

# Efficient Global Navigational Planning in 3D Structures based on Point Cloud Tomography

Bowen Yang<sup>1</sup>, Jie Cheng<sup>1</sup>, Bohuan Xue<sup>1</sup>, *Graduate Student Member, IEEE*, Jianhao Jiao<sup>1,2</sup>, *Member, IEEE*, and Ming Liu<sup>3</sup>, *Senior Member, IEEE*

**Abstract**—Navigation in complex 3D scenarios requires appropriate environment representation for efficient scene understanding and trajectory generation. We propose a highly efficient and extensible global navigation framework based on a tomographic understanding of the environment to navigate ground robots in multi-layer structures. Our approach generates tomogram slices using the point cloud map to encode the geometric structure as ground and ceiling elevations. Then it evaluates the scene traversability considering the robot's motion capabilities. Both the tomogram construction and the scene evaluation are accelerated through parallel computation. Our approach further alleviates the trajectory generation complexity compared with planning in 3D spaces directly. It generates 3D trajectories by searching through multiple tomogram slices and separately adjusts the robot height to avoid overhangs. We evaluate our framework in various simulation scenarios and further test it in the real world on a quadrupedal robot. Our approach reduces the scene evaluation time by 3 orders of magnitude and improves the path planning speed by 3 times compared with existing approaches, demonstrating highly efficient global navigation in various complex 3D environments. The code is available at: [https://github.com/byangw/PCT\\_planner](https://github.com/byangw/PCT_planner).

**Index Terms**—Localization, mapping & planning, Unmanned autonomous systems, Applications (robotics).

## I. INTRODUCTION

Navigating ground robots in 3D environments is essential for a wide range of autonomous applications. However, it's still challenging to efficiently evaluate the multi-layer scenarios with complex terrain conditions and spatial structures, where a proper environmental representation would assist to improve the scene evaluation speed.

Point clouds and meshes can represent detailed 3D structures and are applied in robot navigation problems [1], [2], while their irregular data structures may bring difficulties to scene understanding. Voxels discretize the space into structured 3D grids for the convenience of construction and map processing, which are widely adopted to navigate UAVs [3], [4]. However, it's usually difficult to achieve high mapping efficiency while maintaining the capabilities to represent detailed environment structures. In addition, unlike drones that fly in

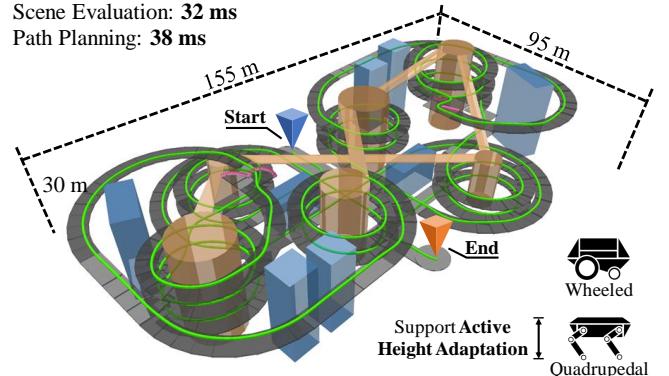


Fig. 1. We present a highly efficient and extensible global navigation framework that generates smooth 3D trajectories in complex multi-layer scenarios. It adopts a novel scene representation which enables rapid scene evaluation and further alleviates the burden of path searching. The framework applies to a wide range of ground robots including wheeled or quadrupedal robots and supports active height adaptation of the platform to avoid overhangs.

3D space, voxels might not be the optimal choice for ground robots that are more concerned about terrain conditions.

Elevation maps [5], [6] are widely adopted in navigating ground robots on complex uneven terrains. Compared with voxel-based representations, elevation maps store continuous ground height values in 2D grids to represent the terrain surface, which balances the mapping efficiency and the capability to represent detailed ground conditions. However, traditional elevation maps fail to identify overhangs or multi-layer structures, restricting their scope of application. Triebel *et al.* [7] extend elevation maps by storing multi-level surfaces into grid-wise lists, which brings difficulties in scene evaluation. Miki *et al.* [8] adopt exclusion areas to reject overhangs and apply overlap clearance to update the ground levels in multi-layer structures, restricting it to only represent a single local terrain surface and the method is unsuitable for large-scale scenarios.

We present a global navigation framework for ground robots in complex 3D scenarios with the following aspects contributing to its high efficiency. First, we propose a novel scene representation approach based on a tomographic understanding of the point cloud. It encodes the geometric structure into multiple tomogram slices containing the ground and ceiling elevations. Our approach maintains the mapping simplicity and terrain representation capability of elevation maps while extending their scope to large-scale multi-layer scenarios. In addition, our scene representation is compatible with a wide range of mapping, scene evaluation, and path planning methods on grid maps, making our framework highly extensible to

<sup>1</sup>B. Yang, J. Cheng, B. Xue, and J. Jiao are with the Hong Kong University of Science and Technology, Hong Kong SAR, China. {byangar, jchengai, bxueaa, jjiao}@connect.ust.hk (Corresponding author: Jianhao Jiao).

<sup>2</sup>J. Jiao is also with the Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK.

<sup>3</sup>M. Liu is with the Hong Kong University of Science and Technology (Guangzhou), Nansha, Guangzhou, 511400, Guangdong, China. eelium@hkust-gz.edu.cn

further enhancements. Second, a kernel-based scene evaluation method is designed for rapid traversability estimation on tomograms. It's aware of the navigation hazards from both ground and ceiling conditions considering the locomotion and height adjustment capabilities of the robot. Both the tomogram construction and the scene evaluation process are accelerated through parallel computation. Third, we reduce the burden of path planning by searching through multiple 2.5D tomogram slices, achieving higher efficiency than directly planning in 3D spaces. Our main contributions include:

- We propose a novel tomographic scene representation to understand 3D environments, extending elevation maps to multi-layer scenarios while maintaining their advantages in mapping efficiency and representation capabilities.
- We design a GPU-based tomographic map construction and scene evaluation method that reduces the computation time by 3 orders of magnitude while considering the robot's locomotion and height adjustment capabilities.
- We develop a trajectory generation approach that efficiently returns 3D trajectories with velocity information by searching and optimizing paths on tomogram slices, improving the path planning speed by 3 times.
- We integrate these modules to present a highly efficient navigation framework that improves the navigation speed by 2 orders of magnitude and successfully navigates a quadrupedal robot in real-world scenarios.

## II. RELATED WORK

In recent years, intensive research has been conducted on navigating ground robots in complex environments. The majority of these approaches can be classified into three categories which respectively solve the navigation problems on point clouds, meshes, and 2.5D or 3D grid maps.

### A. Navigating on Point Clouds

The 3D point clouds are extensively adopted in robotic applications to represent the environments [9]–[12]. Some approaches directly solve navigation problems on point cloud maps. Liu [1] proposes a GPU-accelerated tensor voting framework to evaluate the environment's geometric features on the point cloud and calculate geodesic vectors on 3D structures. It then adopts k-NN to construct a navigation graph on the point cloud and uses Dijkstra for path planning. Krüsi *et al.* [13] analyze the point distributions in local patches for terrain assessment and directly generate optimized trajectories on point clouds, enabling navigation on irregular terrains in multi-level facilities. Waibel *et al.* [14] evaluate the terrain roughness in off-road scenarios by applying a Convolutional-LSTM network on point cloud patches to predict the IMU responses. Although these methods are free of terrain surface reconstruction, the unordered point cloud data increases the complexity of scene understanding and trajectory generation, which may lead to low navigation efficiency.

### B. Navigating on Meshes

Meshes are widely used in computer graphics to model 3D structures as polygon surfaces, which are also used in 3D

navigation problems. Ruetz *et al.* [15] develop OVPC mesh which generates watertight 3D meshes from point clouds and classifies the traversable spaces by calculating the direction of surface normal vectors, enabling path planning under overhangs. Brandão *et al.* [16] propose GaitMesh for navigating quadrupedal robots in large-scale multi-floor structures. A distance field is built from voxelized polygons considering the properties of different gait controllers and a navigation mesh is then reconstructed for path searching using A\* [17]. Pütz *et al.* [18] plan paths on triangular meshes with arbitrary shapes in outdoor environments. It measures multiple geometric attributes of the mesh to generate the layered mesh map and adopts Fast Marching Method (FMM) to compute the travel costs through wavefront propagation. However, these methods still usually require a long computation time to generate and evaluate the navigation mesh in large-scale scenarios.

### C. Navigating on Grid Maps

Grid maps have high compatibility with a wide range of both traditional and learning-based approaches for mapping [19]–[22], scene evaluation [23]–[26], and path planning [27]–[30]. For example, elevation maps [5], [6] are 2.5D grid maps that represent terrain structures with continuous height values and are successfully applied to solve navigation problems on complex terrains in [27], [29]. Nevertheless, traditional elevation maps are unsuitable for scenarios with overhanging objects or multi-floor structures. Miki *et al.* [8] use ramp parameters to exclude the ceiling points and update the local map with overlap clearance when moving through multiple floors. Meng *et al.* [31] introduce a ceiling layer and two ground elevations to capture the overhanging objects, ground obstacles, and terrain conditions, which enables traveling in complex off-road environments. However, these approaches are still unsuitable for global navigation in multi-layer environments. Triebel *et al.* [7] record the height and thickness of various surface patches into the list of each grid to represent multi-layers scenarios, which may bring difficulties for the map processing stage to evaluate the scene from lists.

Some approaches represent the environment using 3D grids. Frey *et al.* [32] adopt a neural network for traversability estimation on occupancy voxels and plan paths in 3D structures with ceilings, while the voxels only provide a rough view of the terrain conditions. Wang *et al.* [2] extract traversable regions from the point cloud using a "Valid Ground Filter". It further obtains the travel costs by constructing Euclidean Signed Distance Field (ESDF) and performing local plane fitting. The trajectories are directly planned and optimized on dense voxels, which may increase the computational burden.

This paper proposes a navigation framework for ground robots in complex 3D environments using a tomographic scene representation based on 2.5D grid maps. Compared with existing extensions of elevation maps [7], [8], [31], our approach adapts to large-scale multi-layer scenarios while maintaining representation simplicity. Rather than directly solving the navigation problems on 3D representations, our approach achieves higher efficiency in map construction, scene evaluation, and trajectory generation than [1], [2], [18].

### III. METHODOLOGY

This section introduces the concept of point cloud tomography and the approach to constructing the tomogram slices. Then it illustrates how we achieve efficient scene understanding and trajectory generation using this new representation.

#### A. Tomogram Construction

We define the “tomogram slices”  $\{S_k \mid k \in [0, N]\}$  which are multi-channel grid maps with the resolution being  $r_g$ . Each slice  $S_k = \{e_k^G, e_k^C, c_k^T\}$  contains a ground  $e_k^G = \{e_{i,j,k}^G\}$  and a ceiling  $e_k^C = \{e_{i,j,k}^C\}$  elevation layer that respectively encode the terrain conditions and the overhanging structures, as well as a corresponding travel cost map  $c_k^T = \{c_{i,j,k}^T\}$ , where  $e, c$  are the elevation and cost values of each grid,  $i, j$  are the row and column indices,  $k$  is the slice index. To construct the tomogram, we analyze the point cloud from a series of cross-sections viewed from the relevant horizontal planes (Fig. 2). The planes are equidistantly stacked with the separation being  $d_s$ . The first plane is placed above the lowest point in the point cloud by  $d_s$  and the number of planes  $N$  is selected to ensure that the last plane is above the highest point.

Each plane splits the point cloud into the “lower” and the “upper” group containing the points below and above the plane. As shown in Algorithm 1 (line 1 to 6), to construct the ground layer  $e_k^G$  of the  $k$ -th slice, we vertically project each point in the “lower group” upwards to the plane and obtain its projection depth by calculating the height difference between the point and the plane. The projected points are rasterized to obtain a grid map. For each grid, the ground elevation  $e_{i,j,k}^G$  is the plane height minus the minimum projection depth among all the points inside. The ceiling  $e_k^C$  is obtained similarly by projecting the “upper group” downwards and  $e_{i,j,k}^C$  is the plane height plus the minimum projection depth. We set  $e_{i,j,k}^G$  or  $e_{i,j,k}^C$  to be invalid if no point is projected into the grid. In this way, these planes generate pairs of ground and ceiling layers for the respective tomogram slices, while the travel cost maps  $c_k^T$  are further obtained in Section III-B.

We note that in an indoor scenario, the building floors and our slices are not necessarily aligned, as a single floor may also contain various height layers. In addition, our approach applies to more complicated structures such as spiral overpasses or caves with intricate passages. To ensure complete coverage of the valid planning search space, we choose  $d_s \leq d_{min}$ , where  $d_{min}$  is the minimum ground-ceiling interval for the robot to move through. Therefore, all the places with sufficiently large intervals would be split by at least one plane. Then the traversable places will be recognized by evaluating the relevant ground and ceiling elevations in Section III-B.

#### B. Traversability Estimation

Next, we estimate the scene traversability and calculate the grid-wise travel costs by analyzing the ground and ceiling layers in the constructed tomogram (Algorithm 1 line 7, 8). Compared with existing scene evaluation methods using single terrain elevations, our approach is also aware of the overhang hazards and supports active body height adjustment of

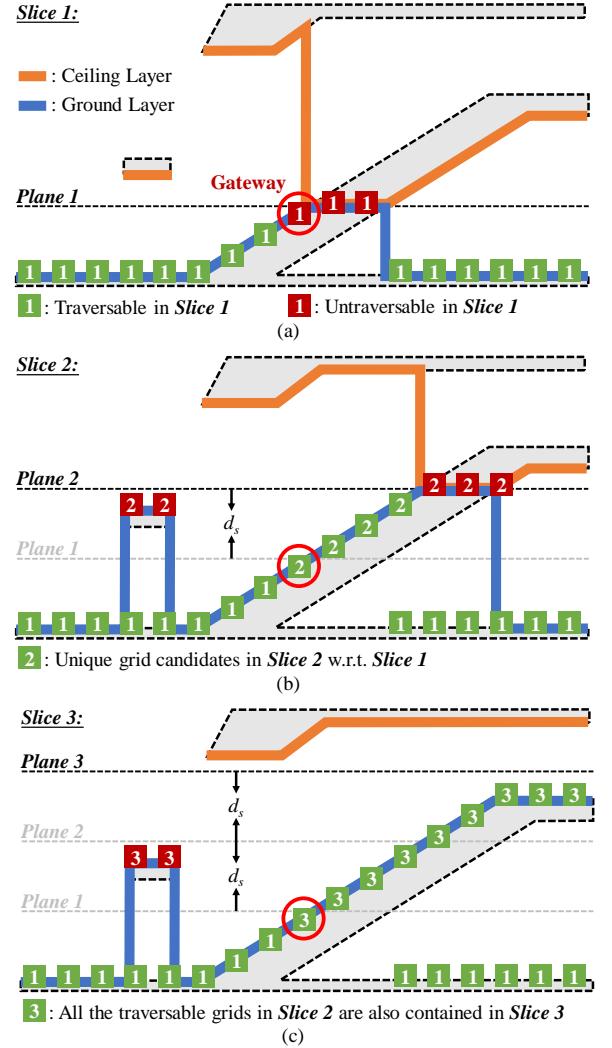


Fig. 2. We construct tomogram slices by projecting the point cloud onto a series of equidistant horizontal planes. Each slice contains a ceiling (orange) and a ground (blue) layer. The squares present the grid traversability (green: traversable, red: untraversable) considering the ground conditions and the ceiling height. As all the traversable grids in slice 2 are contained in the union of slices 1 and 3, slice 2 can be omitted as done in Section III-C. The red-circled grids are the “gateways” defined in Section III-D that connect slice 1 to the upper slices for searching upwards, as these grids share the same spatial position but the cost in the upper slice is lower and reflects the real traversability at that position. By using the gateways, our planner travels through multiple slices to search on the slope and under the overhangs, thus enabling navigation in multi-layer structures.

the robot. To achieve this, we calculate the height difference between each pair of ground and ceiling layers to get the interval distance  $d^I = e^C - e^G$ . Suppose the robot’s body height is adjustable between  $d_{min}$  and a normal operation height  $d_{ref}$ , a grid is considered untraversable if  $d^I < d_{min}$  or will be assigned a penalty for the additional height adjustment effort if  $d^I \in [d_{min}, d_{ref}]$ . For instance, we use the following interval cost map  $c^I$  for our quadrupedal robot as lowering the body height leads to more energy consumption while walking, where  $c^B$  is the cost for untraversable barriers,  $\alpha_d$  is a scaling factor, and each element  $c^I$  is obtained as follows:

$$c^I = \begin{cases} c^B & \text{if } d^I < d_{min} \\ \max(0, \alpha_d(d_{ref} - d^I)) & \text{otherwise} \end{cases}. \quad (1)$$

**Algorithm 1** Point Cloud Tomography

**Input:** global point cloud map  $P = \{\mathbf{p}_u | \mathbf{p}_u = [x_u, y_u, z_u]^T\}$   
**Output:** tomogram slices  $S = \{S_k | S_k = (\mathbf{e}_k^G, \mathbf{e}_k^C, \mathbf{c}_k^T)\}$

- 1: Minimum height of points  $z_{min} = \min\{z_u\}$
- 2: **for** each point  $\mathbf{p}_u = [x_u, y_u, z_u]^T$  **do**
- 3:   Grid index  $i, j = \text{rasterize}(x_u, y_u)$
- 4:   **for** slice index  $k = 0, 1, \dots, N$  **do**
- 5:      $e_{i,j,k}^G = \max(z_u, e_{i,j,k}^G)$  **if**  $z_u \geq z_{min} + kd_s$
- 6:      $e_{i,j,k}^G = \min(z_u, e_{i,j,k}^G)$  **if**  $z_u < z_{min} + kd_s$
- 7:      $\mathbf{c}_k^{init} = \text{travEstm}(\mathbf{e}_k^G, \mathbf{e}_k^C)$  (Eq. 1 to 5)
- 8:      $\mathbf{c}_k^T = \text{inflation}(\mathbf{c}_k^{init})$  (Eq. 6)
- 9:   **for** each slice  $S_k = (\mathbf{e}_k^G, \mathbf{e}_k^C, \mathbf{c}_k^T)$  in  $S = \{S_k\}$  **do**
- 10:     Find unique grids  $U_k = \{\text{unique}(\mathbf{e}_k^G, \mathbf{c}_k^T)\}$  (Eq. 8, 9)
- 11:     **if**  $\text{size}(U_k) = 0$ , **remove**  $S_k$  from  $S$
- 12: **return** unique tomogram slices  $S$

By integrating  $c^I$  into the cost map, our planner generates cost-optimal 3D trajectories in Section III-D and III-E considering the overhangs even though the robot's motion for height adjustment is decomposed from planning on the ground surfaces.

Next, we analyze the ground layers to obtain the cost maps of terrain conditions  $\mathbf{c}^G$ . Compared with [2], our approach also identifies traversable steps and stairs to navigate legged robots. For each grid, we obtain the gradients  $[g^x, g^y]^T$  of the ground elevation  $e^G$  in  $x, y$  directions through the finite-difference method and get the following magnitudes:

$$m^{xy} = \max(|g^x|, |g^y|), \quad m^{grad} = \sqrt{(g^x)^2 + (g^y)^2}, \quad (2)$$

and we have  $m^{xy} \leq m^{grad}$ . Three criteria with thresholds  $\theta_b, \theta_s, \theta_p$  are applied to measure the terrain conditions. The grid is considered the boundary of a barrier if  $m^{xy} > \theta_b$  or a traversable gentle surface if  $m^{grad} < \theta_s$ , and the cost element:

$$c^G = \begin{cases} c^B & \text{if } m^{xy} > \theta_b \\ \alpha_s \left( \frac{m^{grad}}{\theta_s} \right)^2 & \text{if } m^{grad} < \theta_s \end{cases}, \quad (3)$$

where  $\alpha_s$  is a scaling factor. Otherwise, the grid belongs to an edge that might be untraversable for wheeled robots (set  $c^G = c^B$ ) but can still be stepped across by legged robots. We further estimate if the surrounding grids are safe to step on by calculating the percentage  $p_s$  of neighboring grids with  $m^{grad} < \theta_s$  within a local patch. The grid is considered traversable if  $p_s > \theta_p$  and the cost is formulated as:

$$c^G = \begin{cases} \alpha_b \left( \frac{m^{xy}}{\theta_b} \right)^2 & \text{if } p_s > \theta_p \\ c^B & \text{otherwise} \end{cases}. \quad (4)$$

As shown in Fig. 3 (b), the grids near the center of the spiral stairs are untraversable as the surfaces are too narrow. The initial cost map is the clipped sum of  $c^I$  and  $c^G$ :

$$\mathbf{c}^{init} = \min(c^B, c^I + c^G). \quad (5)$$

Finally, we apply an inflation kernel (Fig. 3 (a)) on  $\mathbf{c}^{init}$  to expand the untraversable regions by  $d_{inf} \geq r_c$  considering the robot's collision radius  $r_c$  and create a safe margin within  $d_{sm}$  for smooth cost gradients and safe navigation behaviors.

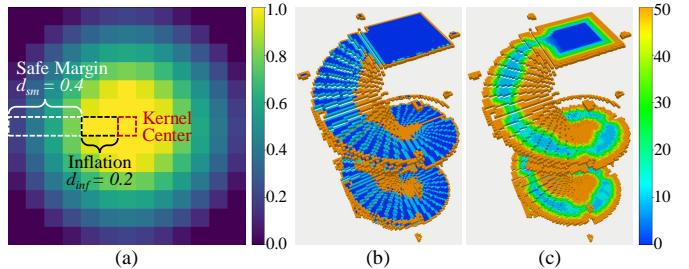


Fig. 3. (a): Illustration of the inflation kernel when  $d_{inf} = 0.2$ ,  $d_{sm} = 0.4$  and  $r_g = 0.1$ . (b): The traversability cost map before inflation, where the untraversable grids are in orange and the traversable regions are in blue. The regions near the spiral center are untraversable due to the insufficient stair width. (c): The final travel cost map, where the untraversable regions are inflated and the costs are gradually reduced within the safe margin after applying the inflation kernel.

The weight  $K(m, n)$  of a kernel cell at  $(m, n)$  with resolution  $r_g$  is calculated based on the Euclidean distance  $d_{mn}$  between the kernel center and the cell:

$$K(m, n) = \max \left( 0, \min \left( 1 - \frac{d_{mn} - d_{inf}}{d_{sm} - r_g}, 1 \right) \right). \quad (6)$$

The final travel cost map  $\mathbf{c}_k^T$  is obtained through the sliding window method, where the kernel performs the Hadamard product with the patches on  $\mathbf{c}_k^{init}$  and returns the maximum value of each resulting matrix as the cost of the patch center.

### C. Tomogram Simplification

During the tomogram construction, the planes are separated by  $d_s = d_{min}$  to guarantee complete coverage of the planning space. However, such a small height increment brings quite a limited expansion of the mapping region in the new layer, resulting in low efficiency in path searching or map storage. Therefore, the tomogram is further simplified (Algorithm 1 line 9 to 11) based on the following principle: Let  $M_k$  denote the set containing all the traversable grids in slice  $S_k$ . If:

$$M_k \subset (M_{k-1} \cup M_{k+1}), \quad (7)$$

which means the search space of an intermediate slice  $S_k$  is already included in the union of its previous and subsequent slice, then  $S_k$  is redundant and can be omitted. If so,  $S_{k+1}$  will become the new intermediate slice and we then check if  $M_{k+1} \subset (M_{k-1} \cup M_{k+2})$ . Otherwise,  $S_k$  will be preserved and we continue to examine if  $M_{k+1} \subset (M_k \cup M_{k+2})$ . The process is repeated until all the slices are checked.

The condition in Eq. 7 can be checked according to the ground elevations and the travel costs. A traversable grid with  $c_{i,j,k}^T < c_{i,j,k-1}^T$  at position  $(i, j, k)$  is considered “unique” if:

$$(c_{i,j,k}^G - c_{i,j,k-1}^G > 0 \text{ or } c_{i,j,k}^T < c_{i,j,k-1}^T) \text{ and} \quad (8)$$

$$(c_{i,j,k+1}^G - c_{i,j,k}^G > 0 \text{ or } c_{i,j,k}^T < c_{i,j,k+1}^T), \quad (9)$$

where Eq. 8 indicates that the grid has a unique spatial position or has the same position but a lower cost that reflects its real traversability compared with the grid at  $(i, j, k-1)$ . Similarly, Eq. 9 checks the relationship with the grid in  $S_{k+1}$ . In this way, we examine the uniqueness of all the grids in each slice and the slices containing no unique grid will be omitted. For instance, in Fig. 2 (b) the traversable grids of slice 2 on the slope (green

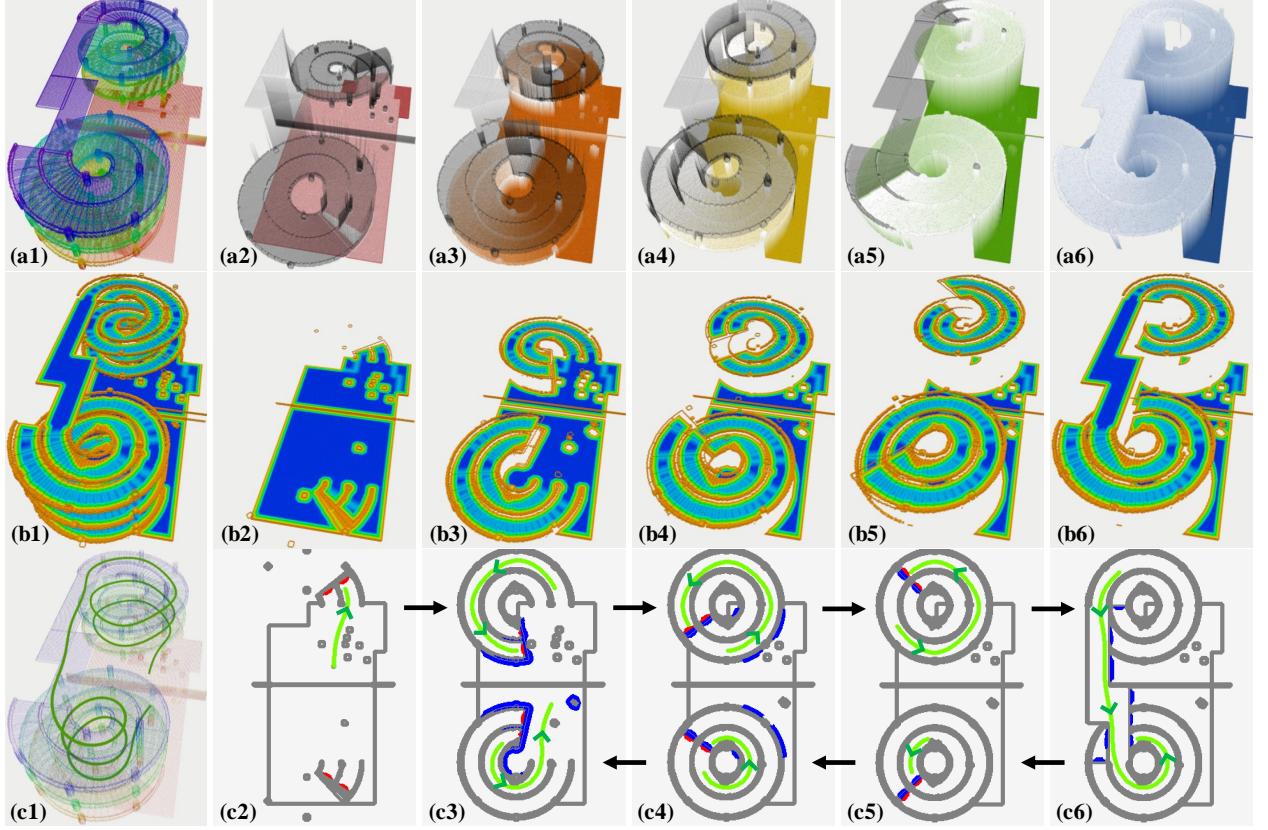


Fig. 4. (row a): The point cloud map (a1) of the *spiral* environment in [2] is used to construct our tomogram. The environment is then represented by five 2.5D tomogram slices (a2) to (a6) after simplification, where the gray layers are ceilings and the colored layers are ground elevations. (row b): The whole planning space (visualized as a point cloud (b1), blue: traversable, orange: untraversable) can be obtained by integrating (b2) to (b6) which are 2.5D travel cost maps with the related ground elevations in (a2) to (a6). Note that (b1) is only for visualization and we directly adopt (b2) to (b6) for trajectory generation in Section III-D and III-E. (row c): The planner searches on multiple tomogram slices along the green arrows and generates the 3D trajectory. It can enter the upper slices through the gateways in red and can also search downwards through the gateways in blue.

grids with a mark “2”) satisfy Eq. 8 and are considered as candidates of unique grids w.r.t. slice 1. However, they violate Eq. 9 as they have the same elevations and costs as the grids in slice 3. Therefore, slice 2 can be omitted.

Fig. 4 shows the simplified tomogram (row a) of the *spiral* scenario in [2] and the corresponding travel cost maps (row b). The point cloud is initially projected onto 46 slices which are then simplified to 5 slices (Fig. 4 (a2) to (a6)) using the above procedures. Compared with storing travel costs into dense 3D voxels as done in [2], our approach achieves higher efficiency by only adopting 5 layers of 2.5D cost maps (Fig. 4 (b2) to (b6)) to represent the whole planning space. The memory usage of our tomogram has the potential to be further reduced by adopting sparse representations or other data structures to store the unique grids only. In this paper, we continue to use multi-layer 2.5D grid maps to represent the environment for the simplicity of the subsequent processing stages.

#### D. Path Planning through Slices

After obtaining the simplified travel cost maps, we modify A\* to search through tomogram slices and plan initial paths. Rather than directly planning on dense voxels in [2], we separately plans on the 2.5D maps and optimizes  $z$ -axis motions to avoid overhangs, achieving higher efficiency. As the cost map already contains the interval cost  $c^I$ , our planner generates optimal solutions considering the ground-ceiling intervals.

Given an initial and a goal position, the planner starts searching on an arbitrary slice  $S_k$  that contains the initial position. Each grid is considered a graph node that connects to its eight neighbors on the same cost map. When a grid is queried, we also check the two grids from the adjacent slices  $S_{k-1}, S_{k+1}$  at the same planimetric position  $(i, j)$ . Taking the grid above at  $(i, j, k + 1)$  as an example, if they share the same ground elevation  $e_{i,j,k}^G = e_{i,j,k+1}^G$ , then they are considered the same node in the 3D space with the node cost  $c^N = \min(c_{i,j,k}^T, c_{i,j,k+1}^T)$ . The cost for graph search between two nodes is defined as  $c^N$  of the target node plus their Euclidean distance, where the connection breaks if  $c^N = c^B$ . The heuristic cost is the diagonal distance from the queried node to the goal. If  $c_{i,j,k+1}^T < c_{i,j,k}^T$ , then this node is considered as a “gateway” (e.g., red-circled grids in Fig. 2) to travel from  $S_k$  to  $S_{k+1}$  so that the planner can continue to search on  $S_{k+1}$ . The same logic adapts to the grid below at  $(i, j, k - 1)$  when searching downwards, where the gateway occurs if  $e_{i,j,k-1}^G = e_{i,j,k}^G$  and  $c_{i,j,k-1}^T < c_{i,j,k}^T$ . In this way, our approach plan paths on different slices and connect the paths at the gateways to generate a 3D result on the multi-layer structure. Fig. 4 (row c) presents how our planner travels on multiple slices along the green arrows. The planner travels to the upper slices through the gateway grids in red and enters the lower slices through the gateway grids in blue.

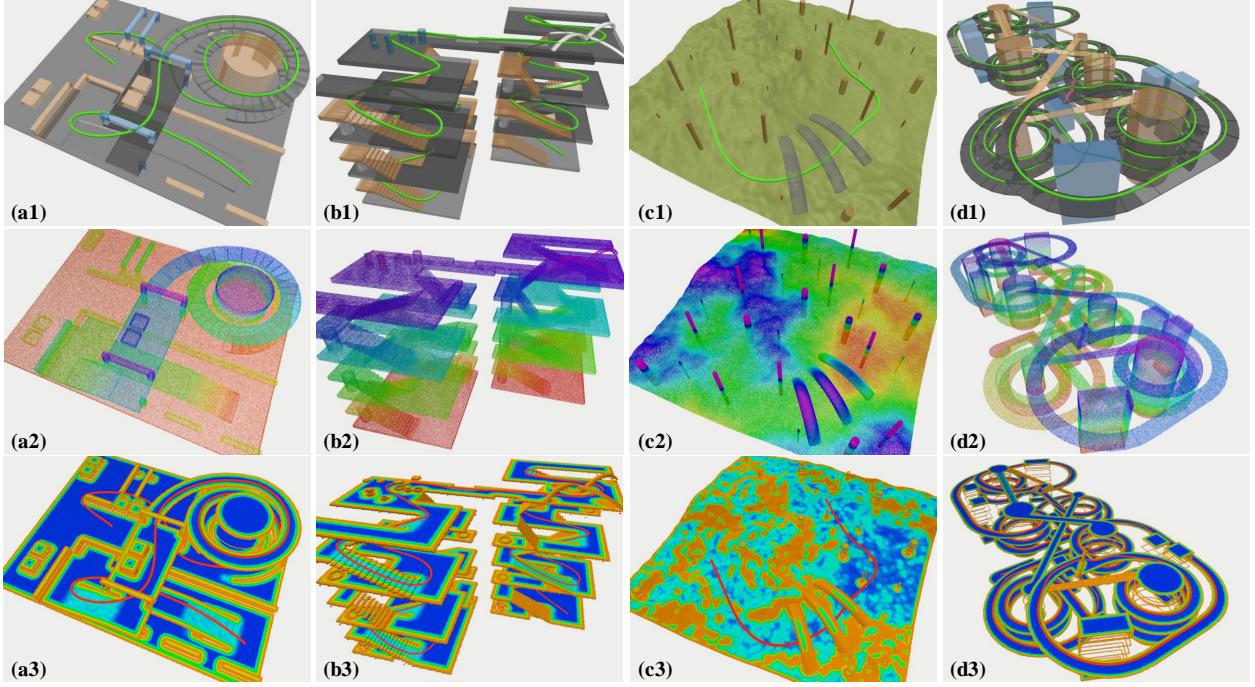


Fig. 5. The simulation scenarios and the trajectories generated by our approach. The first row presents the 3D models of environments: *factory* (a1), *building* (b1), *forest* (c1), and *overpass* (d1). The second row shows the point clouds of the corresponding scenarios and the third row presents the integrated travel cost maps generated by our approach. Our approach provides feasible and smooth trajectories in these complex 3D scenarios with various structured or irregular terrain features and overhanging objects.

### E. Trajectory Optimization

The proposed scene representation is compatible with various gradient-based trajectory optimization techniques. In this paper, we represent the trajectory as  $M$ -piece 3-dimensional polynomials. The  $i$ -th piece of the trajectory is defined by a polynomial with a degree of  $N = 5$ :

$$\mathbf{q}_i(t) = \boldsymbol{\sigma}_i^T \boldsymbol{\beta}(t), \forall t \in [0, T_i], \quad (10)$$

where  $\boldsymbol{\sigma} \in \mathbb{R}^{(N+1) \times m}$  is the coefficient matrix and  $\boldsymbol{\beta} = [1, t, \dots, t^N]^T$  is the natural bias. The trajectory optimization is formulated as a minimal control effort problem:

$$\min_{\boldsymbol{\sigma}, T} J_c + w_z \|q_z(t) - Z_{ref}(\mathbf{q}(t))\|_2 + w_T T \quad (11a)$$

$$s.t. \quad \mathbf{q}_1(0) = \bar{\mathbf{q}}_0, \quad \mathbf{q}_M(T) = \bar{\mathbf{q}}_f, \quad (11b)$$

$$\mathbf{q}_i^{[3]}(T_i) = \mathbf{q}_{i+1}^{[3]}(0), \quad i = \{1, \dots, M-1\}, \quad (11c)$$

$$\mathcal{C}(\mathbf{q}(t)) \geq C_{safe}, \quad \forall t \in [0, T], \quad (11d)$$

$$\mathcal{G}(q(t), \dots, q^{(2)}(t)) \preceq \mathbf{0}, \quad \forall t \in [0, T], \quad (11e)$$

$$\mathcal{H}_g(\mathbf{q}(t)) \leq q_z(t) \leq \mathcal{H}_c(\mathbf{q}(t)), \quad \forall t \in [0, T], \quad (11f)$$

where  $J_c$  is the jerk control effort to smooth the trajectory in the 3D space following [33], [34],  $Z_{ref}(\mathbf{q}(t)) = e^G(\mathbf{q}(t)) + d_{ref}$  is the preferred operation height of the robot,  $\mathcal{C}$  is the travel cost generated in Section III-B,  $C_{safe}$  is the safety margin, and  $\mathcal{G}(\cdot)$  are kinematic constraints on the maximum value of velocity, acceleration, and the changing rate of headings. For ground robots with adjustable heights, additional maximum height constraints are imposed on the  $z$ -axis of the trajectory, where  $\mathcal{H}_g(\mathbf{q}(t)) = e^G(\mathbf{q}(t)) + d_{min}$  and  $\mathcal{H}_c(\mathbf{q}(t))$  is the inflated ceiling elevation queried from the tomogram.

## IV. EXPERIMENTS

### A. Implementation Details

1) *Evaluation scenarios*: The navigation approaches are evaluated in simulated and real-world scenarios with complex 3D structures. In simulation, the following environments are prepared to analyze the navigation performance, where the dimensions are presented in (*length, width, height*):

- *Factory*: An outdoor scenario with regular urban terrain features and overhanging structures (Fig. 5 (a1)). The scene dimension is  $(84 \times 68 \times 12)$ m.
- *Building*: An indoor scenario containing multiple floors with stairs and slopes of different steepness (Fig. 5 (b1)). The scene dimension is  $(22 \times 20 \times 16)$ m.
- *Forest*: A wilderness environment with a tunnel, irregular terrain, and thin obstacles (Fig. 5 (c1)). The scene dimension is  $(40 \times 40 \times 7)$ m.
- *Overpass*: A large-scale spiral overpass with multiple layers and a complex route (Fig. 5 (d1)). The scene dimension is  $(155 \times 95 \times 30)$ m.

The resolution of scene representations is 0.2 in *factory*, *overpass* and 0.1 in *building*, *forest* to capture the terrain features. To validate the trajectories, we simulate a Pioneer 3-DX robot in CoppeliaSim [35] for *factory*, *forest*, and *overpass*. In addition, to evaluate the trajectory generation performance, we uniformly sample 50 goals that are 26.5m away from the starting point at the map center in each of the following scenarios:

- *Plaza*: An outdoor scenario with various urban structures (Fig. 6 (a1)). The scene dimension is  $(56 \times 56 \times 5)$ m.
- *Hills*: A wilderness scenario with irregular obstacles and rough terrain (Fig. 6 (b1)). Dimension:  $(60 \times 60 \times 3)$ m.

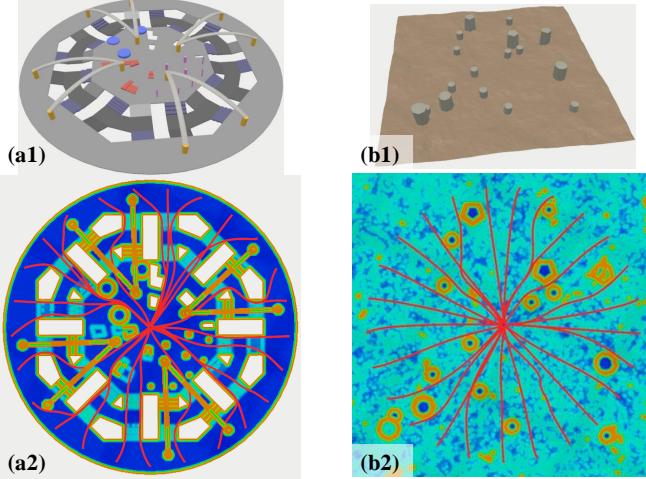


Fig. 6. The *plaza* (a1) scenario with multiple urban structures and the *hills* (b1) scenario with irregular terrain and obstacles for the evaluation of trajectory quality. (b1), (b2): example trajectories generated by our approach.

In real-world experiments, we scan the following environments with LiDARs and build the point cloud maps using A-LOAM<sup>1</sup> [36], where we manually remove the noise points and dynamic objects for rational navigation behaviors:

- *Stairs*: A multi-layer scenario where the robot starts from the ground floor and moves up the winding stairs (Fig. 9 (a1)). The scene is scanned by a Livox Mid-70. The scene dimension is  $(14 \times 14 \times 7)$ m.
- *Boxes*: A sandbox where the robot moves through narrow arches and reaches the upper platform from the stairs or the slope (Fig. 9 (b1)). The scene is scanned by an RS-Helios-32 onboard the robot. Dimension:  $(22 \times 22 \times 4)$ m.

We use a Jueying Mini [37] quadrupedal robot to validate the trajectories generated by the proposed framework. The algorithm parameters are set considering the motion capabilities of robots. For the quadrupedal robot used in the experiments, the parameter values in Table I are adopted for map construction and traversability estimation. For wheeled robots, we can set  $\theta_p = 1.0$  to disable planning over stairs or steps.

TABLE I  
ALGORITHM PARAMETERS

| Items                     | Notations                             | Values             |
|---------------------------|---------------------------------------|--------------------|
| Height Adaptation         | $[d_s, d_{min}, d_{ref}]$             | [0.50, 0.50, 0.65] |
| Thresholds                | $[\theta_b, \theta_s, \theta_p]$      | [1.70, 0.36, 0.20] |
| Costs and Scaling Factors | $[c^B, \alpha_d, \alpha_b, \alpha_s]$ | [50, 20, 20, 15]   |

2) *Baseline approaches*: Four methods with the following typical scene representations are implemented to compare the navigation capabilities with our approach. The last three methods which plan in multi-layer 3D structures are further quantitatively evaluated to compare the navigation efficiency in mapping, scene evaluation, and trajectory generation:

- *Elevation maps*: Yang's method [29] is adopted to evaluate traditional elevation maps [8] on GPU and navigate quadrupedal robots on complex terrains using A\*.
- *Points*: Liu's approach [1] is used to perform GPU-accelerated tensor voting directly on point clouds for

<sup>1</sup><https://github.com/HKUST-Aerial-Robotics/A-LOAM>

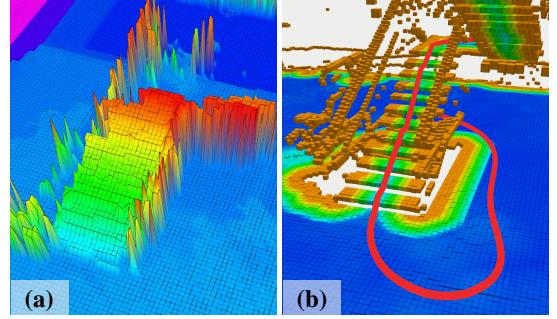


Fig. 7. (a): Yang's method [29] using traditional elevation maps [8] only evaluates a single terrain layer at a time and fails to plan in *stairs*. (b): Our approach extends elevation maps to represent 3D planning spaces and generates trajectory from the place below the stairs to the platform above.

scene evaluation. The path search algorithm in the original implementation is replaced by A\* for better efficiency.

- *Meshes*: Pütz's method [18] is adopted to evaluate the attributes of meshes and generate the layered navigation mesh map. The meshes are built from point clouds using the Ball-Pivoting algorithm. Then we also use A\* to find paths on the mesh maps for comparison.
- *Voxels*: We use Wang's method<sup>2</sup> [2] which processes the point clouds into ESDF maps. The initial paths are obtained by searching on 3D grids using A\* and then optimized to generate smooth trajectories.

We evaluate the navigation efficiency by comparing the time  $T_p$  for pre-processing the point cloud map to construct the scene representations as well as the *Size* of constructed maps that reflect the memory usage, the time  $T_e$  for traversability evaluation and obtaining the navigation cost maps, the time  $T_s$  for searching the initial paths, the number of nodes  $N_s$  that the graph traversal algorithm visits, the time  $T_o$  for trajectory optimization, and the total time  $T_{all}$  to solve the navigation problem. The operations on GPU are synchronized for accurate measurement of the computation time. The time of Wang's method to build 3D occupancy grids is excluded from  $T_p$  and  $T_{all}$  for fair comparison as its CPU implementation is inefficient. For trajectory evaluation in *plaza* and *hills*, we compare the average time for path searching  $\bar{T}_s$  and optimization  $\bar{T}_o$ , the average trajectory length  $\bar{L}_t$  and curvature  $\bar{C}_t$ , and the success rate  $S_t$  of finding feasible trajectories. All the approaches are evaluated on a desktop with an Intel i9-12900KF @ 3.2GHz CPU and NVIDIA RTX 3080 Ti GPU. In addition, we deploy our approach on an NVIDIA Jetson AGX Orin (MAXN mode) to further demonstrate the high efficiency of our approach even on a mobile computation device.

TABLE II  
SCENE EVALUATION AND TRAJECTORY GENERATION CAPABILITIES

| Methods     | Planning in 3D Spaces | Body Height Adaptation | Planning on the Stairs | Motion Capabilities Awareness | Trajectory with Velocities |
|-------------|-----------------------|------------------------|------------------------|-------------------------------|----------------------------|
| Yang's [29] | ✗                     | ✗                      | ✓                      | ✓                             | ✗                          |
| Liu's [1]   | ✓                     | ✗                      | ✓                      | ✗                             | ✗                          |
| Pütz's [18] | ✓                     | ✗                      | ✗                      | ✓                             | ✗                          |
| Wang's [2]  | ✓                     | ✓                      | ✗                      | ✓                             | ✓                          |
| Ours        | ✓                     | ✓                      | ✓                      | ✓                             | ✓                          |

<sup>2</sup><https://github.com/ZJU-FAST-Lab/3D2M-planner>

TABLE III  
EVALUATION OF COMPUTATION AND MEMORY EFFICIENCY IN MAP CONSTRUCTION, SCENE EVALUATION, AND TRAJECTORY GENERATION

| Scenarios       | Approaches         | Map Construction    |               | Scene Evaluation     |                     | Trajectory Generation |                              |                              | Overall |
|-----------------|--------------------|---------------------|---------------|----------------------|---------------------|-----------------------|------------------------------|------------------------------|---------|
|                 |                    | $T_p$ [ms] ↓        | $Size$ [MB] ↓ | $T_e$ [ms] ↓         | $T_s$ [ms] ↓        | $N_s$ ↓               | $T_o$ [ms] ↓                 | $T_{all}$ [ms] ↓             |         |
| <i>Factory</i>  | Liu's [1]          | —                   | 12.31         | $178.28 \times 10^3$ | 189.50              | 135,290               | —                            | $178.75 \times 10^3$         |         |
|                 | Pütz's [18]        | $3.44 \times 10^3$  | 28.84         | $13.36 \times 10^3$  | 100.94              | <b>80,333</b>         | —                            | $16.91 \times 10^3$          |         |
|                 | Wang's [2]         | $2.22 \times 10^3$  | 34.27         | $24.54 \times 10^3$  | $1.37 \times 10^3$  | 3,892,807             | 953.46                       | $29.08 \times 10^3$          |         |
|                 | Ours (PC-CPU only) | 630.76              | <b>4.57</b>   | 785.68               | <b>25.66</b>        | 133,648               | <b>475.64</b>                | $1.92 \times 10^3$           |         |
|                 | Ours (PC)          | <b>2.39</b>         | <b>4.57</b>   | <b>2.41</b>          | <b>25.66</b>        | 133,648               | <b>475.64</b>                | <b>542.55</b>                |         |
|                 | Ours (Orin)        | 20.06               | <b>4.57</b>   | 12.53                | 75.56               | 133,648               | $1.36 \times 10^3$           | $1.56 \times 10^3$           |         |
| <i>Building</i> | Liu's [1]          | —                   | 7.40          | $116.47 \times 10^3$ | 765.54              | 208,973               | —                            | $117.51 \times 10^3$         |         |
|                 | Pütz's [18]        | $3.41 \times 10^3$  | 17.10         | $11.36 \times 10^3$  | —                   | —                     | —                            | —                            |         |
|                 | Wang's [2]         | $1.91 \times 10^3$  | 28.16         | $11.53 \times 10^3$  | —                   | —                     | —                            | —                            |         |
|                 | Ours (PC-CPU only) | 290.98              | <b>3.17</b>   | 234.95               | <b>7.33</b>         | <b>39,760</b>         | <b>359.39</b>                | 892.65                       |         |
|                 | Ours (PC)          | <b>3.10</b>         | <b>3.17</b>   | <b>3.42</b>          | <b>7.33</b>         | <b>39,760</b>         | <b>359.39</b>                | <b>398.14</b>                |         |
|                 | Ours (Orin)        | 23.05               | <b>3.17</b>   | 17.69                | 17.89               | <b>39,760</b>         | 989.66                       | $1.11 \times 10^3$           |         |
| <i>Forest</i>   | Liu's [1]          | —                   | 6.94          | $93.86 \times 10^3$  | 66.24               | <b>29,234</b>         | —                            | $94.18 \times 10^3$          |         |
|                 | Pütz's [18]        | $4.52 \times 10^3$  | 17.94         | $12.88 \times 10^3$  | 68.27               | 57,162                | —                            | $17.47 \times 10^3$          |         |
|                 | Wang's [2]         | $1.79 \times 10^3$  | 44.80         | $24.67 \times 10^3$  | 917.61              | 3,014,581             | <b>42.95</b>                 | $27.42 \times 10^3$          |         |
|                 | Ours (PC-CPU only) | 293.43              | <b>6.40</b>   | 460.51               | <b>19.56</b>        | 85,450                | 65.43                        | 838.93                       |         |
|                 | Ours (PC)          | <b>1.42</b>         | <b>6.40</b>   | <b>3.02</b>          | <b>19.56</b>        | 85,450                | 65.43                        | <b>109.99</b>                |         |
|                 | Ours (Orin)        | 12.50               | <b>6.40</b>   | 19.88                | 60.50               | 85,450                | 175.75                       | 317.40                       |         |
| <i>Overpass</i> | Liu's [1]          | —                   | <b>16.35</b>  | $1.19 \times 10^6$   | 820.49              | 596,386               | —                            | $1.19 \times 10^6$           |         |
|                 | Pütz's [18]        | $18.30 \times 10^3$ | 67.73         | $45.21 \times 10^3$  | 840.81              | 399,943               | —                            | $64.35 \times 10^3$          |         |
|                 | Wang's [2]         | $4.56 \times 10^3$  | 220.88        | $97.64 \times 10^3$  | $19.81 \times 10^3$ | 55,489,651            | $5.24 \times 10^3$           | $127.26 \times 10^3$         |         |
|                 | Ours (PC-CPU only) | $1.49 \times 10^3$  | 26.65         | $3.30 \times 10^3$   | <b>37.82</b>        | <b>224,651</b>        | <b>2.91 × 10<sup>3</sup></b> | $7.74 \times 10^3$           |         |
|                 | Ours (PC)          | <b>18.11</b>        | 26.65         | <b>14.35</b>         | <b>37.82</b>        | <b>224,651</b>        | <b>2.91 × 10<sup>3</sup></b> | <b>3.17 × 10<sup>3</sup></b> |         |
|                 | Ours (Orin)        | 123.55              | 26.65         | 81.78                | 92.59               | <b>224,651</b>        | $7.55 \times 10^3$           | $8.19 \times 10^3$           |         |
| <i>Stairs</i>   | Liu's [1]          | —                   | 1.34          | $10.46 \times 10^3$  | 67.68               | 43,302                | —                            | $10.75 \times 10^3$          |         |
|                 | Pütz's [18]        | $3.19 \times 10^3$  | 3.69          | $3.36 \times 10^3$   | —                   | —                     | —                            | —                            |         |
|                 | Wang's [2]         | 346.40              | 5.49          | $2.58 \times 10^3$   | —                   | —                     | —                            | —                            |         |
|                 | Ours (PC-CPU only) | 55.41               | <b>0.47</b>   | 58.10                | <b>2.06</b>         | <b>12,426</b>         | <b>54.29</b>                 | 169.86                       |         |
|                 | Ours (PC)          | <b>0.42</b>         | <b>0.47</b>   | <b>0.20</b>          | <b>2.06</b>         | <b>12,426</b>         | <b>54.29</b>                 | <b>61.23</b>                 |         |
|                 | Ours (Orin)        | 3.23                | <b>0.47</b>   | 1.36                 | 6.03                | <b>12,426</b>         | 132.19                       | 161.50                       |         |
| <i>Boxes</i>    | Liu's [1]          | —                   | 2.75          | $22.56 \times 10^3$  | 19.00               | 5,935                 | —                            | $22.82 \times 10^3$          |         |
|                 | Pütz's [18]        | $1.62 \times 10^3$  | 5.10          | $1.61 \times 10^3$   | 3.91                | <b>2,875</b>          | —                            | $3.23 \times 10^3$           |         |
|                 | Wang's [2]         | 935.63              | 7.74          | $3.55 \times 10^3$   | 90.94               | 245,412               | 16.00                        | $4.60 \times 10^3$           |         |
|                 | Ours (PC-CPU only) | 107.55              | <b>1.16</b>   | 18.70                | <b>1.64</b>         | <b>7,794</b>          | <b>14.62</b>                 | 142.51                       |         |
|                 | Ours (PC)          | <b>0.37</b>         | <b>1.16</b>   | <b>0.95</b>          | <b>1.64</b>         | <b>7,794</b>          | <b>14.62</b>                 | <b>24.34</b>                 |         |
|                 | Ours (Orin)        | 2.93                | <b>1.16</b>   | 5.52                 | 4.53                | <b>7,794</b>          | 41.65                        | 75.72                        |         |

Evaluation results of the navigation approaches in simulated (row 1 to 4) and real-world (row 5, 6) scenarios.  $T_p$ : point cloud pre-processing time,  $Size$ : memory usage of the constructed scene representation,  $T_e$ : traversability estimation time,  $T_s$ : path searching time,  $N_s$ : visited graph nodes for planning,  $T_o$ : trajectory optimization time,  $T_{all}$ : total navigation time.

### B. Navigation Capabilities

The first four items in Table II compare the scene evaluation capabilities of different methods. Although Yang's method [29] plans with motion capabilities awareness of robots on stairs and other complex terrains, it fails to plan in multi-layer 3D spaces as the traditional elevation map [8] only recognizes a single terrain layer at a time (Fig. 7). Other approaches can navigate in 3D multi-layer scenarios. However, only Wang's and our approach can adjust the robot's body height along the  $z$ -axis for active adaptation to narrow environments. Both Pütz's and Wang's methods are unsuitable for planning on the stairs for legged robots, for they consider the vertical surfaces of the steps as untraversable. Although Liu's approach plans on the stairs, it fails to capture the motion capabilities of different robots and may generate infeasible paths. The last item in Table II compares the trajectory generation capability. Only Wang's and our approach generate trajectories with velocity information, as it's normally easier to optimize paths on grid maps. While Pütz *et al.* [18] also present the Continuous Vector-field planner (CVP) to generate smooth paths, it does

not plan the robot's velocities. Our approach provides smooth trajectories considering the robot's capabilities of locomotion and body height adjustment in complex environments, presenting strong potential in a wide range of navigation applications.

### C. Simulation Results

The first four rows of Table III present the evaluation results in simulation scenarios. For *building*, both Pütz's and Wang's methods can not recognize the traversable stairs, thus resulting in the failure of planning. As our map construction and scene evaluation steps are able to be accelerated through parallel computation, the processing speed is significantly improved by 2 to 3 orders of magnitude with less  $T_p$  and  $T_e$ . The constructed maps also maintain high efficiency in memory usage with smaller  $Size$  using our proposed representation approach. Although Liu's approach also runs on GPU, the tensor voting process is still quite time-consuming. Also, Wang's method takes a long time to perform plane fitting, resulting in large  $T_e$ . In addition, our approach reduces the burden of path searching with less time  $T_s$ , which benefits the tasks with re-planning requirements. Wang's method directly plans on dense

TABLE IV  
EVALUATION OF TRAJECTORY GENERATION PERFORMANCE IN RUN-TIME AND QUALITY

| Scenarios | Approaches  | $\bar{T}_s$ [ms] ↓            | $\bar{T}_o$ [ms] ↓ | $\bar{L}_t$ [m] ↓ | $\bar{C}_t$ [m $^{-1}$ ] ↓ | $S_t$ ↑ |
|-----------|-------------|-------------------------------|--------------------|-------------------|----------------------------|---------|
| Plaza     | Liu's [1]   | $358.17 \pm 25.78$            | —                  | $28.91 \pm 0.71$  | $0.353 \pm 0.645$          | 0.12    |
|           | Pütz's [18] | $44.16 \pm 4.43$              | —                  | $30.69 \pm 1.53$  | $0.164 \pm 0.018$          | 0.92    |
|           | Wang's [2]  | $1.19 \times 10^3 \pm 335.25$ | $33.83 \pm 19.19$  | $47.13 \pm 9.59$  | $0.037 \pm 0.010$          | 0.94    |
|           | Ours (PC)   | $13.93 \pm 3.93$              | $52.57 \pm 25.99$  | $28.32 \pm 1.55$  | $0.018 \pm 0.007$          | 1.00    |
| Hills     | Liu's [1]   | $219.31 \pm 9.19$             | —                  | $29.65 \pm 0.57$  | $0.149 \pm 0.015$          | 0.28    |
|           | Pütz's [18] | $62.42 \pm 3.36$              | —                  | $28.79 \pm 0.30$  | $0.130 \pm 0.031$          | 0.80    |
|           | Wang's [2]  | $1.05 \times 10^3 \pm 528.27$ | $33.58 \pm 30.33$  | $40.53 \pm 13.73$ | $0.011 \pm 0.010$          | 0.86    |
|           | Ours (PC)   | $14.23 \pm 4.39$              | $32.89 \pm 22.87$  | $27.79 \pm 0.44$  | $0.011 \pm 0.006$          | 0.92    |

Trajectory evaluation results in simulated *plaza* and *hills* scenarios (mean and standard deviation).  $\bar{T}_s$ : average path searching time,  $\bar{T}_o$ : average trajectory optimization time,  $\bar{L}_t$ : average trajectory length,  $\bar{C}_t$ : average trajectory curvature,  $S_t$ : success rate of finding feasible trajectories in 50 attempts.

3D voxels, which introduces unnecessarily large numbers of node visits. Both Liu's and Pütz's approaches have small  $N_s$ , as they might generate navigation graphs sparser than grid maps by evaluating points or meshes. However, they take a longer time to query the neighboring nodes and finish a single visit compared with searching on grid maps. The total time  $T_{all}$  of our approach which already includes the time for data exchange between computation devices is still small. Our approach generates trajectories in a short time even on a mobile device. The computation time is still rational in large-scale scenarios like *overpass*.

Fig. 5 presents the 3D models (row 1), the point clouds (row 2), and the traversability estimation results of our approach (row 3) together with our generated trajectories. Our approach provides feasible, smooth, and executable trajectories in both indoor and outdoor multi-layer environments with structured or irregular terrains. The trajectories in *factory*, *forest*, and *overpass* are successfully validated on the Pioneer 3-DX robot in physical simulation (please see the attached video for more detail on the simulation results).

Fig. 8 compares the planning results of different approaches in *factory* (a) and *forest* (b). Without the awareness of the robot's motion capabilities, Liu's approach (blue) generates infeasible paths. It fails to consider the body size of the robot in *factory* and the path collides with the pillars in *forest* as it shows a strong preference for following the geodesic direction. Pütz's method (yellow) plans feasible paths without providing the velocity information. Both Wang's (red) and our approach (green) generate smooth trajectories. However, Wang's method may fail to capture detailed terrain structures using voxels and provide sub-optimal solutions. For example, in *factory*, the red trajectory takes an early turn and flies down the first slope from the side (bottom left corner of Fig. 8 (a)).

Table IV evaluates the trajectory generation performance. Without the awareness of robots' motion capabilities, Liu's method [1] frequently moves through untraversable obstacles or narrow passages. Wang's method [2] using rough voxels for scene evaluation fails to capture detailed terrain information and plans with unnecessary detours, resulting in long trajectory length. Our approach provides high-quality smooth trajectories in a short time with lower trajectory length  $\bar{L}_t$  and curvature  $\bar{C}_t$  on average in both scenarios. By analyzing continuous ground elevations, our approach better understands the terrain conditions and achieves higher success rates.

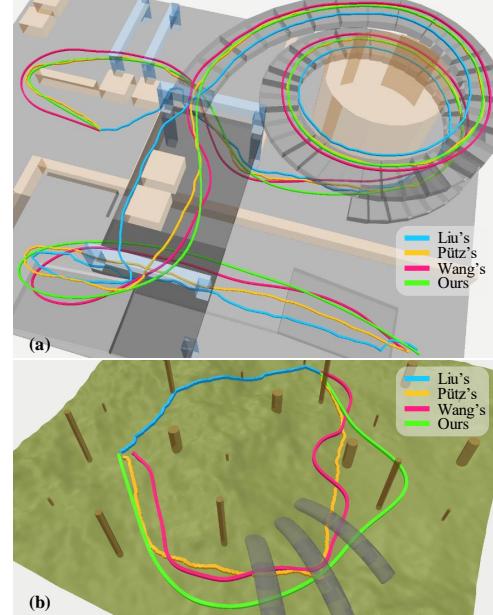


Fig. 8. Planning results of the evaluated approaches in *factory* (a) and *forest* (b). Green: our approach, blue: Liu's method, yellow: Pütz's method, red: Wang's method. Liu's method provides infeasible solutions as it fails to consider the robot's shape and motion capabilities. Pütz's method provides paths without velocity information. Wang's trajectory flies down from the side of the slope in the bottom left corner of (a), resulting in irrational behavior.

#### D. Real-world Performance

The last two rows of Table III show the computation time of the evaluated approaches in real-world experiments. Our approach still generates feasible trajectories on the point cloud maps and outperforms the baseline approaches using other scene representations in efficiency. It performs scene evaluation and initial path planning in only several milliseconds and the total navigation speed is more than 2 orders of magnitude faster than existing approaches.

Fig. 9 shows the photos (row 1), point cloud maps with the generated trajectories (row 2), and the estimated travel costs (row 3) of the real-world scenarios. In *stairs*, our approach successfully navigates the quadrupedal robot to reach the second floor through the winding stairs in this multi-layer structure. In addition, by capturing the terrain features and rationally designing the travel cost functions, our framework navigates the robot with locomotion capabilities awareness. The robot can automatically adjust the body height to walk through the narrow arches in *boxes*. Also, it chooses to move

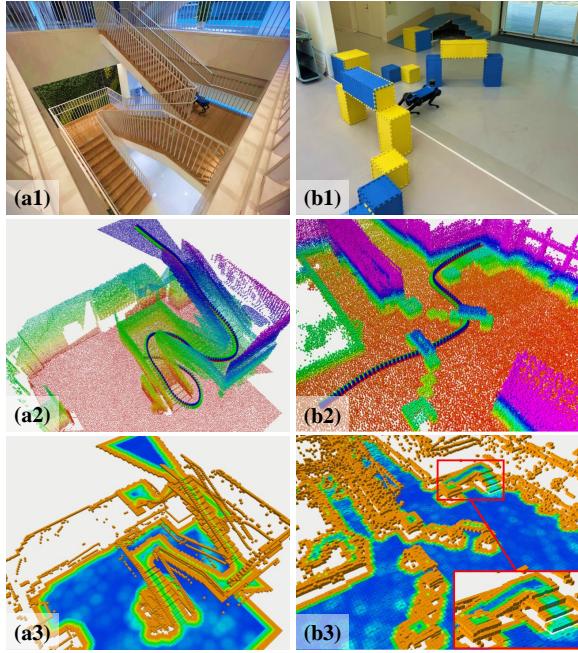


Fig. 9. The real-world experiment scenarios and the navigation results. The first row shows the photos of environments: *stairs* (a1) and *boxes* (b1). Row 2 presents the corresponding point cloud maps generated by our approach. Our quadrupedal robot successfully finished these navigation tasks in real-world experiments following the generated trajectories. In *boxes*, the robot prefers to move up to the platform through the gentle slope as the slope has lower travel costs and is safer than the steep stairs.

up from the slope rather than the stairs beside to reach the target as walking on the steep stairs is more risky and the travel costs on the stairs are higher. More details of the real-world experiments are also presented in our video.

## V. DISCUSSION

In addition to its strong performance in the experiments, the proposed navigation framework is also extensible to be applied in more complex navigation tasks. Our scene representation is compatible with a wide range of traditional or learning-based methods on elevation maps to further enhance the performance of map construction, scene evaluation, and path planning. Semantic information could also be concatenated to the tomogram slices to provide additional environment features beyond the geometric structures for more robust navigation behavior. By understanding the 3D environment as 2.5D layers, our framework achieves high simplicity and computation efficiency, which also presents its strong potential for online navigation and exploration tasks.

There are some limitations of the presented framework. The current approach requires a relatively dense and high-quality point cloud map of the scene. It may generate sub-optimal trajectories or fail to find a solution if the point cloud is too sparse or contains too much noise and dynamic objects. Although existing methods can link faraway nodes across the missing areas for path planning [1], [18] or fill in the blanks with neighboring elevations [8] for dense mapping, the lack of concrete terrain information in unobserved regions may lead to dangers in downstream tasks [22]. Learning-based approaches could be introduced to recover the structure of noisy or

occluded regions better with the awareness of reconstruction uncertainty (as done in [22]) and remove dynamic points to reduce human burden in pre-processing point clouds.

## VI. CONCLUSION

This paper presented a highly efficient and extensible global navigation framework for ground robots in complex multi-layer structures. We introduced a novel scene representation to analyze the point cloud map from a tomographic view. The resulting tomogram slices extend traditional elevation maps to represent multi-layer 3D structures while maintaining their simplicity in mapping and processing. Both terrain conditions and spatial structures are evaluated with the awareness of the robot's locomotion and height adjustment capabilities. The map construction and the scene evaluation stages can be accelerated through parallel computation to reduce the processing time. In addition, our scene representation approach reduces the burden of path search. Our trajectory generation module efficiently provides cost-optimal 3D trajectories and supports active body height adaptation to the narrow environments. The proposed framework is evaluated in both simulated and real-world experiments, demonstrating its highly efficient navigation performance in various complicated 3D environments.

## ACKNOWLEDGMENT

We would like to thank Ren Xin, Peng Yun, and Xiangcheng Hu for their valuable suggestions.

## REFERENCES

- [1] M. Liu, "Robotic online path planning on point cloud," *IEEE Transactions on Cybernetics*, vol. 46, no. 5, pp. 1217–1228, 2016.
- [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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7858–7864, 2023.
- [3] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.
- [4] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.
- [5] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.
- [6] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.
- [7] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2276–2282, 2006.
- [8] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation mapping for locomotion and navigation using gpu," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2273–2280, 2022.
- [9] J. Jiao, F. Chen, H. Wei, J. Wu, and M. Liu, "Lce-calib: Automatic lidar-frame/event camera extrinsic calibration with a globally optimal solution," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2023.
- [10] Y. Wu and J. Zhao, "A robust and precise lidar-inertial-gps odometry and mapping method for large-scale environment," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5027–5036, 2022.
- [11] K. Liu and M. Cao, "Dlc-slam: A robust lidar-slam system with learning-based denoising and loop closure," *IEEE/ASME Transactions on Mechatronics*, pp. 1–9, 2023.
- [12] P. Shi, Z. Zhu, S. Sun, X. Zhao, and M. Tan, "Invariant extended kalman filtering for tightly coupled lidar-inertial odometry and mapping," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2023.

- [13] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.
- [14] G. G. Waibel, T. Löw, M. Nass, D. Howard, T. Bandyopadhyay, and P. V. K. Borges, "How rough is the path? terrain traversability estimation for local and global path planning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16462–16473, 2022.
- [15] F. Ruetz, E. Hernández, M. Pfieffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges, "Ovpc mesh: 3d free-space representation for local ground vehicle navigation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8648–8654, 2019.
- [16] M. Brandão, O. B. Aladag, and I. Havoutis, "Gaitmesh: Controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3596–3603, 2020.
- [17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [18] S. Pütz, T. Wiemann, M. K. Piening, and J. Hertzberg, "Continuous shortest path vector field navigation on 3d triangular meshes for mobile robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2256–2263, 2021.
- [19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.
- [20] D. Hoeller, N. Rudin, C. Choy, A. Anandkumar, and M. Hutter, "Neural scene representation for locomotion on structured terrain," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8667–8674, 2022.
- [21] M. Stölzle, T. Miki, L. Gerdes, M. Azkarate, and M. Hutter, "Reconstructing occluded elevation information in terrain maps with self-supervised learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1697–1704, 2022.
- [22] B. Yang, Q. Zhang, R. Geng, L. Wang, and M. Liu, "Real-time neural dense elevation mapping for urban terrain with uncertainty estimations," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 696–703, 2023.
- [23] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.
- [24] B. Yang, J. Jiao, L. Wang, and M. Liu, "An online interactive approach for crowd navigation of quadrupedal robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13556–13562, 2022.
- [25] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak, "Coupling vision and proprioception for navigation of legged robots," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17273–17283, 2022.
- [26] J. Huang, B. Zhou, Z. Fan, Y. Zhu, Y. Jie, L. Li, and H. Cheng, "Fael: Fast autonomous exploration for large-scale environments with a mobile robot," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1667–1674, 2023.
- [27] J. Guzzi, R. O. Chavez-Garcia, M. Nava, L. M. Gambardella, and A. Giusti, "Path planning with local motion estimations," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2586–2593, 2020.
- [28] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6914–6921, 2021.
- [29] B. Yang, L. Wellhausen, T. Miki, M. Liu, and M. Hutter, "Real-time optimal navigation planning using learned motion costs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9283–9289, 2021.
- [30] X. Cai, M. Everett, L. Sharma, P. R. Osteen, and J. P. How, "Probabilistic traversability model for risk-aware motion planning in off-road environments," *arXiv preprint arXiv:2210.00153*, 2022.
- [31] X. Meng, N. Hatch, A. Lambert, A. Li, N. Wagener, M. Schmittle, J. Lee, W. Yuan, Z. Chen, S. Deng, G. Okopal, D. Fox, B. Boots, and A. Shaban, "Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation," 2023.
- [32] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, "Locomotion policy guided traversability learning using volumetric representations of complex environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5722–5729, 2022.
- [33] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.
- [34] J. Cheng, Y. Chen, Q. Zhang, L. Gan, C. Liu, and M. Liu, "Real-time trajectory planning for autonomous driving with gaussian process and incremental refinement," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8999–9005, IEEE, 2022.
- [35] E. Rohmer, S. P. N. Singh, and M. Freese, "CoppeliaSim (formerly vrep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.
- [36] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and systems*, vol. 2, pp. 1–9, Berkeley, CA, 2014.
- [37] "Jueying Mini." [https://www.deeprobotics.cn/en/products\\_jy\\_301.html](https://www.deeprobotics.cn/en/products_jy_301.html). Accessed: 2021-08-30.



**Bowen Yang** received the B.Eng. degree in the Hong Kong Polytechnic University, HKSAR, China, in 2019. He is currently pursuing the Ph.D. degree in the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, HKSAR, China, supervised by Prof. Ming Liu. His research mainly focuses on scene understanding and path planning for autonomous robots.



**Jie Cheng** received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently pursuing the Ph.D. degree in the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, HKSAR, China, supervised by Prof. Ming Liu. His research mainly focuses on motion planning and motion forecasting for robots and autonomous vehicles.



**Bohuan Xue** received the B.Eng. degree in computer science and technology from College of Mobile Telecommunications, Chongqing University of Posts and Telecom, Chongqing, China, in 2018. He is currently working toward the Ph.D. degree in electrical engineering with the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, HKSAR, China. His research interests include SLAM, computer vision, and 3D reconstruction.



**Jianhao Jiao** received the B.Eng. degree in instrument science from Zhejiang University, Hangzhou, China, in 2017, and the Ph.D. from the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, HK-SAR, China, in 2021, supervised by Prof. Ming Liu. He is now a research associate at the same university. His research interests include state estimation, SLAM, dense mapping, sensor fusion, and computer vision.



**Ming Liu** received the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zurich, Switzerland, in 2013, supervised by Prof. Roland Siegwart. He is currently with the Robotics and Autonomous Systems, The Hong Kong University of Science and Technology (Guangzhou), the director of Intelligent Autonomous Driving Center, as an Associate Professor. Prof. Liu is currently an Associate Editor for *IEEE Robotics and Automation Letters*, *IET Cyber-Systems and Robotics*, *International Journal of Robotics and Automation*. His research interests include dynamic environment modeling, deep-learning for robotics, 3D mapping, machine learning, and visual control.