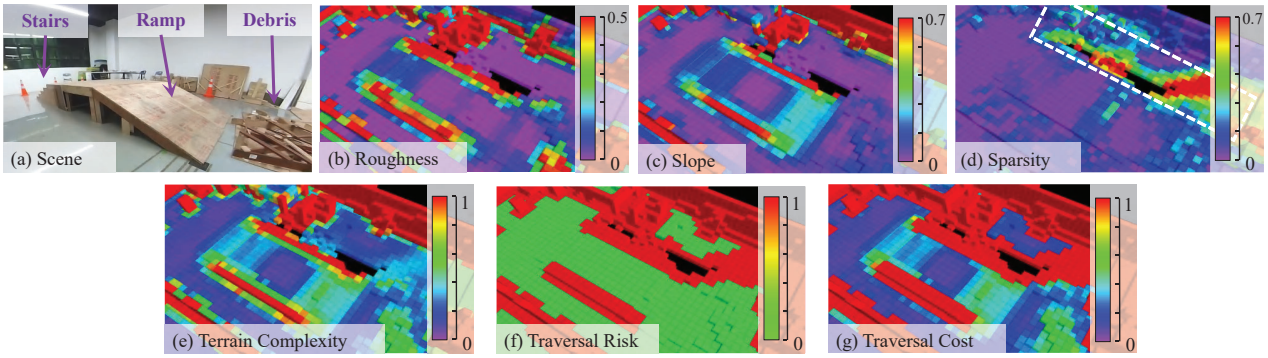


Fig. 3. Terrain analysis of a field scene (a) featuring typical terrains, such as stairs, ramp, and debris. Results for terrain metrics, such as (b) roughness, (c) slope, and (d) sparsity. Outcomes of (e) terrain complexity, (f) traversal risk, and (g) traversal cost.

eigenvectors ($\boldsymbol{n}_1$, $\boldsymbol{n}_2$, $\boldsymbol{n}_3$) and the corresponding eigenvalues (λ_1, λ_2, λ_3) arranged in ascending order. Since eigenvalues can depict the shape of Gaussian distribution, roughness is defined as follows:

$$r(v) = 1 - \frac{\lambda_2 - \lambda_1}{\lambda_2 + \lambda_1}. \tag{3}$$

If the smallest eigenvalue is noticeably smaller than the other two, the distribution approximates a plane, resulting in a low roughness value. Conversely, if the smallest eigenvalue is relatively large, it indicates rougher terrain with higher terrain complexity. As shown in Fig. 3(b) , the stairs, debris, and object edges exhibit evident roughness values.

The slope metric also impacts the difficulty of terrain traversal, with steeper inclination increasing the likelihood of vehicle sliding or overturning. As illustrated in Fig. 3(c) , the field scene features some inclined structures. To determine the slope, it is assumed that the z-axis of the map is aligned with the direction opposite to gravity during initialization. The normal vector of the x-y plane is used as the reference vector $\boldsymbol{n}_z = [0, 0, 1]^T$, while the eigenvector $\boldsymbol{n}_1$ corresponding to the smallest eigenvalue represents the normal vector of the terrain surface. The slope is then defined as follows:

$$s(v) = \arccos\left(\frac{\boldsymbol{n}_1 \cdot \boldsymbol{n}_z}{|\boldsymbol{n}_1| \cdot |\boldsymbol{n}_z|}\right). \tag{4}$$

The sparsity metric is measured by the average point density of neighboring voxels, highlighting the areas with insufficient observations and implying traversability uncertainty. Fig. 3(d) illustrates an under-scanned area within the white dashed box, posing potential risk as it is unclear if a cliff exists. The point density is defined as the ratio of the point number in the voxel to the saturation threshold K_{crit}. However, self-occlusions of geometric structures may produce biased density estimations (see Fig. 4). The view directions (indicated by the dashed arrows) can observe the neighboring voxels (colored green and orange) on the front side. However, the shadowed voxels on the back side remain unobserved due to self-occlusion. Using all neighboring voxels within the dashed circle to calculate average point density leads to an underestimation, making well-observed front-side voxels appear high sparsity. Inspired by the fronto-parallel window model [39], we estimate sparsity based only on visible voxels along fronto-parallel view directions

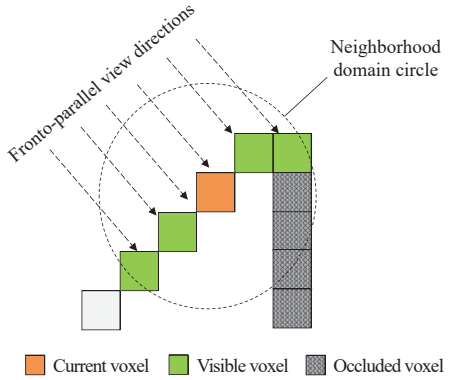


Fig. 4. Fronto-parallel window model. The orange and green voxels represent those visible along the fronto-parallel view directions within the neighborhood, while the shadowed voxels are self-occluded.

opposite to the current voxel's normal vector (see Fig. 4). The visible voxel set is denoted as $\{\boldsymbol{v}_f\}, f = 1, 2, ..., N_f$, where N_f is the number of visible voxels. Then, terrain sparsity is defined as follows:

$$d(v) = 1 - \frac{1}{N_f}\sum_{f=1}^{N_f}\frac{k_{v_f}}{K_{crit}}, \tag{5}$$

where k_{v_f} represents the point count in voxel v_f, and k_{v_f}/K_{crit} denotes the point density.

Subsequently, the terrain complexity $\tau(v)$ of the voxel v is defined as the weighted sum of roughness, slope, and sparsity:

$$\tau(v) = \alpha_r\frac{r(v)}{r_{crit}} + \alpha_s\frac{s(v)}{s_{crit}} + \alpha_d\frac{d(v)}{d_{crit}}, \tag{6}$$

where weights α_s, α_r and α_d are summed to 1, and s_{crit}, r_{crit} and d_{crit} are thresholds. Worse terrain conditions lead to increased terrain complexity. As shown in Fig. 3(e) , the terrain complexity highlights the hazardous regions detected by the terrain metrics. Since terrain complexity is closely related to voxel size and robot dimensions, the voxel size should match the minimum obstacle size that impedes the robot's movement to ensure sensitivity to terrain changes. The robot's radius is used as the fusion radius of the joint Gaussian distribution of neighboring voxels to derive the terrain metrics. Furthermore,

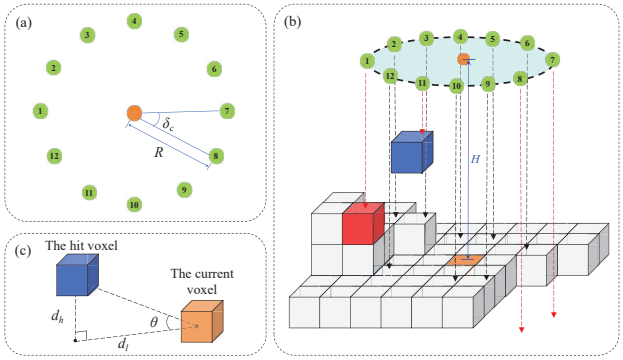


Fig. 5. Traversal risk assessment. (a) Circular checkpoints (green) around the robot's current position (orange). (b) Ray-casting (dashed arrows) from the elevated checkpoints to detect falling, terrain, and collision risks. Black arrows indicate low risks, while red arrows indicate high risks. (c) Collision risk is determined by the tilt angle associated with the current and hit voxels.

the weights and thresholds for estimating terrain complexity in formula (6) depend on robot capabilities. For example, higher weights for roughness and slope should be set for a robot with limited traversal ability. Without loss of generality, parameters can be set empirically or learned using methods like [25].

C. Traversal Risk Assessment and Traversability Estimation

Path planning for ground mobile robots generally relies on collision detection and other constraints, but the extensive map retrieval operations required hinder its efficiency. Point cloud segmentation separating ground and potential obstacles is commonly used for constructing 2D ESDF or 2.5D elevation maps to speed up path planning. Nevertheless, accurate segmentation in complex scenes remains difficult, impacting the effectiveness of path generation. To this end, we further propose a traversal risk assessment approach for 3D terrain. Traversal risks are categorized into three types: terrain risk, collision risk, and falling risk, all of which collectively determine traversal cost (i.e., traversability). Terrain risk depends on terrain complexity, with areas of high complexity typically exhibiting increased terrain risk. Collision risk occurs when the robot approaches obstacles and overhangs, while falling risk is associated with proximity to cliffs and steep ramps.

As shown in Fig. 5(a) , traversal risk is assessed for the current position (in orange) using the checkpoints (in green) generated based on the robot's dimensions—radius R and height H. The checkpoints are distributed in the x-y plane, centered at the current voxel with radius R. The angular interval δ_c between checkpoints ensures that the separation between adjacent points matches the voxel size. As depicted in Fig. 5(b) , the checkpoints are then elevated by the robot height H above the current voxel. Rays are cast from each checkpoint toward the ground, and the voxels hit by the rays are inspected. If the terrain complexity of the voxel hit by the ray exceeds the threshold τ_{crit}, the current voxel is deemed near a hazardous region with terrain risk. As shown in Fig. 5(b) , ray 1 hits a voxel with high complexity, implying a terrain risk. If the inclination of the occupied voxel that the ray hits surpasses the threshold s_{crit}, i.e., the tilt angle is too large, there is a collision risk (see Fig. 5(c)). The inclination is defined with the tilt angle $\theta = \operatorname{atan}(d_h, d_l)$, where d_h is the height difference between the hit voxel and current voxel, and d_l is the horizontal distance between them. As shown in Fig. 5(b) , ray 11 hits a voxel (in blue) on the overhang object, resulting in an excessive inclination and thus a collision risk. If some rays, like rays 7 and 8, do not hit any occupied voxel, it implies the presence of suspended areas around the robot, such as cliffs, which pose a falling risk.

With the above process, we can determine terrain risk $\mathcal{R}_\tau(v)$, collision risk $\mathcal{R}_c(v)$, and falling risk $\mathcal{R}_f(v)$ as boolean variables of a voxel v. The traversal risk $\mathcal{R}_s(v)$ is then represented with a boolean variable:

$$\mathcal{R}_s(v) = \mathcal{R}_\tau(v) \vee \mathcal{R}_c(v) \vee \mathcal{R}_f(v), \tag{7}$$

where $\vee$ denotes the logical "*or*" operation. Voxels with high traversal risk are categorized as untraversable voxels, while those with low traversal risk are deemed traversable. Subsequently, both traversal risk and terrain complexity are factored into the traversal cost $\mathcal{T}(v)$ as follows:

$$\mathcal{T}(v) = \begin{cases} \tau(v), & \text{if } \mathcal{R}_s(v) = false \\ +\infty, & \text{otherwise} \end{cases}. \tag{8}$$

Traversal cost serves as an indicator of traversability and is stored in the leaf nodes during online map integration. As illustrated in Fig. 3(f) , the sharp edges of the terrain and the boundaries of the map holes are classified as untraversable areas (in red) with high traversal risks. Accordingly, these regions exhibit high traversal costs, as shown in Fig. 3(g) . The traversability map is further utilized as an essential component of global path planning, enabling safe and effective navigation in complex 3D environments.

D. Global Path Planning

The traversability map can be integrated into global planning with various pathfinding algorithms, such as sampling-based methods [6], [23] and graph-based methods [5], [24]. For sampling-based methods, traversability should be checked for not only the voxels at the sampled nodes but also those along the edges between adjacent nodes. This leads to extensive voxel retrieval and time consumption. However, in many time-critical situations, global planning must quickly identify a path with minimal traversal cost. Additionally, most existing path planning techniques for ground mobile robots depend on elevation maps and operate in SE(2) space, which struggles with height ambiguities and is inadequate for multi-layered 3D environments.

To solve these issues, we design a terrain-aware global path planning approach based on the graph search algorithm A* [24]. Novel strategies are designed for both node expansion and cost evaluation, addressing the limitations of existing methods and enabling efficient planning with our traversability map. Specifically, 3D nodes are expanded by assessing the neighboring voxels of the current node, with traversable voxels being selected as candidate nodes. Node costs are further evaluated with two primary factors: cost-so-far and cost-to-go.

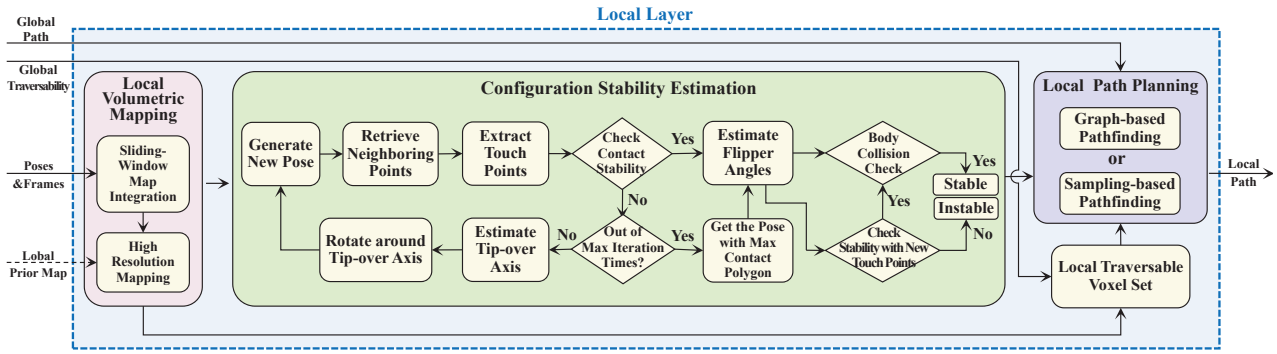


Fig. 6. Pipeline of the local layer consists of three main components: local volumetric mapping, configuration-stability estimation, and local path planning.

The traversal cost is integrated into the cost-so-far function for terrain-aware path planning:

$$g(v_{succ}) = g(v) + \alpha_l \cdot dist(v, v_{succ}) + \alpha_\tau \cdot \mathcal{T}(v_{succ}), \quad (9)$$

where $dist(v, v_{succ})$ denotes the Euclidean distance between the two voxels, and $\mathcal{T}(v_{succ})$ is the traversal cost defined in formula (8). The function (9) accumulates the transition cost from the current voxel v to the next voxel v_{succ}, with α_l and α_τ serving as weights. Additionally, the cost-to-go is estimated with the Euclidean distance from the current node to the goal, thereby prioritizing nodes that directly lead toward it.

To further accelerate the path planning process, we introduce region growing for traversable voxels and manage a connected voxel set using the union-find algorithm [40]. The traversable voxel set is updated following map integration. Node expansion is then limited to the traversable voxel set, excluding free, unknown, and untraversable voxels. Therefore, the search space is reduced to a constrained 3D space. Unnecessary collision checks and constraint evaluations for ineffective nodes can be saved, further enhancing planning efficiency (see Sec. V.A). Consequently, our approach overcomes the SE(2) limitation of existing methods, enabling effective path planning in complex 3D environments like multi-layered structures (refer to Sec. VI.A.2.).

V. Iterative Geometric Evaluation-based Local Path Planning

Global paths are planned using traversability information from the coarse global map, providing only waypoints with headings rather than precise robot configurations. To generate a kinematically feasible path, further local planning is required. Herein, we design an iterative geometric evaluation-based local path planning approach for ground mobile robots, such as tracked and wheeled vehicles. The pipeline of the local layer is illustrated in Fig. 6, where the configuration-stability estimation is based on the local volumetric map and then integrated into local planning to generate smooth paths.

A. Local Map Representation

Local planning relies on a detailed terrain representation to estimate configuration stability. Since updating a dense map is time-consuming, a robot-centric map using a high resolution is maintained in the local layer. The local map is confined within a sliding window that moves with the robot. Herein, the resolution of the local map is ξ_l, and the size of the local map is $\boldsymbol{S}_l$. Newly measured points within the sliding window are integrated into the map, while the map voxels outside the window are deleted to keep update efficiency. Similar to the global layer, a set of traversable voxels within the local map is maintained to restrict the search space for motion planning. The local set is maintained by querying traversability information shared by the global layer and is organized using a k-dimensional tree to accelerate retrieval operations.

B. Configuration-Stability Estimation for Tracked Vehicle

Configuration-stability estimation refers to the process of identifying configurations that allow a robot to maintain stable contact with the terrain at a given position and orientation. However, accurate configuration-stability estimation requires a detailed robot model and precise terrain representation, both of which make real-time computation challenging. Traditional approaches [14], [15], [37] simplify the robot model to reduce computational overhead, which often compromises reliability. To this end, we propose a novel configuration-stability estimation technique that combines carefully abstracted robot models with rational iterative steps for geometric evaluation. This ensures accuracy and robustness without introducing significant computational overhead. Furthermore, our approach is scalable across various robot configurations, including wheeled, tracked, and articulated tracked vehicles.

Given the exceptional mobility and load capacity of articulated tracked robots, our approach adopts this robot as the primary case study, with extensions to wheeled robots.

1) Abstract Robot Model: The structure of the articulated tracked robot is shown in Fig. 7(a) , comprising the body, main tracks, and flippers (also known as sub-tracks). The analysis of forces on the robot located on a ramp terrain is illustrated in Fig. 7(b) , where $\boldsymbol{P}_c$ is the equivalent CoM, and $\boldsymbol{P}_o$ denotes the robot position. Assuming quasi-static dynamics, the forces are balanced among ground support $\boldsymbol{F}_N$, friction $\boldsymbol{F}_f$, and gravity $\boldsymbol{g}$. Ground friction is essential for vehicle movement and varies with different terrain materials, such as soil, gravel, or snow. If the friction force is insufficient to counteract the component of the gravity $\boldsymbol{g}$ along the slope, instability issues such as slippage or tip-over may occur. However, predicting

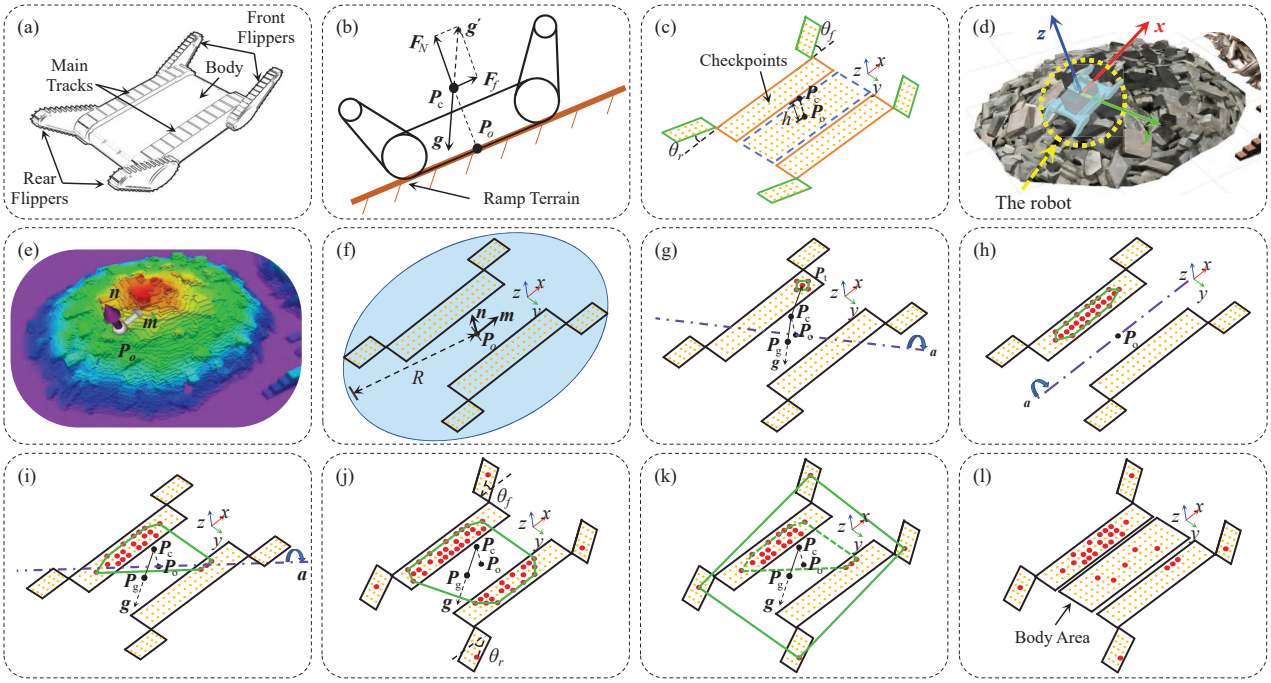


Fig. 7. The process of configuration-stability estimation for an articulated tracked robot. (a) Components of the robot. (b) Force analysis of the robot under the quasi-static dynamics assumption. (c) Abstracted robot model with flippers (green boxes), tracks (orange boxes), and body (blue dashed box). (d) Robot on rough terrain. (e) Local map of the terrain. (f) The initial robot model. (g) Single-point contact with a few track contact points (in red). (h) Single-side contact with a narrow support polygon (in green). (i) Insufficient contact with the projected CoM point $\boldsymbol{P}_g$ lying outside the support polygon. (j) Stable contact with the projected CoM point lying within the support polygon. The contact point (in red) in each flipper region forms the maximum angle relative to the support polygon. (k) Configuration-stability estimation with flipper contact points in an unstable pose. The support polygon of the main tracks is shown with a green dashed polygon, while that of both main tracks and flippers is shown with a green solid polygon. (l) Checkpoints for body collision detection, where red dots within the body area indicate collisions.

terrain materials and estimating friction coefficients are beyond the scope of this paper. Instead, we focus on analyzing how body poses and flipper angles affect configuration stability.

Unlike existing approaches that abstract the tracks and sub-tracks into skeletons, we explicitly model the robot using the bottom surfaces of the tracks and sub-tracks. This detailed modeling provides a more precise representation of the robot's interaction with the terrain. As illustrated in Fig. 7(c) , the model includes two main tracks (outlined in orange box) and four sub-tracks (outlined in green box), along with the robot's body (outlined in blue dashed box) for collision detection under the body. The robot's configuration is defined by the base position $\boldsymbol{P}_o$ in Cartesian coordinates $[x, y, z]$, the base orientation $\boldsymbol{q}$ represented with Euler angles $[\gamma, \rho, \phi]$, and the flipper angles $[\theta_f, \theta_r]$ with subscripts "f" and "r" denoting the front and rear flippers, respectively. The front flippers are assumed to be rigidly connected to the same axis and thus share the same angle, as do the rear flippers. The flippers provide two extra degrees of freedom, enhancing the robot's mobility but increasing the complexity of state estimation. Since the robot moves along the terrain surface, the base roll and pitch angles $[\rho, \phi]$ and flipper angles $[\theta_f, \theta_r]$ can be determined based on the terrain for a given target position $[x, y, z]$ and heading γ. This reduces the search space of configuration-stability estimation to the $[x, y, z, \gamma]$ dimensions.

To estimate the stability of a robot configuration, we use the support polygon formed by the contact between the robot and terrain, along with the ZMP criterion [34]. The support polygon is defined as the convex hull formed by the robot's ground contact points. The robot configuration is stable if the CoM projection along gravity lies within the support polygon. Since the flippers contribute only a tiny portion of the robot's total weight, their impact on the CoM can be neglected. To clarify the relationship between the CoM and the support polygon, we assume that the base position $\boldsymbol{P}_o$ is centered at the bottom plane of the base, and the CoM $\boldsymbol{P}_c$ is fixed above the base position $\boldsymbol{P}_o$ with a vertical separation of h (see Fig. 7(c)). The simplified robot model can be viewed as the projection of the robot on the ground. For contact and collision detection with the terrain, checkpoints are uniformly generated within the main track and flipper areas, as illustrated in Fig. 7(c) . The main tracks provide primary ground contact when operating in rough terrain, while the flippers only assist in buffering falling vibrations and preventing tip-overs. Therefore, we assume that the base pose is primarily determined by the interaction of the main tracks with the ground, with the flipper angles adjusted to provide additional support and stability. Along with the quasi-static assumption, both the base pose and the flipper angles can be estimated alternatively within their respective operational sub-spaces, which simplifies the problem by avoiding exhaustive evaluation across the entire workspace. Sequentially, the configuration-stability estimation involves two primary steps: predicting the base pose by analyzing the contact of the main tracks with the ground, and determining the flipper angles to

ensure proper support based on the estimated base pose.

2) Iterative Geometric Evaluation: An iterative geometric evaluation is designed to simulate the robot's falling behavior under gravity and estimate configuration stability. The process begins by specifying the robot's target position $\boldsymbol{P}_o$ and heading $\boldsymbol{m}$ on the local map (see Fig. 7(d)). Next, the initial base pose is calculated such that the normal vector of the robot model aligns with the terrain's normal vector $\boldsymbol{n}$, and the base's yaw angle matches the target heading $\boldsymbol{m}$ (see Fig. 7(e) and (f). Assuming that the local map is aligned with the global map, the z-axis of the local map also points in the opposite direction of gravity. The terrain's normal vector can be retrieved from the global map. Following this, neighboring points within the local map are extracted using the robot's radius R as the search range. These map points are then transformed into the robot's coordinate system to recognize contact points and support polygons.

Fig. 8 illustrates this process more clearly using a stair scene as a typical example. Given the target position and heading on a stair (see Fig. 8(a)), the neighbor point cloud is extracted and transformed into the robot's frame (see Fig. 8(b)). The transformed point cloud is subsequently used to generate an elevation map (see Fig. 8(c)). Valid checkpoints, where height values can be queried within the elevation map, are used to determine potential contact points. Among these valid checkpoints, the highest point is identified with height h_{max}. The checkpoints within the height range $h_{max} \pm \delta_h$ are deemed as contact points, where the tolerance δ_h is based on the map accuracy. Next, the support polygon (i.e., the convex hull) formed by the contact points is calculated using the QuickHull algorithm [41]. This efficient method recursively identifies the extreme points that define the boundary, dividing the problem into smaller subsets until the convex hull is fully determined. As demonstrated in Fig. 8(d) , the resulting support polygon in this case is approximately a quadrangle.

As for the rubble terrain shown in Fig. 7(d) , the support polygons can be more complex, with three typical cases illustrated in Fig. 7(g), (h) and (i). Recalling that the normal vector queried from the global map is estimated over a neighborhood domain, which only produces an initial base pose coarsely. Therefore, multiple iterations are further required to accurately estimate the robot's pose until it forms stable contact with the terrain. Subsequently, the CoM $\boldsymbol{P}_c$ is projected along the gravity $\boldsymbol{g}$ onto the support polygon plane, producing the projected point $\boldsymbol{P}_g$. According to ZMP criterion [34], the position relationship between the point $\boldsymbol{P}_g$ and the support polygon is used to determine the stability of the contact. Suppose the polygon has N_A vertices arranged in clockwise, with the vertex set $\mathcal{A} = \{\boldsymbol{A}_i\}$, where $i = 1, ..., N_A$. The corresponding edge set is $\mathcal{E} = \{\boldsymbol{e}_i\}$, where $i = 1, ..., N_A - 1$. If the cross product $\overrightarrow{\boldsymbol{P}_g \boldsymbol{A}_i} \times \boldsymbol{e}_i$ yields the same sign for every edge $\boldsymbol{e}_i \in \mathcal{E}$, the point $\boldsymbol{P}_g$ is inside the polygon. As illustrated in Fig. 7(j) , when the support polygon is sufficiently large such that the projected CoM $\boldsymbol{P}_g$ is within the polygon, the contact is considered stable. Conversely, if the polygon is small or the point $\boldsymbol{P}_g$ falls outside the polygon (i.e., it does not satisfy the ZMP criterion), the contact is deemed unstable. Then, estimating the potential rotation of an unstable configuration under gravity is required to determine the stability criterion for the next iteration. Unstable contact includes three cases:

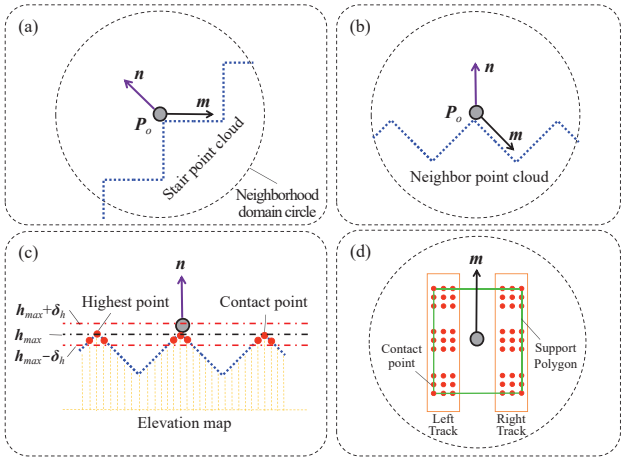


Fig. 8. Contact point identification and support polygon generation for the main tracks on a stair terrain, with the given position $\boldsymbol{P}_o$, heading $\boldsymbol{m}$, and the normal vector $\boldsymbol{n}$. (a) The point cloud (in blue) within the neighborhood domain (dashed circle) is extracted. (b) The neighbor point cloud is transformed into the robot frame, with the normal vector $\boldsymbol{n}$ aligned to the z-axis. (c) An elevation map (in orange) is generated using the neighbor point cloud, and contact points (in red) are identified as those within the maximum height range (red dash-dot lines). (d) Top view of the robot model (in orange) with the extracted contact points and support polygon (in green).

a) Single-point contact occurs when the contact points are concentrated in a small area (see Fig. 7(g)). The geometric center $\boldsymbol{P}_t$ of these contact points is used to determine the tip-over axis, denoted by $\boldsymbol{a} = \boldsymbol{g} \times \overrightarrow{\boldsymbol{P}_c \boldsymbol{P}_t}$.

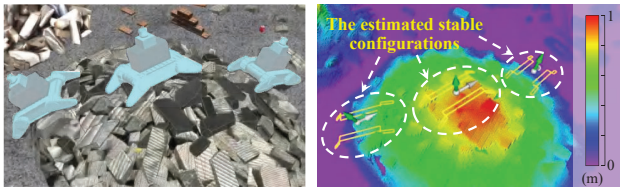
b) Single-side contact happens when contact points are confined to a narrow polygon on a single track (see Fig. 7(h)). SVD is performed on these points, with the eigenvector corresponding to the largest eigenvalue serving as the tip-over axis $\boldsymbol{a}$. The axis points forward if the left track is in contact and backward if the right track is in contact.

c) Insufficient contact arises when contact points form a relatively large polygon across both tracks, but the projected CoM $\boldsymbol{P}_g$ is outside the polygon (see Fig. 7(i)). The tip-over axis $\boldsymbol{a}$ is chosen from the polygon edges, minimizing the distance from the point $\boldsymbol{P}_g$ to this edge $\boldsymbol{e}_i$:

$$\boldsymbol{e}_m = \arg\min_{\boldsymbol{e}_i \in \mathcal{E}} dist(\boldsymbol{P}_g, \boldsymbol{e}_i), \tag{10}$$

where the distance is $dist(\boldsymbol{P}_g, \boldsymbol{e}_i) = \left|\overrightarrow{\boldsymbol{A}_i \boldsymbol{P}_g} \times \boldsymbol{e}_i\right| / |\boldsymbol{e}_i|$, and $\boldsymbol{A}_i$ is the start point of edge $\boldsymbol{e}_i$.

Since it is difficult to directly calculate the rotation angle required to establish a stable contact, we simulate the effect of gravity by rotating around axis $\boldsymbol{a}$ at an angle interval of δ_a. The direction of rotation is determined by the right-hand rule. After the rotation, contact stability of the new pose $\boldsymbol{q}'$ is re-evaluated with the above steps. This process is repeated until stable contact is achieved or the iteration number exceeds the threshold N_{crit}. If a stable base pose $\boldsymbol{q}_s$ is found, the contact points under flippers are checked to determine the contact angles (see Fig. 7(j)). If stable contact is not achieved after the iterative termination, the base pose $\boldsymbol{q}_m$ with the largest polygon generated during the iterations is selected.



(a) (b)

Fig. 9. Configuration-stability estimation results in simulated rough terrain. (a) The simulated rough terrain with the robot's configurations at different positions. (b) The local map of the scene is colored with elevation, and the feasible robot configurations are generated at the corresponding locations within the white dashed circles.

The flipper angles are then estimated to determine whether the configuration is stable with flipper support (see Fig. 7(k)).

3) Flipper Angle Estimation: With the current base pose, either $\boldsymbol{q}_s$ or $\boldsymbol{q}_m$, the neighboring points under the flippers are transformed into the robot's coordinate system to generate an elevation map. Height is queried from the elevation map for each checkpoint in the flipper regions (see Fig. 7(j)). The flipper contacts the terrain at a positive angle if a checkpoint is above the support polygon. Conversely, a negative angle is formed if a checkpoint is below the polygon. The maximum angle for each flipper is selected among all potential flipper angles formed by the checkpoints. With the obtained front and rear flipper angles θ_f and θ_r, a candidate configuration $[\boldsymbol{P}_o, \boldsymbol{q}_s, \theta_f, \theta_r]$ or $[\boldsymbol{P}_o, \boldsymbol{q}_m, \theta_f, \theta_r]$ is formed. If a new support polygon formed by the initially unstable base pose $\boldsymbol{q}_m$ using the checkpoints of both the main tracks and the flippers satisfies the ZMP criterion, the configuration $[\boldsymbol{P}_o, \boldsymbol{q}_m, \theta_f, \theta_r]$ is considered stable, as illustrated in Fig. 7(k) .

4) Body Collision Check: The robot's body is checked for collisions with the terrain to ensure sufficient clearance. Like detecting contact points for the main tracks, neighboring points under the body area are extracted to generate an elevation map (see Fig. 7(l)), and the checkpoint with maximum height h'_{max} in the body region is queried. Likewise, the maximum height h_{max} of the checkpoints in the main track regions is also obtained. If the body's maximum height h'_{max} exceeds the track's maximum height h_{max} by more than the safe ground clearance h_{crit}, i.e., $h'_{max} - h_{max} > h_{crit}$, the configuration is judged to collide with the terrain (see Fig. 7(l)). Otherwise, it is considered feasible.

Following these steps, the configuration-stability estimation for a goal position and heading is completed. As illustrated in Fig. 9, the simulation demonstrates the ability to accurately estimate the configuration stability in challenging terrains.

C. Configuration-Stability Estimation for Wheeled Vehicle

For demonstrating scalability, we apply our approach to a wheeled vehicle, as depicted in Fig. 10(a) . The robot model is abstracted with the wheel approximated as cylindrical (see Fig. 10(b)). Therefore, the arc surface within the central angle range $\delta_w < \delta_{crit}$ under each wheel is approximated as the contact surface. As shown in Fig. 10(b) , checkpoints (in orange) are evenly distributed across both the body and wheel areas. To simplify the calculation, the arc surface of each wheel is projected onto the ground, creating a 2D rectangle (red dashed box). The projected points within the rectangular area correspond to the checkpoints on the arc surface. The height of a projected checkpoint $\boldsymbol{P}_k$ can be obtained using an elevation map, as described in Sec. V.B. The height of the corresponding point $\boldsymbol{P}'_k$ on the arc surface needs to be compensated with the distance $|\overrightarrow{\boldsymbol{P}_k\boldsymbol{P}'_k}|$. To this end, the checkpoint $\boldsymbol{P}'_k$ is first projected perpendicularly to the wheel's rotation axis $\boldsymbol{a}_w$, resulting in the perpendicular line $|\overrightarrow{\boldsymbol{P}'_k\boldsymbol{A}_k}|$. This line is then projected onto the rectangular contact plane to determine the projection length d_k. Given the wheel radius R_w, the compensation distance is calculated as $|\overrightarrow{\boldsymbol{P}_k\boldsymbol{P}'_k}| = R_w - \sqrt{R_w^2 - d_k^2}$. The simplified model is shown in Fig. 10(c) , where the orange dashed boxes represent the wheel areas and the blue dashed box represents the body area.

Following the procedure detailed in Sec. V.B, configuration stability is then estimated for a given target point $\boldsymbol{P}_o$ and heading $\boldsymbol{m}$. The robot model is initially aligned with the target heading $\boldsymbol{m}$ and the normal vector $\boldsymbol{n}$. Neighboring points are extracted from the local map to generate an elevation map. Heights for wheel checkpoints are calculated, and contact points are identified to generate a support polygon. Then, configuration stability is analyzed by examining the positional relationship between the CoM and the support polygon. If three or more wheels are in contact, the ZMP criterion is typically satisfied, indicating stable contact (see Fig. 10(h)). If fewer than three wheels are in contact, the configuration is deemed unstable, and rotations are iteratively conducted to search for stable contact. Unstable contact includes three cases:

a) Single-point contact occurs when the contact points are concentrated on a single wheel with the center point $\boldsymbol{P}_t$ (see Fig. 10(d)). The tip-over axis is $\boldsymbol{a} = \boldsymbol{g} \times |\overrightarrow{\boldsymbol{P}_c\boldsymbol{P}_t}|$.
b) Single-side contact happens when contact points are spread across two wheels on the same side (either left or right), as shown in Fig. 10(e) . SVD is performed on these points, with the eigenvector corresponding to the largest eigenvalue serving as the axis $\boldsymbol{a}$.
c) Insufficient contact arises when the contact points form a small polygon (see Fig. 10(f) and (g). The edge of the support polygon nearest to the projected CoM $\boldsymbol{P}_g$ is selected as the axis $\boldsymbol{a}$.

The iteration proceeds until a stable contact is achieved or it exceeds the threshold N_{crit}. If no stable configuration is found, the target point with the heading is considered infeasible. Finally, collision is detected for the body area, and a collision-free configuration is deemed feasible.

D. Local Path Planning

The proposed configuration-stability estimation approach evaluates a given waypoint and determines if a feasible configuration can be generated. It needs to be further integrated into a pathfinding algorithm to create smooth paths. To avoid the high latency caused by extensive node evaluation and the randomness of node generation in sampling-based methods, we design a terrain-aware local path planning approach based

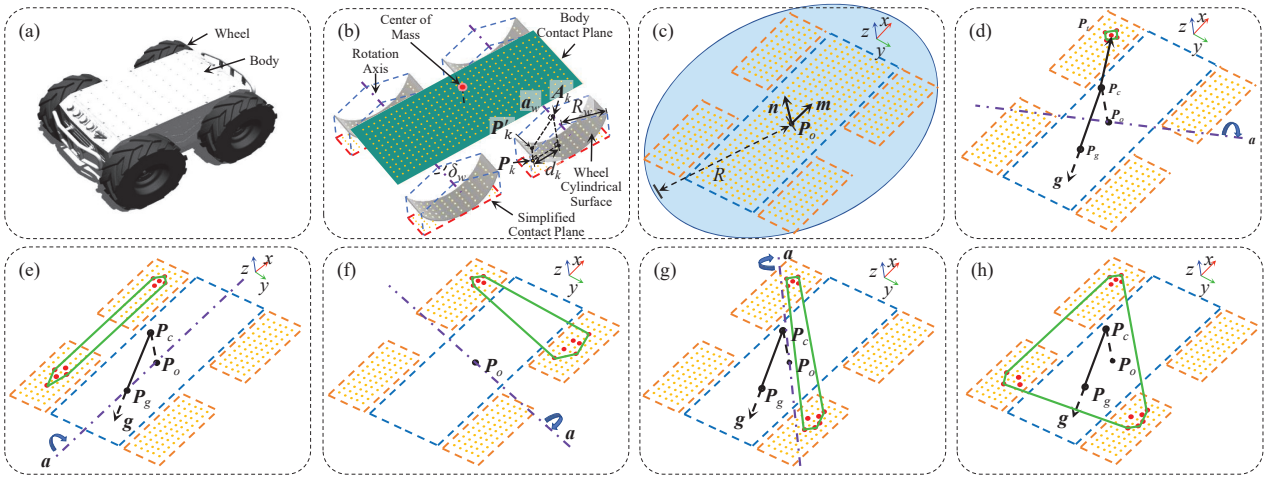


Fig. 10. The process of configuration-stability estimation for a differential wheeled robot. (a) Differential wheeled vehicle. (b) Abstract robot model with the body (green plane) and wheels (gray arcs). (c) Simplified robot model. (d) Single-point contact with a few wheel contact points (red dots). (e) Single-side contact with the two left-side wheels in contact with the ground. (f) (g) Insufficient contact with the projected CoM point $\boldsymbol{P}_g$ lying outside the support polygon (in green). (h) Stable contact with the projected CoM point located inside a large contact polygon.

on the graph search algorithm Hybrid A* [7]. This approach expands nodes in continuous space while adhering to the robot's kinematic constraints using motion primitives.

As shown in Fig. 6, information on the global layer is shared with the local layer to accelerate local planning. This information includes the normal vector and traversability data from the global map, as well as the generated global path. First, as mentioned in Sec. V.B, the normal vector for a given base position is retrieved directly from the global map to initialize the base pose, thereby eliminating redundant calculations of normal vector estimation. The subsequent iterative evaluation of configuration-stability estimation can converge faster based on the initial base pose. Second, the traversability of local map voxels is queried from the global map, and a set of traversable voxels is maintained as potential nodes for local planning. To speed up retrieval, a k-dimensional tree is constructed for local traversable voxels. Configuration-stability estimation is applied to expanded nodes, and those that meet the stability criteria are selected as candidate nodes. By introducing global traversability, the search space for local path planning is effectively narrowed, avoiding unnecessary configuration-stability estimation in untraversable areas.

Third, the global path produced by the global layer is used to guide the generation of the local path. Due to the limited scope of the local map, some waypoints on the global path may fall outside its boundaries. Therefore, directly using the global path's endpoint as the target for local planning is impractical. Instead, a waypoint at index W_{crit} on the global path is chosen as the target for local planning; if the global path has fewer than W_{crit} waypoints, the endpoint is chosen as the target. This ensures that the truncated global path is within the scope of the local map. Afterward, the truncated global path is used to calculate the heuristic cost, eliminating the need to search for the path from each expanded node to the target point. Specifically, the global path is first linearly interpolated to generate dense waypoints and identify the nearest global waypoint $\mathbf{P}_n$ to the current node with distance D_n. Next, the remaining length after the point $\mathbf{P}_n$ on the global path is combined with the distance D_n to determine the heuristic cost. Consequently, the global path effectively guides the direction of node expansion, ensuring efficient local planning. Finally, the generated local path is sent to the base controller to enable autonomous navigation in rough terrains.

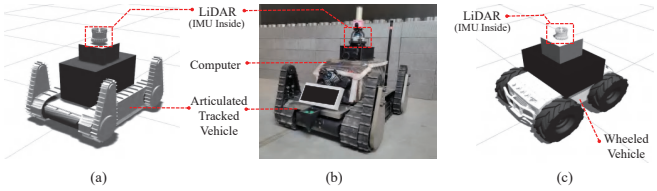


Fig. 11. The robots in simulation and the real world. (a)(b) The articulated tracked robot in simulation and the real world. (c) The differential wheeled robot in simulation.

VI. Simulations and Experiments

The proposed framework was thoroughly evaluated through simulations and real-world experiments, compared with state-of-the-art approaches. As shown in Fig. 11(a) and (c), we used the Gazebo[2] simulator to model an articulated tracked robot and a differential wheeled robot. As depicted in Fig. 11(b) , in the real world, the robot was equipped with LiDAR and IMU, with the captured data fed into the SLAM module [42] for pose estimation. In the simulation, ground truth odometry and LiDAR frames were directly used to update the local and global maps. The controller involved model predictive control to drive the main tracks and PID control to operate the flippers. Evaluations were executed on the robot's onboard computer, equipped with an Intel Core i9-10900K CPU and 32GB RAM. The parameters are listed in Table I.

[2]https://gazebosim.org/

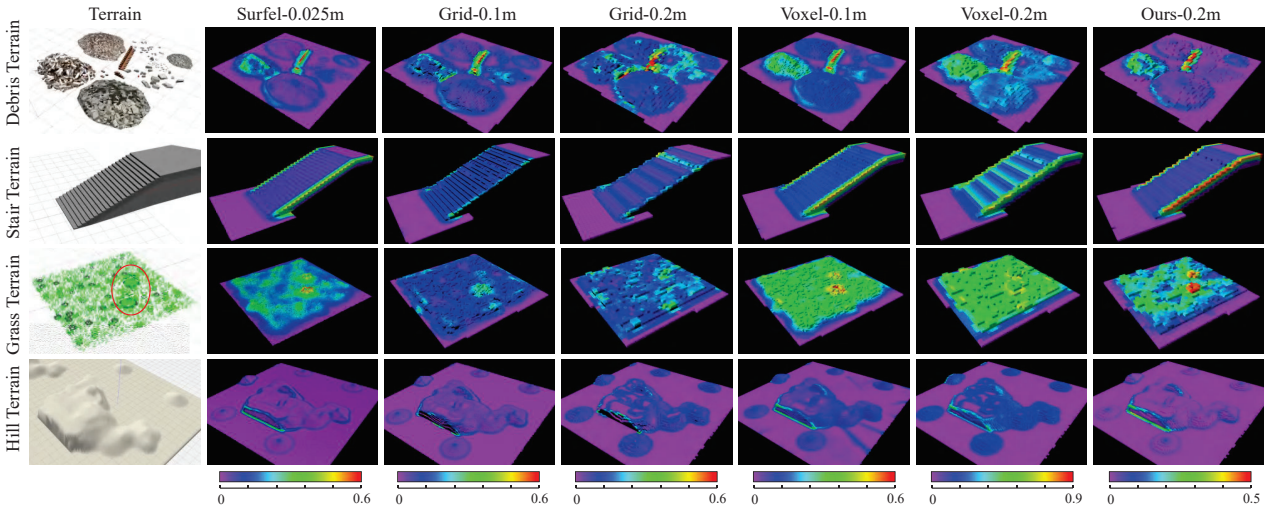


Fig. 12. Roughness results under different terrain representations in four simulated terrains. The first column depicts the simulated scenes. The other columns show the actual roughness calculated using different map representations.

TABLE I
PARAMETERS

Parameter	Symbol	Value	
		Tracked Vehicle	Wheeled Vehicle
Roughness Weight	α_r	0.3	0.4
Roughness Threshold	r_{crit}	0.5	0.3
Slope Weight	α_s	0.5	0.4
Slope Threshold	s_{crit}	38°	25°
Sparsity Weight	α_d	0.2	
Sparsity Threshold	d_{crit}	0.7	
Terrain Complexity Threshold	τ_{crit}	0.805	
Robot Radius	R	0.6 m	1 m
Robot Height	H	0.6 m	0.7 m
CoM Height	h	0.2 m	0.3 m
Ground Clearance	h_{crit}	0.08 m	0.13 m
Wheel Central Angle Range	δ_{crit}	-	120°
Saturate Threshold	K_{crit}	40	
Checkpoint Angular Interval	δ_c	20°	
Distance Cost Weight	α_l	0.5	
Traversal Cost Weight	α_τ	0.5	
Global Map Resolution	ξ_g	0.2 m	
Local Map Resolution	ξ_l	0.025 m	
Local Map Size	$\mathcal{S}_l$	4×4×2 m³	
Contact Point Tolerance	ϵ	0.025 m	
Rotation Angle Interval	δ_a	5°	
Max Iterative Times	N_{crit}	40	
Truncated Waypoint Number	W_{crit}	10	

A. Terrain Analysis Evaluation

1) Terrain Metric Accuracy: Multiple scenes were created in simulation to evaluate the terrain analysis (see Fig. 12). The proposed approach was compared to three popular approaches: grid map [19], voxel map [13], and surfel map [16]. The grid map [19] excels in terrain analysis by maintaining the elevation map, simplifying calculation into 2D operation. The voxel map [13] enhances terrain awareness by extracting planes with neighboring cells for estimating geometric metrics. The surfel map [16] models the terrain by fitting surfels and maintains the most precise terrain metrics with the dense point clouds. Herein, the ground truth point clouds were generated by the Gazebo plugin using a resolution of 0.025 m, with terrain metrics derived from the surfel map serving as the benchmark. Considering the sparsity metric cannot be reflected in the dense ground truth point cloud, its evaluation was omitted. We focused on evaluating the accuracy of roughness and slope metrics, as shown in Figs. 12-15. To ensure fairness in comparison, formulas (3) and (4) were uniformly applied to calculate terrain metrics on different terrain representations.

Map resolution significantly impacts terrain metric results, traversability estimation, and computation runtime. Therefore, it is crucial to choose a resolution that matches the robot's motion capabilities and geometric dimensions while balancing the accuracy of terrain representation with real-time performance. Herein, our implicit map was evaluated at the resolution of 0.2 m, while the grid and voxel map were evaluated at the resolutions of 0.1 m and 0.2 m. Due to significant covariance differences across multiple map representations and resolutions, roughness values can vary considerably. We utilized Pearson's correlation coefficient to measure the consistency of the roughness metric, defined as follows:

$$r(X,Y) = \frac{cov(X,Y)}{\sqrt{D(X)D(Y)}}, \tag{11}$$

where $cov(X,Y)$ denotes the covariance of two variables X and Y, and $D(X)$ and $D(Y)$ indicate their standard deviations. The terrain roughness and correlations between the surfel-0.025m map and those from others were calculated (see Figs. 12 and 14). As for the slope metric, normal vectors obtained from the surfel-0.025m map served as the benchmark. The slope error was assessed by comparing the intersection angles of estimated normal vectors across different maps (see Figs. 13 and 15). In addition, the map integration runtimes for these approaches are listed in Table II (refer to Sec. VI.A.2.), where the bolded values indicate the optimal results across comparative approaches.

Ours-0.2m map preserved roughness sensitivities, consistent with the surfel-0.025m map (see Fig. 12). Our approach

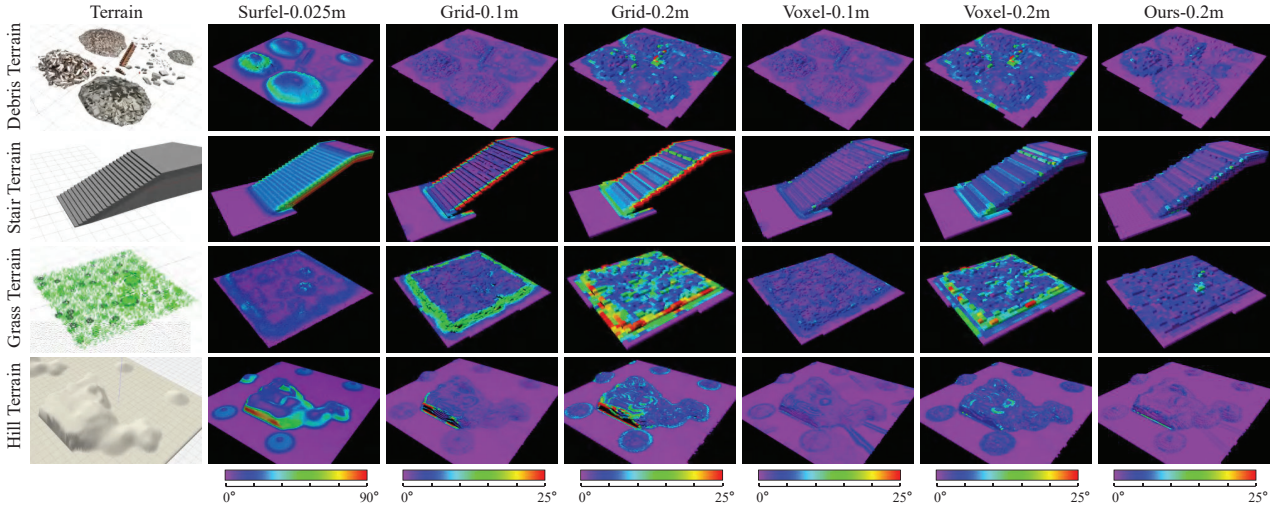


Fig. 13. Slope results under different terrain representations in four simulated terrains. The first column depicts the simulated scenes. The second column shows the actual slopes calculated using surfel-0.025m. The five columns on the right display the differences between slopes from the surfel-0.025m map and those from other map representations, highlighting the distribution of errors across different areas.

TABLE II
TERRAIN ANALYSIS AND GLOBAL PLANNING RESULTS

Scene	Approach	Map	Map Integration Runtime(s)	Terrain Assessment Runtime(s)	Terrain Update Runtime(s)[1]	Path Planning Runtime(s)	Total Runtime(s)[2]	Path Length(m)	Passing Shortcut
Multi-layer Stairs (8×8×10m³) 858,447 points	ECTN	Grid-0.1m	**0.014**	**0.051**	**0.065**	-	-	-	False
	PUTN	Voxel-0.1m	0.039	0.743	0.782	-	-	-	False
	SSSN	Surfel-0.1m	0.135	4.895	5.030	0.122	5.152	40.924	**True**
	Ours	Ours-0.1m	0.049	0.727	0.776	0.129	0.905	39.446	**True**
		Ours-0.2m	0.028	0.066	0.094	**0.021**	**0.115**	**38.484**	**True**
Uneven Terrain (15×20×3m³) 1,299,595 points	ECTN	Grid-0.1m	**0.048**	**0.093**	**0.141**	0.202	0.343	18.990	False
	PUTN	Voxel-0.1m	0.088	2.078	2.166	0.324	2.490	20.100	False
	SSSN	Surfel-0.1m	0.323	17.658	17.981	0.383	18.364	15.960	**True**
	Ours	Ours-0.1m	0.097	1.655	1.752	0.332	2.084	16.800	**True**
		Ours-0.2m	0.068	0.141	0.209	**0.043**	**0.252**	**15.380**	**True**
Debris Terrain (20×30×4m³) 3,665,823 points	ECTN	Grid-0.1m	**0.127**	**0.233**	**0.360**	0.472	0.832	34.350	False
	PUTN	Voxel-0.1m	0.222	6.068	6.290	0.836	7.126	34.130	False
	SSSN	Surfel-0.1m	0.970	52.618	53.588	0.866	54.454	27.630	**True**
	Ours	Ours-0.1m	0.267	4.804	5.071	0.803	5.874	27.277	**True**
		Ours-0.2m	0.191	0.369	0.560	**0.103**	**0.663**	**26.460**	**True**
Large Scale Terrain (200×200×10m³) 405,055,675 points	ECTN	Grid-0.1m	**3.544**	**17.322**	**20.866**	44.122	64.988	384.361	False
	PUTN	Voxel-0.1m	15.331	600.247	615.578	99.689	715.267	358.058	**True**
	SSSN	Surfel-0.1m	63.924	5223.400	5287.324	105.047	5392.371	336.228	**True**
	Ours	Ours-0.1m	22.050	470.035	492.085	89.009	581.094	**329.191**	**True**
		Ours-0.2m	11.783	33.875	45.658	**13.073**	**58.731**	329.950	**True**

[1] The terrain update runtime is the sum of the map integration runtime and the terrain assessment runtime.
[2] The total runtime is the sum of the path planning runtime and the terrain update runtime.

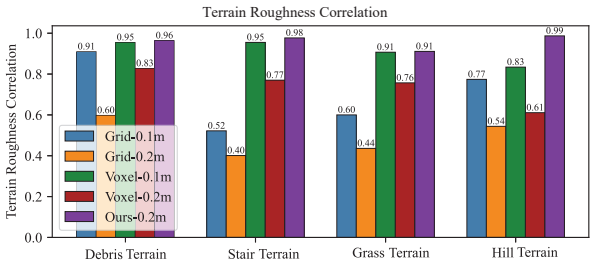


Fig. 14. Terrain roughness correlations for different terrain representations in four simulated terrains, with the surfel-0.025m map serving as the benchmark.

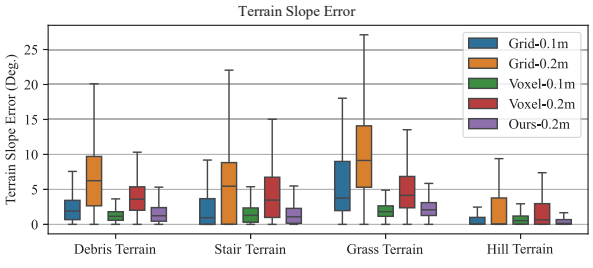


Fig. 15. Terrain slope errors for different terrain representations in four simulated terrains, with the surfel-0.025m map serving as the benchmark.

obtained the roughness correlation coefficients approximately equal to 1.0, outperforming the grid-0.1m and voxel-0.1m

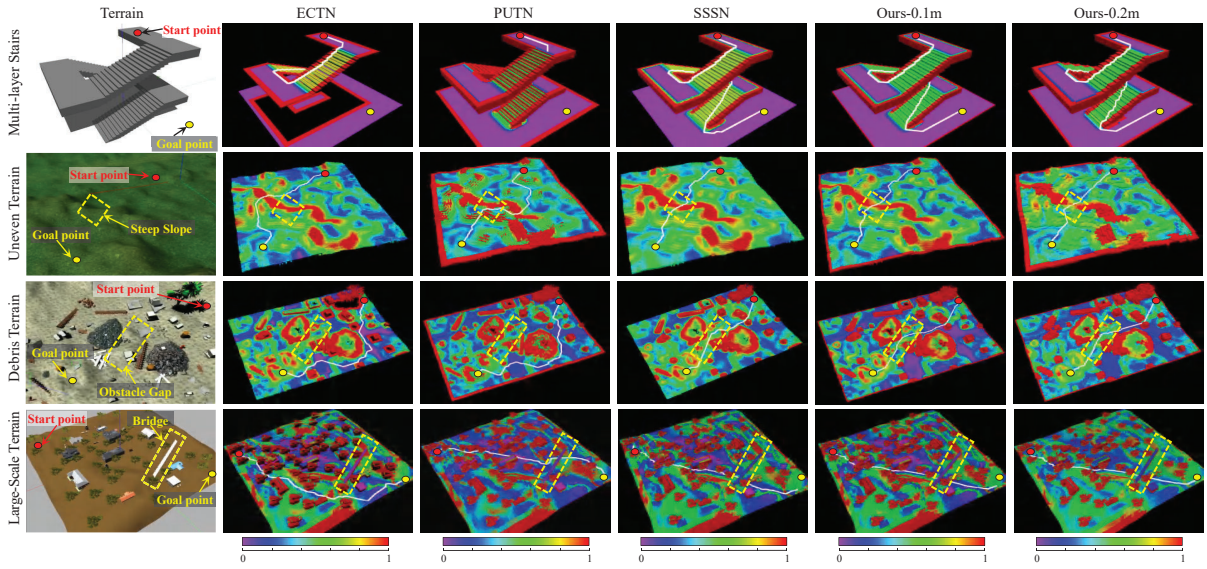


Fig. 16. Terrain analysis and global planning across four complex terrains. Red dots indicate the start points, while yellow dots present the goal points. The planned paths are shown with white curves, and the map is colored with traversal cost. The yellow dashed box in each scene highlights a key shortcut that exists in a challenging area.

maps (see Fig. 14). As for the debris terrain shown in Fig. 12, the surfel-0.025m map revealed different roughness values for each pile. With the cell size increasing, roughness differences between piles became less distinct in the grid-0.2m and voxel-0.2m maps, whereas our approach maintained clear roughness distinctions. This trend was also observed in other terrains. In the stair terrain, increasing the cell size led to noticeable roughness errors on the steps in the grid-0.2m and voxel-0.2m maps due to discretization errors. In contrast, ours-0.2m map exhibited no significant degradation. In the grass terrain, the surfel-0.025m map displayed varying heights of grasses with different roughness levels (see Fig. 12). The roughness differences in grid-0.2m and voxel-0.2m maps became less pronounced, whereas our approach still effectively distinguished roughness between tall grass areas. In the hill terrain, the discretization error of grid-0.2m and voxel-0.2m maps amplified the roughness on the ground and ramps with minor variations. Conversely, ours-0.2m map preserved roughness features similar to those in the surfel-0.025m map.

As illustrated in Fig. 15, ours-0.2m map, performing similarly to the voxel-0.1m map, provided lower slope errors than the grid-0.1m map across the debris, stair, and grass terrains. In the hill terrain, ours-0.2m map surpasses the voxel-0.1m map in slope accuracy. Slope errors for both grid-0.1m and voxel-0.1m were concentrated in areas with significant terrain changes, with errors becoming more pronounced at lower resolutions (see Fig. 13). In contrast, ours-0.2m map achieved lower slope errors by utilizing normal vectors fused with neighboring Gaussian distributions.

2) Global Path Planning Performance: Four challenging scenes were simulated to evaluate the global planning performance under different terrain analysis approaches (see Fig. 16). Shortcuts were created for key terrain features within specific traversable areas. These shortcuts turned out to be feasible by manually driving the base and operating the flippers. Afterward, the shortcuts were used to test whether the planned paths could pass through them. As shown in Fig. 16, some shortcuts that mainly affect the path generations are marked by yellow dashed boxes. However, errors in terrain analysis may lead to misclassified untraversable areas, potentially blocking the shortcuts, causing detours, and increasing path lengths.

The proposed global planning approach was evaluated with three comparative traversability-estimation approaches, such as elevation-based challenging terrain navigation (ECTN) [19], plane-fitting based uneven terrain navigation (PUTN) [13], and surfel-based state-space navigation (SSSN) [16]. The three approaches used the grid map, the voxel map, and the surfel map, respectively. ECTN [19] evaluated traversability by considering roughness, slope, and step height for collision detection. PUTN [13] added a sparsity metric to identify suspended regions by analyzing neighboring vacancies. SSSN [16] calculated traversability with roughness, slope, step height, and ground clearance. The recommended parameters from [13], [16], [19] were applied, with the maps set to the resolution of 0.1 m. Our approach was evaluated using resolutions of 0.1 m and 0.2 m, respectively, to determine its efficiency across different spatial scales. These approaches supported various pathfinding algorithms such as A* [24], RRT [23], and PRM [6]. Therefore, the comparison focused on how different terrain representations and traversability estimations affect path planning, rather than evaluating path pathfinding algorithms. For a fair comparison, all the path planning was conducted using the approach described in Sec. IV.D with the results presented in Fig. 16. The computation runtimes and path lengths across four terrains are listed in Table II.

Regarding traversing shortcuts and path lengths (see Table II), the proposed approach yielded competitive results across all four terrain scenes, particularly in those with complex features (see Fig. 16). Our approach, with a 0.2 m resolution, did not experience significant degeneration due to its ability

to represent sub-resolution details, producing consistent paths similar to those generated at a 0.1 m resolution. The ECTN and PUTN approaches were less effective due to limitations in terrain analysis and representation. They missed more shortcuts and produced longer paths. Specifically, in the first scene with multi-layered stairs, both the ECTN and PUTN approaches failed to reach the target. The ECTN approach was hindered by its 2.5D grid map, losing elevation information for the bottom stairs. Meanwhile, the PUTN approach, despite using a sparsity metric to detect cliffs, misidentified stairs as untraversable. The second scene featured uneven terrain with varying heights of humps and hollows, where a shortcut was located in a region with significant terrain fluctuations. Again, the ECTN and PUTN approaches missed the shortcut due to incorrect traversability on steep slopes. The third scene presented a rubble terrain with scattered obstacles, such as bricks and stone piles, where a shortcut passed through gaps between these obstacles. Still, the ECTN and PUTN approaches failed to detect potential paths through these gaps. The fourth scene depicted a large-scale terrain with numerous trees and buildings, where a shortcut was located under a bridge. Similarly, the ECTN approach failed to recognize multi-layered structures, creating an untraversable region under the bridge and forcing a detour. Meanwhile, the PUTN approach planned a path under the bridge but caused a local detour by misclassifying the regions with steep slopes near the target point as untraversable. In contrast, both the proposed and the SSSN approaches effectively identified and traversed the shortcuts. With superior terrain analysis, they maintained a safe clearance from dangerous edges while reaching the targets via shorter paths. Note that our approach outperformed the SSSN approach in computational efficiency, as detailed below.

Computation runtimes, such as map integration, terrain assessment, terrain update, path planning, and total runtimes, were further evaluated (see Table II). The map integration runtime is the time taken by converting the original point cloud into the map representation. The terrain assessment runtime is the time taken to calculate geometric metrics and traversability. The terrain update runtime consists of the map integration and terrain assessment runtimes, while the total runtime is the sum of the terrain update and path planning runtimes. Our approach was first evaluated using the same resolution (0.1m) as the other approaches to ensure a fair comparison. Next, we utilized a resolution of 0.2m to highlight the performance difference of our approach at sub-resolution. At a resolution of 0.1m, our approach exhibited a disadvantage compared to the ECTN and PUTN approaches in map integration runtime, as the update and fusion of each voxel's Gaussian distribution introduced additional computational overhead. However, our approach outperformed the PUTN approach in terrain update runtime. Regarding path planning runtime (see Table II), our approach was comparable to the PUTN and SSSN approaches when using the same resolution. When operating at a 0.2m resolution, our approach achieved significant acceleration. Although terrain update runtimes were slightly longer than those of the ECTN approach, the overall efficiency of our approach surpassed all other approaches.

TABLE III
ROUTE PASS RATE

Scene (Route #)	NF	CS	GEO	Ours
Gravel (12)	7	9	8	**11**
Rubble (12)	6	9	7	**11**
Bump (12)	6	7	9	**10**
Bar (12)	3	6	7	**10**
Stair (12)	4	5	10	**12**
Total (60)	26	36	41	**54**
Pass Rate	43.0%	60.0%	68.0%	**90.0%**
False Negative	-	12	**0**	3
False Positive	-	12	19	**3**

B. Configuration-Stability Estimation For Tracked Vehicle

As depicted in Fig. 17, five challenging scenes were simulated to evaluate configuration-stability estimation for the tracked vehicle described in Sec. V.B. The gravel terrain, with unevenly distributed protrusions, challenged the robot's stability through body collisions and insufficient contact. The rubble terrain, featuring obstacles of varying sizes, posed risks of both collisions and tipping. The bump terrain posed higher risks of immobilization and tipping, requiring adaptive flippers to absorb shocks and maintain balance. The bar terrain's uneven distribution caused driving instability, necessitating coordinated flippers to bridge gaps and maintain ground contact. The stair terrain's irregular risers complicated flipper operation, demanding adaptive strategies to alleviate tipping during ascent or descent. For each terrain, 12 routes were generated to traverse the area from various positions and orientations, with lengths of 6 m, 5 m, 15 m, 15 m, and 12 m, respectively. These routes turned out to be feasible by manually driving the base and operating the flippers. Subsequently, point clouds of the simulated terrains with the resolution of 0.025 m were generated as the ground truth using the Gazebo map plugin. A sequence of waypoints was created for each route at 0.025 m intervals to evaluate configuration-stability estimation.

Our approach was compared with a reference approach (NF) and two state-of-the-art approaches: configuration-space motion planning (CS) [37] and geometry-based motion planning (GEO) [15]. The NF approach does not estimate configuration stability. Instead, it indicates the robot's ability to traverse complex terrain using only the main tracks, with the flippers resetting to an initial angle of 45° during operation. The CS approach [37] models the tracks and flippers as a skeletal morphology, determining the base pose and flipper angles through the pivot of each track. The optimal configuration that satisfies the stability criteria is selected, ensuring the robot's adaptability in complex terrains. The GEO approach [15] simplifies computational complexity by modeling the tracks into three interconnected line segments. Flipper motion planning is conducted using dynamic programming, optimizing multiple stability factors to ensure smooth movements. The above approaches were evaluated with three metrics: pass rate (see Table III), configuration orientation error (see Fig. 18(a)), and configuration height error (see Fig. 18(b)). The pass rate is the proportion of the 60 routes that are successfully traversed. A route is considered successfully traversed if each waypoint has an estimated stable configuration, and the robot

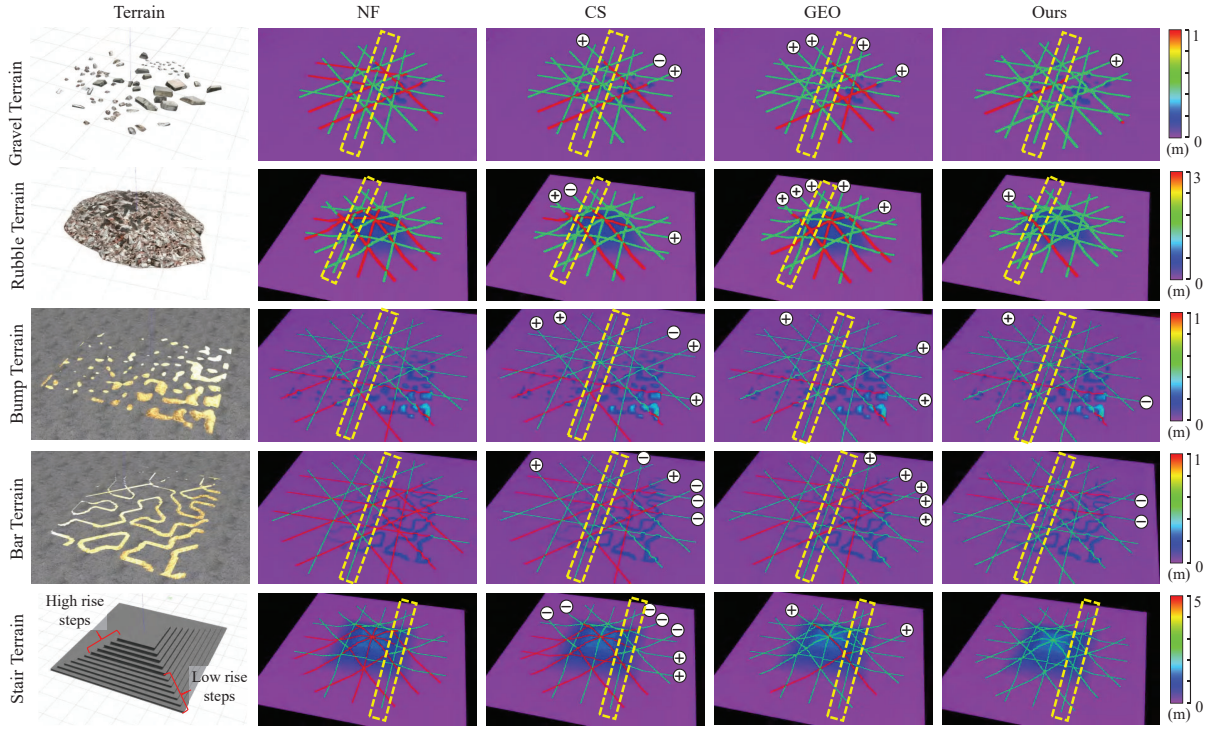


Fig. 17. Traversal results across various terrains. The first column shows five simulated scenes, where the last one features pyramid-shaped stairs with high-rise steps at the top and low-rise steps at the bottom. The four columns on the right display local maps colored by elevation. Routes marked with ⊕ are false positives, while those marked with ⊖ are false negatives.

safely passes through all the waypoints. On one hand, a false positive route occurs when each waypoint has an estimated stable configuration, but the robot cannot fully traverse the route. On the other hand, a false negative route refers to a route that is actually feasible but lacks an estimated stable configuration at one or more waypoints.

As illustrated in Fig. 17, green segments indicate successfully traversed portions, while red segments denote infeasible configurations or traversal failures. The false positive and false negative routes are marked with ⊕ and ⊖, respectively. Additionally, configuration orientation and height errors are key for evaluating the accuracy of configuration-stability estimation. Orientation error measures the angular difference between the estimated and actual orientation at a waypoint, while height error captures the discrepancy in the z-axis position. A single route per scene, where all approaches successfully navigate, was selected for error evaluation, as indicated by the yellow dashed boxes in Fig. 17. Note that, for the NF approach, the terrain's normal vector is defined as the z-axis when calculating the orientation error.

The NF approach, which lacks flipper support, exhibited the lowest pass rate (see Table III). It only traversed low-rise steps in the stair terrain and low obstacles in the bar terrain, producing the highest errors in configuration height and orientation (see Fig. 18). In contrast, the other three approaches achieved higher pass rates. Both the proposed and the CS approaches iteratively estimated potential contact with terrain to search for stable configurations, with configuration heights and orientations closely aligning with the actual terrain. However, the CS approach had a relatively low pass rate due to its heavy reliance on the initial pivot selection. This dependency resulted in situations where iterative rotation, assuming a main track's endpoint as the initial pivot, may fail to produce a stable configuration. The CS approach simplified the flippers and tracks to a skeletal model, neglecting changes in contact points during iterative rotations, which further introduced errors in configuration-stability estimation (see Fig. 18(b)). As a result, the CS approach reported 12 false negatives and 12 false positives (see Table III), indicating a balance between the misidentification of stable and unstable configurations.

By comparison, the GEO approach employed the most simplified robot model but maintained a relatively high success rate due to its dynamic programming process. However, the loss in terrain sensitivity resulting from the model's simplification was not thoroughly addressed. This issue was evident in the configuration orientation and height errors see (Fig. 18(a) and (b)). Consequently, the GEO approach was more prone to overestimating stable configurations and experienced a higher number of false positives (see Table III). In contrast, our approach overcomes terrain challenges by directly analyzing the contact surfaces of the tracks and flippers rather than relying on the skeleton abstractions. Furthermore, the rotation axis in our approach is determined based on the support polygon during the iterative geometric evaluation rather than assuming fixed pivots like the CS approach. Based on detailed geometric evaluation, our approach not only achieved the highest pass rate but also mitigated false negatives and false positives, with only three instances of each. The reduced false

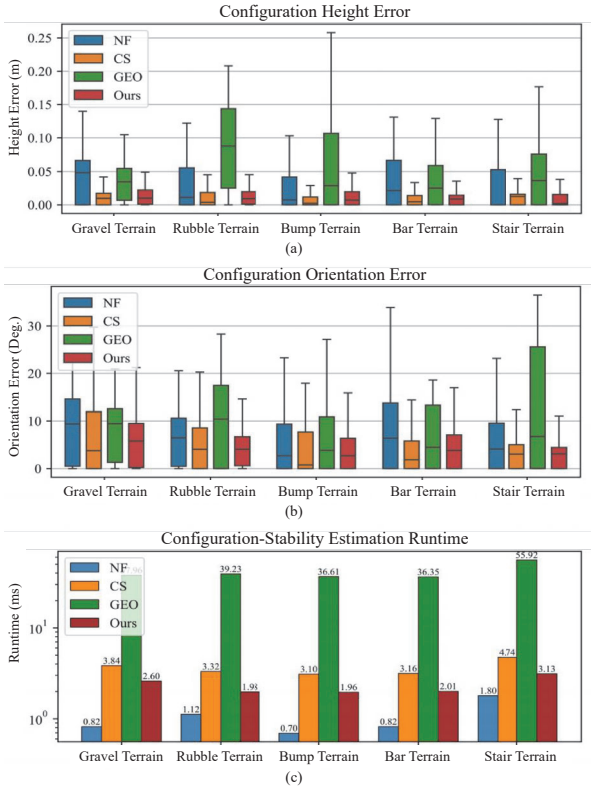


Fig. 18. Configuration errors and computation runtimes in simulation scenes. (a) Configuration height error. (b) Configuration orientation error. (c) Configuration-stability estimation runtime.

positive rate guaranteed that most infeasible configurations are correctly identified, while the decreased false negative rate ensured fewer incorrect rejections of stable configurations.

The average time each approach took to estimate configuration stability for each waypoint along the route is shown in Fig. 18(c) . The NF approach, which calculates only the normal vector with neighboring points for each waypoint, incurred minimal runtime. Both the proposed approach and the CS approach presented higher time consumption due to multiple iterations. The proposed approach detects main track contact during iterations and then analyzes flipper contact and body collision, terminating as soon as a stable configuration is identified. By comparison, the CS approach alternates between left and right pivots to search for all possible contact configurations and selects the optimal one, requiring more time to complete. The GEO approach, due to its computationally intensive dynamic programming, produced the longest process. Overall, our approach outperformed the comparative approaches in both accuracy and efficiency.

C. Application For Wheeled Vehicle

A simulated scene depicted in Fig. 19(a) was further built to evaluate the applicability of the proposed framework to a wheeled robot. Since wheeled vehicles have lower traversal capability than tracked vehicles, we applied stricter traversal thresholds for global planning (see Table I). Specifically, slope weight α_s and roughness weight α_r were set to 0.4. The slope

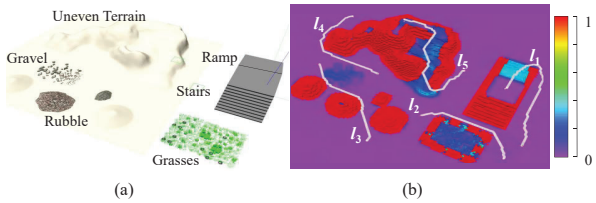


Fig. 19. Global planning results of the wheeled vehicle for uneven terrain. (a) The simulated scene. (b) The global map is colored with traversal cost, and planned global paths are depicted as white curves.

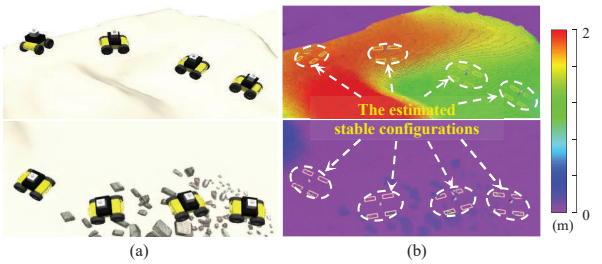


Fig. 20. Configuration-stability estimation results for the wheeled vehicle in the simulated scene. (a) Rough terrain and historical robot positions. (b) Local map and configuration-stability estimation results, with voxel colors indicating elevation and white dashed circles marking the estimated configurations.

threshold s_{crit} was reduced to 25°, and the roughness threshold r_{crit} was decreased to 0.3. These adjustments prevented the wheeled vehicle from traversing dangerous rough areas, such as stairs. The robot parameters are detailed in Table I. As shown in Fig. 19(b) , the proposed approach generated several representative global paths for the wheeled vehicle traversing the rough terrain, effectively avoiding areas with high-traversal cost. Path l_1, constrained by roughness, opted to traverse the ramp rather than the stairs, resulting in a longer route. Similarly, paths l_2, l_3, and l_4 bypassed rougher areas and preferred flatter terrain. Path l_5, while traversing uneven terrain, preferred regions with lower traversal cost. For local planning of the wheeled vehicle, the approach described in Sec. V.C for configuration-stability estimation was evaluated in the simulated environment shown in Fig. 20. The robot successfully traversed ramp and gravel areas, with the planned configurations making stable contact with the terrain. Additionally, the robot navigated the terrain smoothly with the aid of the controller, demonstrating that the proposed approach is well-suited for the wheeled vehicle.

D. Field Experiments

Typical complex scenes from urban search and rescue missions were created to evaluate the proposed framework using an articulated tracked robot, with the map volume reaching up to 1,360 m³ (see Fig. 21). These scenes highlighted a range of challenging terrains. Bump and debris regions required precise coordination between the flippers and tracks to navigate rugged structures. Two staircases and a tent area posed difficulties for multi-level navigation. An office area with obstacles like desks, chairs, and walls emphasized the need for effective obstacle detection and avoidance. These terrain features could evaluate the comprehensive ability to handle real-world challenges.

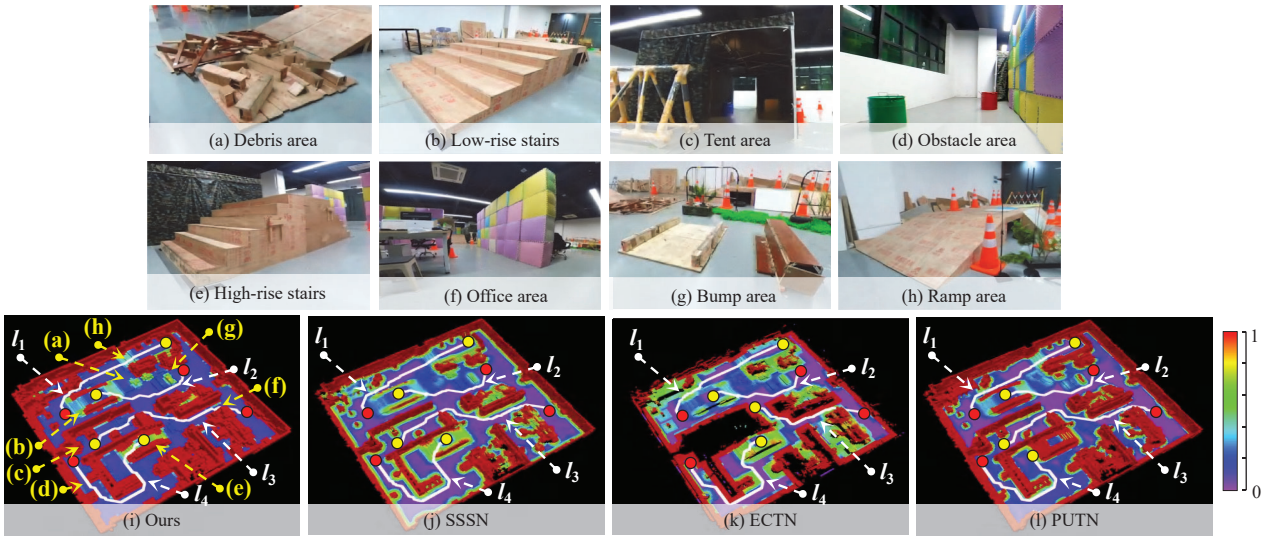


Fig. 21. Real-world complex scene with dimensions of 16×17×5=1,360 m³. (a) Debris area. (b) Low-rise stairs. (c) Tent area. (d) Obstacle area. (e) High-rise stairs. (f) Office area. (g) Bump area. (h) Ramp area. (i)–(l) Global traversability maps and planned global paths are generated by four approaches, such as (i) ours, (j) SSSN, (k) ECTN, and (l) PUTN. The maps are colored with traversal costs similar to Figs. 16 and 3(g). Each path originates at a red dot with a yellow dot as its destination.

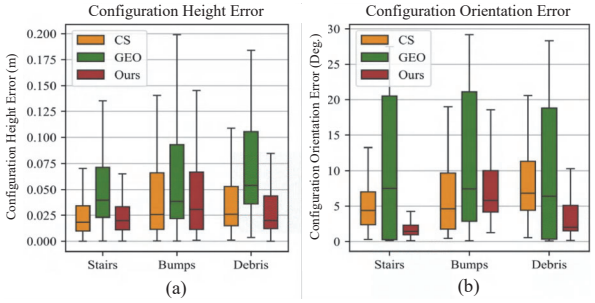


Fig. 22. Configuration errors in real-world scenes. (a) Configuration height error. (b) Configuration orientation error.

Given a sequence of target points, the robot autonomously navigated multiple areas using the proposed pipeline (see Fig. 1). The experiments demonstrated the effectiveness of the proposed approach in planning both global and local paths across various terrains. Fig. 21(i) displays the generated traversability map and four representative global paths.

After traversing these paths, the scene was fully scanned. Similar to the evaluation in Sec. VI.A, the obtained dense point cloud was further utilized to evaluate terrain analysis and path planning with three comparative approaches, including ECTN [19], PUTN [13], and SSSN [16]. The produced maps and paths are illustrated in Figs. 21(j), (k) and (l). The maps were truncated at the tent roof height (1.2 m) to prevent occlusion in the visualization. Planned paths originated from red dots and aimed toward yellow dots. The maps and paths produced by the SSSN approach served as the reference benchmark. As a result, paths l_1 and l_2, generated by the ECTN and PUTN approaches, successfully navigated around the stairs and traversed the ramp to reach the target points. However, the comparative approaches still faced the shortcomings described in Sec. VI.A.2. Specifically, the ECTN approach's grid map was unable to represent 3D structures, resulting in missing elevation data inside the tent and causing the path l_3 to fail in reaching the target point. Meanwhile, the PUTN approach's sparsity metric misclassified high-rise stairs as untraversable, preventing path l_4 from reaching the destination. In contrast, our approach generated traversability maps and planned paths that closely resembled those of the SSSN approach.

Three challenging terrains were further selected to evaluate the configuration-stability estimation performance. Since the robot was unable to traverse these terrains without using flippers, our approach was compared only with two state-of-the-art approaches: CS [37] and GEO [15]. The first terrain featured high-rise stairs, as shown in Fig. 21(e) , with a single step arising 0.16 m in height. The second terrain consisted of two low bumps of 0.1 m and one high bump of 0.2 m, depicted in Fig. 21(g) . The third terrain involved debris, illustrated in Fig. 21(a) , with various sizes and shapes of wooden strips and blocks scattered on the ground. These features posed significant difficulties for configuration-stability estimation, requiring precise coordination of the flippers and tracks to maintain balance. Following the evaluation procedure outlined in Sec. VI.B, start and goal points were manually selected to form straight routes across the terrains. Waypoints along each route were used to estimate configuration stability. The robot then attempted to follow these routes, and configuration errors were recorded (see Fig. 22).

We further analyzed the robot's configurations at various stages across these terrains (see Fig. 23). The improperly estimated flipper angles are highlighted in red dashed boxes. The inaccurate configuration-stability estimations led to traversal failures and undesired behaviors that could damage the robot. As for the pass rate, our approach successfully navigated all the terrains. In contrast, the CS and GEO approaches could only traverse the stair terrain and failed to navigate the bump and debris terrains due to false positives. Specifically, the

Fig. 23. Robot configurations when traversing three challenging terrains in real-world scenes. The red dashed boxes highlight inproper flipper angles.

CS approach tended to overestimate flipper angles, leading to struggling and getting stuck. As shown in Fig. 22, although the CS approach demonstrated accuracy in configuration height, it exhibited greater errors in orientation. As for the GEO approach, it exhibited the highest uncertainty in both configuration height and orientation due to its coarse stability estimation (see Fig. 22), leading to additional issues like downthrust, sideslip, and shocks. These results are consistent with the configuration error (see Fig. 18) and pass rate (Table III) from simulations. In contrast, our approach maintained the most accurate configuration orientation and kept the configuration height similar to the CS approach (see Fig. 22). This allowed it to navigate all terrains smoothly, with the flippers adapting closely to the terrain (see Fig. 23). Although configuration errors in real-world scenes were slightly larger than those observed in simulations due to localization and mapping uncertainty, the results of the real-world experiment were closely aligned with those obtained in simulations. Our approach successfully navigated complex 3D environments and outperformed the others, demonstrating its robustness and reliability across both simulated and real-world conditions. For more details, please refer to the supplementary video.

VII. Conclusion and Future Work

In this paper, we introduce a multi-level terrain-aware path planning framework designed for ground mobile robots operating in large-scale rough 3D terrains. At the global layer, an implicit map representation is introduced for online terrain analysis, traversal risk assessment, and traversability estimation. Based on the traversability map, a graph-based global planning approach is designed to navigate complex 3D environments with overhangs and multi-layered structures. At the local layer, an iterative geometric approach that simulates the falling behavior under gravity for configuration-stability estimation is incorporated into local planning to generate smooth paths. The framework accommodates different ground mobile robots and supports various pathfinding algorithms. Challenging scenes in simulations and real-world environments were constructed to validate the framework's effectiveness across various ground mobile robots. Results demonstrated that the proposed approach enables autonomous navigation through diverse rough terrains and outperforms state-of-the-art approaches.

Future work will focus on three areas to enhance perfor-

mance and ensure broader applicability in real-world scenes. First, parallelizing the configuration-stability estimation to support real-time local planning with sampling-based pathfinding algorithms. Second, incorporating dynamic force analysis into configuration-stability estimation to address the limitations of quasi-static assumptions. Third, improving information sharing by feeding local layer data back into the global layer to enhance planning robustness and reliability.

REFERENCES

[1] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano *et al.*, "The current state and future outlook of rescue robotics," *J. Field Robot.*, vol. 36, no. 7, pp. 1171–1191, 2019.

[2] J. Wang, L. Xu, H. Fu, Z. Meng, C. Xu, Y. Cao, X. Lyu, and F. Gao, "Towards efficient trajectory generation for ground robots beyond 2d environment," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2023, pp. 7858–7864.

[3] T. Azayev and K. Zimmermann, "Autonomous state-based flipper control for articulated tracked robots in urban environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7794–7801, 2022.

[4] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2021, pp. 6914–6921.

[5] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *IEEE Int. Conf. Mech. Autom. Control. Eng.* IEEE, 2011, pp. 1067–1069.

[6] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *IEEE Int. Conf. Intell. Robots Syst.*, vol. 2. IEEE, 2004, pp. 1606–1611.

[7] K. Kurzer, "Path planning in unstructured environments: A real-time hybrid a* implementation for fast and deterministic path generation for the kth research concept vehicle," Master's thesis, KTH Royal Inst. Technol., 2016.

[8] C. Wang, L. Meng, S. She, I. M. Mitchell, T. Li, F. Tung, W. Wan, M. Q.-H. Meng, and C. W. de Silva, "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2017, pp. 109–116.

[9] C. Cao, H. Zhu, F. Yang, Y. Xia, H. Choset, J. Oh, and J. Zhang, "Autonomous exploration development environment and the planning algorithms," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2022, pp. 8921–8928.

[10] F. Yang, C. Cao, H. Zhu, J. Oh, and J. Zhang, "Far planner: Fast, attemptable route planner using dynamic visibility update," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2022, pp. 9–16.

[11] J. Zhang, C. Hu, R. G. Chadha, and S. Singh, "Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation," *J. Field Robot.*, vol. 37, no. 8, p. 1300–1313, 2020.

[12] F. Ruetz, E. Hernández, M. Pfeiffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges, "Ovpc mesh: 3d free-space representation for local ground vehicle navigation," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2019, pp. 8648–8654.

[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 *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2022, pp. 7160–7166.

[14] Y. Yuan, Q. Xu, and S. Schwertfeger, "Configuration-space flipper planning on 3d terrain," in *IEEE Int. Symp. Safety, Secur. Rescue Robot. (SSRR).* IEEE, 2020, pp. 318–325.

[15] B. Chen, K. Huang, H. Pan, H. Ren, X. Chen, J. Xiao, W. Wu, and H. Lu, "Geometry-based flipper motion planning for articulated tracked robots traversing rough terrain in real-time," *J. Field Robot.*, vol. 40, no. 8, pp. 2010–2029, 2023.

[16] F. Atas, G. Cielniak, and L. Grimstad, "Elevation state-space: Surfel-based navigation in uneven environments for mobile robots," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2022, pp. 5715–5721.

[17] L. Gan, Y. Kim, J. W. Grizzle, J. M. Walls, A. Kim, R. M. Eustice, and M. Ghaffari, "Multitask learning for scalable and dense multilayer bayesian map inference," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 699–717, 2022.

[18] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "Tamols: Terrain-aware motion optimization for legged systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3395–3413, 2022.

[19] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2016, pp. 1184–1189.

[20] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189–206, 2013.

[21] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3d navigation mesh generation for path planning in uneven terrain," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 212–217, 2016.

[22] A. Dixit, D. D. Fan, K. Otsu, S. Dey, A.-A. Agha-Mohammadi, and J. Burdick, "Step: Stochastic traversability evaluation and planning for risk-aware navigation results from the darpa subterranean challenge," *Field Robot.*, vol. 4, pp. 182–210, 2024.

[23] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[24] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *Int. J. Comput. Sci. Net. Secur.*, vol. 11, no. 1, pp. 125–130, 2011.

[25] J. Ahtiainen, T. Stoyanov, and J. Saarinen, "Normal distributions transform traversability maps: lidar-only approach for traversability mapping in outdoor environments," *J. Field Robot.*, vol. 34, no. 3, pp. 600–621, 2017.

[26] X. Liu, D. Li, and Y. He, "Multiresolution representations for large-scale terrain with local gaussian process regression," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2021, pp. 5497–5503.

[27] W. Tabib, K. Goel, J. Yao, C. Boirum, and N. Michael, "Autonomous cave surveying with an aerial robot," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1016–1032, 2021.

[28] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2017, pp. 1366–1373.

[29] A. Mitriakov, P. Papadakis, and S. Garlatti, "An open-source software framework for reinforcement learning-based control of tracked robots in simulated indoor environments," *Auton. Robot.*, vol. 36, no. 11, pp. 519–532, 2022.

[30] J. Guzzi, R. O. Chavez-Garcia, M. Nava, L. M. Gambardella, and A. Giusti, "Path planning with local motion estimations," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2586–2593, 2020.

[31] M. Norouzi, J. V. Miro, and G. Dissanayake, "Planning stable and efficient paths for reconfigurable robots on uneven terrain," *J. Intell. Robot. Syst.*, vol. 87, pp. 291–312, 2017.

[32] D. Kanoulas, C. Zhou, A. Nguyen, G. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Vision-based foothold contact reasoning using curved surface patches," in *IEEE Int. Conf. Humanoid Robot.* IEEE, 2017, pp. 121–128.

[33] N. Kita, M. Morisawa, and F. Kanehiro, "Foot landing state estimation from point cloud at a landing place," in *IEEE Int. Conf. Humanoid Robot.* IEEE, 2013, pp. 252–259.

[34] J. Kim, W. K. Chung, Y. Youm, and B. H. Lee, "Real-time zmp compensation method using null motion for mobile manipulators," in *IEEE Int. Conf. Robot. Autom.*, vol. 2. IEEE, 2002, pp. 1967–1972.

[35] S. A. A. Moosavian and K. Alipour, "On the dynamic tip-over stability of wheeled mobile manipulators," *Int. J. Robot. Autom.*, vol. 22, no. 4, p. 322, 2007.

[36] S. Fabian, S. Kohlbrecher, and O. Von Stryk, "Pose prediction for mobile ground robots in uneven terrain based on difference of heightmaps," in *IEEE Int. Symp. Safety, Secur. Rescue Robot. (SSRR).* IEEE, 2020, pp. 49–56.

[37] Y. Yuan, L. Wang, and S. Schwertfeger, "Configuration-space flipper planning for rescue robots," in *IEEE Int. Symp. Safety, Secur. Rescue Robot. (SSRR).* IEEE, 2019, pp. 37–42.

[38] C. Schulz, R. Hanten, and A. Zell, "Efficient map representations for multi-dimensional normal distributions transforms," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2018, pp. 2679–2686.

[39] B. Hepp, M. Nießner, and O. Hilliges, "Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction," *ACM Trans. Graph.*, vol. 38, no. 1, pp. 1–17, 2018.

[40] Z. Galil and G. F. Italiano, "Data structures and algorithms for disjoint set union problems," *ACM Comput. Surveys*, vol. 23, no. 3, pp. 319–344, 1991.

[41] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Soft.*, vol. 22, no. 4, pp. 469–483, 1996.

[42] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, 2022.