2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

3D Global Path Planning for Walking Robots on Sparse Volumetric Maps

Marvin Grosse Besselmann[1] Ramona Häuselmann[1] Samuel Mauch[1] Lennart Puck[1]
Tristan Schnell [1] Arne Rönnau[1] Rüdiger Dillmann[1]

Abstract— **The use of mobile robots has become increasingly common in multiple areas of daily life. To increase their autonomy for performing various tasks, efficient navigation skills are essential. The most crucial component of such navigation is the ability to calculate a global path between two points. The global path planning problem for mobile robots is typically limited to two-dimensional environments, in which the environment is projected onto a planar surface. While this approach works well in structured environments like industrial settings, it may not be suitable for all applications of mobile robots. With modern walking robots, capable of navigating complex terrain, more advanced path planning approaches are necessary. This work proposes a path-planning approach that utilizes the entire three-dimensional space, allowing for navigation in even the most challenging terrain. The central idea is to extend a traditional A* path planner to work directly on a fast volumetric map structure to generate optimal paths through the environment. Multiple optimizations and adjustments are introduced to improve the algorithm's performance. By applying morphology operators to sparse maps, sensor inaccuracies during the map construction are mitigated. Additionally, adjustments are made to handle the added complexity introduced by the extra search space dimension and to comply with the limitations of autonomous walking robots. This is paired with an efficient caching strategy to enhance the overall path-planning speed. The capability of the path planning approach is evaluated using both artificial and real-world maps. The results demonstrate that this approach shows great potential for enabling mobile ground robots to autonomously navigate even the most demanding terrains utilizing the entire three-dimensional space.**

I. Introduction

The field of mobile robotics is rapidly evolving and an active area of research. Mobile robots are becoming increasingly prevalent in various industries, including storage facilities and production plants in which they are used to automating logistics and workflows. Simpler systems like vacuum cleaning or lawn mowing robots have also become popular in everyday life. As mobile robot technology advances, they are becoming more capable of supporting and complementing us in performing tasks that may be too dangerous, expensive or burdensome for humans, such as search and rescue or exploratory missions. To navigate autonomously in an uncontrolled environment, mobile robots require the ability to plan and follow a path between their current and a target position. To achieve this, mobile robots typically require an accurate map representation of the real world. This map is either provided in advance or more commonly built up on the go, using current sensor readings of the robot.

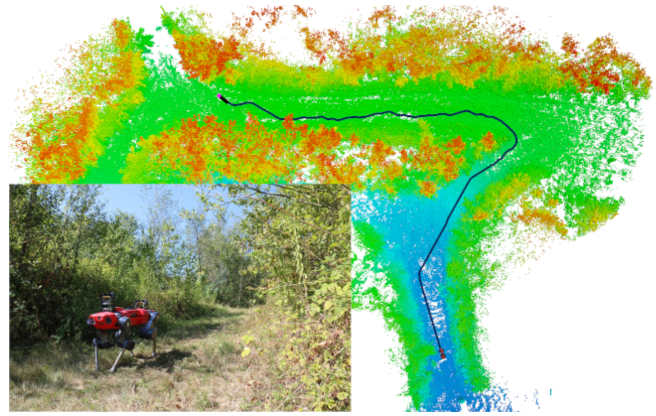

Fig. 1: Map of a search and rescue testing facility of the THW in Aachern. The map was generated by an ANYmal robot using a Velodyne VLP-16 and covers an area of about 62 m×53 m×8 m. The blue path shows the result of the proposed planning algorithm. In total, the path was around 55 m long and traversed through rough overgrown terrain and multiple inclines.

Most current work in the field of mapping and path planning primarily focuses on the 2D or 2.5D space in an industrial setting, while neglecting more complex terrains. One of the most common map representations for this task is the 2D representation of the surroundings with an Occupancy Grid Map [1]. Here the world is split into a discrete 2D space, where each pixel denotes if an area is free or occupied. These approaches can be enhanced by additional information such as the height of the cell, which leads to the work by Herbert et al. [2] where an elevation map is generated. This approach is further extended by Fankhäuser et al. [3] into a multilayer concept. These layers can include additional information like for example surface normals or traversability. The PlexMap framework by Heppner et al. [4] applies a quadtree as the underlying data structure instead of a 2D grid for the same concept. Triebel et al. [5] proposed the Multi-Level Surface maps, where each cell contains multiple height entries. This enables the mapping of multi-level buildings and pathways. A different approach to 2D mapping would be a visibility graph or waypoint graph as proposed by Lozano-Perez and Weslay in [6]. However, all these approaches represent the environment from a 2D perspective and are limited for ground operations. However, even for ground operations the

[1] Department of Interactive Diagnosis and Service Systems (IDS), FZI Research Center for Information Technology, Haid-und-Neu-Straße 10–14, 76131 Karlsruhe, Germany.

The research leading to these results has received funding under Grant Agreements No. 50RA2026 and No. 50RA2404 by the German Space Agency (DLR) on the basis of a decision by the German Bundestag.

2D limitation becomes infeasible in more complex structures like bridges or multi-floor environments. For 3D navigation a volumetric representation is essential. The most well-known 3D mapping framework is the OctoMap by Hornung et al. in [7]. The OctoMap consists of octrees as an underlying data structure, thereby generating a hierarchical tree structure. A different approach called SkiMap was proposed by De Gregorio and Di Stefano in [8], which is a highly parallel framework based on SkipLists. In both Macenski et al. [9] and Grosse Besselmann et al. [10] OpenVDB [11] is used as an efficient data structure for robot mapping.

Most path-planning approaches also only consider 2D environments. In most 2.5D applications the same algorithms can be reutilized by creating a 2D traversability layer from the additional height value. Nevertheless, some planners like the ART Planner [12] plan directly on a 2.5D gridmap. However, especially for UAVs there are multiple examples of path planning in a 3D space. Generally, path planning can be divided into five sub-categories as stated in [13]. The authors divide the path planners into the categories: sampling-based, node-based, mathematic model-based, bio-inspired, and multifusion-based algorithms. Sampling-based algorithms include Probabilistic Roadmaps [14] and Rapidly Exploring Random Trees [15] or its derivates like RRT*[16], RRT-connect[17] or, the combination of both approaches, RRT*-connect[18]. The group of node-based algorithms on the other hand includes approaches like for example A* [19] and D* [20]. Since the proposed work deals with path planning in volumetric spaces, these categories are expanded.

A classical implementation of an A* for 3D Navigation was proposed by Khuswendi et al. in [21]. Colas et al. [22] propose a 3D Navigation algorithm on pointclouds that is capable of traversing challenging terrain. Their work does not utilize a complete environment model but is rather based on a lazy tensor voting to assess traversability. A highly efficient approach was proposed by Oleynikova et al. [23] with VoxBlox, which uses signed distance fields to quickly generate 3D paths for Micro-UAVs. Additionally, there exist learning-based approaches such as a 3D path planning based on a hierarchical Deep Q Network by Sun et al. in [24].

However, most approaches, while being capable of planning in known 3D environments, lack the capability to explore unknown terrains and are based on UAVs. Ground robots navigating in complex 3D environments face additional constraints due to their movement restrictions, which need to be taken into consideration during path planning. Our proposed work focuses on handling complex scenarios for ground robots operating in challenging terrains. To enable these robots to navigate through even the most challenging terrain, the proposed planner utilizes the complete three-dimensional geometric information of the environment. Unlike existing approaches, our planner takes the specific movement constraints of ground robots into consideration. This means that the generated path must not violate the physical restrictions of these robots, such as their inability to fly or pass through obstacles within their bounding box. However, the plan should still allow the robot to traverse smaller obstacles, debris, staircases or slopes. In many cases, mobile robots do not have a complete overview of the entire environment they are operating in. As a result, the navigation system must also be able to handle planning into previously unknown areas. To achieve all this, we present an adapted A* algorithm that operates directly on an efficient volumetric map structure, capable of representing the complete three-dimensional environment. To address the additional restrictions posed by ground vehicles, we have supplemented the algorithm with extensive surface and collision checks, ensuring that the calculated path adheres to the movement constraints of the robot. The structure of this paper is as follows:

- In Section II, we provide a comprehensive overview over the proposed planning algorithm. Furthermore, we elaborate the optimization techniques we used to speed up the planning process.
- Section III contains an evaluation of our proposed method. The results of the algorithm are presented in various simulated and real-world scenarios.
- Section IV highlights the prospects and limitations of path planning directly on a three-dimensional data structure. In addition, a discussion about further improvements to the planning algorithm is given.

II. Approach

Mobile ground robots are becoming increasingly common to assist human operators or explore environments. Recent advancements in dynamic walking robots, such as Boston Dynamics Spot or ANYbotics ANYmal, have enabled them to even traverse challenging terrain. To operate autonomously, the robot has to be able to calculate an optimal collision-free path between its own and a target position. Most available planning algorithms reduce this problem to a two-dimensional search space. Assuming each obstacle can be reduced to a single point, the entire environment is projected onto a planar surface. However, this assumption

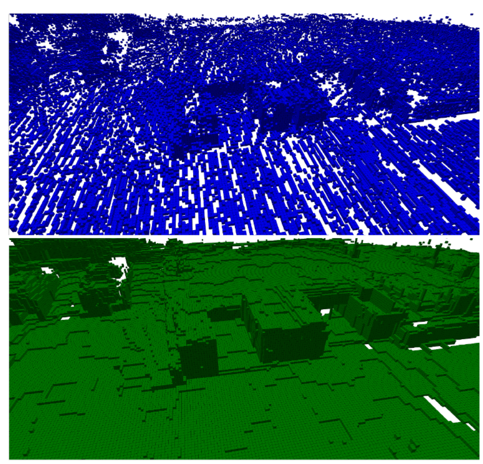

Fig. 2: Sparse Map of a parking lot before(top) and dense transformation after(bottom) the morphological closing operation.

often breaks down in challenging terrain with multiple obstacles, requiring the use of the full three-dimensional space of the environment. Simply applying traditional algorithms to the three-dimensional space becomes infeasible due to the increased dimension of the search space, and the resulting performance degradation. Moreover, when three-dimensional information is available, the size of the robot becomes much more relevant, as we can more accurately determine whether the robot would fit through any given area. Finally, the movement restrictions of autonomous ground vehicles, which require steady ground and can only traverse certain angles of incline, present an additional challenge that requires time-consuming surface checks. To alleviate these challenges, we propose a novel three-dimensional path planning algorithm specialized for autonomous ground vehicles in challenging terrain. Our approach builds upon the core concept of a traditional A* algorithm, but with several adjustments to enable it to work directly on an efficient volumetric map representation. Our approach is based on the VDB-Mapping [10] framework which creates a volumetric voxel map from LIDAR data and stores it in a memory-efficient hierarchical tree structure. The mapping framework uses OpenVDB [11] as the data storage backbone, which provides fast access to the underlying data structure. The library also includes inbuilt grid operators which are used to efficiently adjust the generated map. The morphological operators such as erosion, dilation and the combined closing operator are particularly crucial in our approach. These operators are extensively used during the planning process to improve the map quality and simplify the collision check of the planner.

A. Map Preprocessing

Due to limited sensor coverage and discretization errors during the mapping process, the resulting sparse map may contain smaller voxel gaps. Depending on the mapping process and chosen parameters it might require multiple sensor readings of a cell before it is marked as either occupied or free. As a result, some areas in the map may appear to be full of gaps even though they represent a smooth surface. These gaps can make it difficult for the planning algorithm to accurately determine the present traversablilty state leading to a more complicated planning process. To address this issue, we apply a grid preprocessing step into our planner approach, to transform the map into a dense representation. Specifically, we use morphological closing directly on the OpenVDB data structure [11] to smooth the surfaces and close off smaller gaps. The outcome of this preprocessing step is depicted in Fig. 2. Here, we utilized a somewhat noisy and incomplete map as input, which was subsequently transformed into a more precise and smoothed representation of the environment.

B. Search Graph Creation

Our approach offers the flexibility to accommodate various types of moves on the grid for the A* search algorithm. This means that the search graph is constructed based on connectivity rules in three dimensions, including 6, 18, and 26-connectivity. Opting for lower connectivities has its advantages, as it results in a search graph with lower density and shorter average planning times. However, this comes at the expense of path quality and optimality, which necessitates more post-processing to refine the paths. Choosing an appropriate heuristic for the A* search is crucial, and it must align with the allowed movement rules on the grid. For instance, while the Manhattan distance heuristic is suitable for a 6-connectivity grid, it tends to overestimate diagonal distances in the case of 18-connectivity grids. Consequently, in such scenarios, we employ an adjusted 2D-Octile distance heuristic to ensure more accurate path estimates. Furthermore, in the context of 26-connectivity, our work has extended the 2D-Octile distance heuristic to account for changes in all three dimensions, including spatial diagonals:

$$
\begin{aligned}
d_{\max} &= \max(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|) \\
d_{\min} &= \min(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|) \\
d_{\mathrm{mid}} &= |x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1| - d_{\min} - d_{\max} \\
D_{\text{3D-Octile}} &= d_{\max} + (\sqrt{2} - 1) \cdot d_{\mathrm{mid}} + (\sqrt{3} - \sqrt{2}) \cdot d_{\min} \quad (1)
\end{aligned}
$$

This 3D-Octile heuristic better captures the movements within a 26-connectivity without overestimating the costs, resulting in more precise plans without unnecessarily expanding the search space. In the context of path planning algorithms like A*, it's worth noting that when using heuristics such as Manhattan, Octile, and 3D Octile distance, their values remain unchanged in the immediate vicinity of a node until the goal is approached by at least one dimension. This behavior leads to the expansion of redundant nodes in the priority queue, a situation that not only results in suboptimal paths but also significantly increases the computational workload and, consequently, the required processing time. To address this issue, a technique known as 'tie-breaking' comes into play. In tie-breaking, an additional function is employed to marginally adjust heuristic values, making them distinct in the priority queue. The approach introduced in this research incorporates a fraction of the orthogonal distance between the current node and the vector connecting the start and the goal into the heuristic calculation. This modification prioritizes paths along the line of sight between the start and goal, ultimately leading to more efficient and optimized path planning results. In the context of path planning, we define three nodes in a 3D graph. The start node, $\mathbf{v_s}$. The goal node, $\mathbf{v_g}$, represents the destination to be reached. The current node, $\mathbf{v_i}$, is the node currently being evaluated by the heuristic:

$$\mathbf{v_{gs}} = \mathbf{v_g} - \mathbf{v_s} \quad (2)$$

$$\mathbf{v_{is}} = \mathbf{v_i} - \mathbf{v_s} \quad (3)$$

N corresponds to the orthogonal distance of the currently evaluated node to the direct line between the start and destination node. This value is then used to increase the heuristic value $h(\mathbf{v_i})$ by a fraction of N for the more distant nodes.

$$N = \frac{\|\mathbf{v_{gs}} \times \mathbf{v_{is}}\|}{\|\mathbf{v_{is}}\|} \quad (4)$$

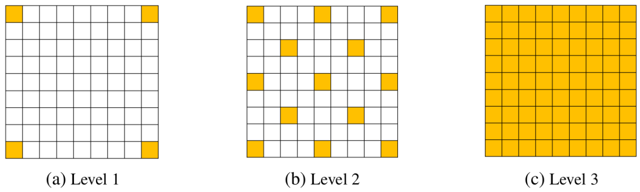

Fig. 3: Different levels of detail for the surface check (top view of the surface box). The test positions are marked in yellow.

$$h'(\mathbf{v_i}) = h(\mathbf{v_i}) + \frac{N}{1000} \tag{5}$$

The last point to consider is how the neighborhood function handles unknown voxels within the map. The occurrence of these voxels is quite common in real-world data and is mostly caused by sensor occlusions or noise. To enable the planner to overcome these areas, all movement restrictions are negated within these areas, to enable the planner to overcome the unknown space in all directions. This, however, leads to a much larger search space. Therefore, the planner tries to prioritize known areas to prevent the algorithm from deteriorating.

C. Collision Check

The A* algorithm normally plans a path at the voxel level and uses the occupancy information of each voxel to determine the traversability of the represented area. However, in many cases, the robot takes up more space than is represented by a single voxel (depending on the map resolution and robot size). Therefore, considering only a single voxel for the traversability of a path can lead to collisions with the environment. To address this problem and guarantee a collision-free path, the algorithm needs to account for the robot size during the planning process. Since checking a volume consisting of several voxels in each step to cover the robot size is very time-consuming, we apply a map inflation preprocessing step, using the efficient morphological dilation operation provided by the OpenVDB [11] framework. During this step each occupied voxel is extended by a certain amount in each dimension, thus obstacles become wider, passages narrower and overhangs/ceilings lower. In the most basic case, the amount should represent the dimensions of the robot's body. However, increasing this parameter would also implicitly increase the additional distance between the wall and the planned path, which can be leveraged for a path safety margin. This allows the planner to treat the robot as a single point at the center of the robot body while implicitly accounting for collisions with the environment through the transformed map.

D. Surface Check

To account for the limitations of ground robots (i.e., steady ground required and limited ability to overcome steps and slopes) we propose an extensive surface check and slope detection algorithm. The surface check is a frequently executed operation, therefore the efficiency is critical to the overall runtime. The process is divided into two phases. The first phase uses a simple model of the surface which allows for faster processing but can only handle flat and slightly uneven surfaces and steps/slopes with limited inclination. The second phase is executed if the first phase is not successful. It performs a more advanced slope detection that can handle slopes of arbitrary inclination. Both phases are performed on the non-inflated grid to prevent the surface area from being artificially enlarged leading to a false ground estimate. The model in phase one uses a horizontal box to represent the area where a surface is required for the robot to stand. The dimensions of the box are determined by the robot's footprint as well as the maximum step height of the robot. The fixed horizontal alignment of the box introduces limitations on the inclination angle of the slopes that can be covered but allows for a comparatively fast check due to the simplicity. The surface check is then performed by checking for occupied cells within the box. Depending on the number of checked positions in the box, varying levels of detail can be used, as shown in Fig. 3. The levels of detail provide a trade-off between computational complexity and accuracy.

Phase two uses a more complex approach that aims to detect the inclination of the surface by fitting a plane. The concept is visualized in Fig. 4. The algorithm searches for anchor points for the plane in a certain height range relative to the robot center point, i.e., the currently examined cell. If at least three anchor points are found and the fitted plane does not exceed the maximum allowed inclination angle, it uses a surface box similar to phase one to check for surface points in the enclosed area. In contrast to the fixed horizontal alignment in phase one, here the box is tilted according to the orientation of the plane. This allows for the detection of slightly uneven surfaces of varying inclinations. If either one of these checks succeeds, the voxel is considered a traversable point in the map.

The surface check and slope detection algorithm can be very time-consuming. They are executed multiple times during a step of the A* algorithm since the result is needed for each voxel that is considered for the path. Furthermore, this check can occur multiple times per voxel since each voxel is the neighbor of multiple other voxels. To alleviate the negative effect this has on the computational time, we

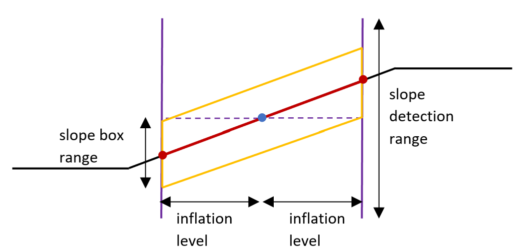

Fig. 4: Concept of the slope detection algorithm. Black: surface, blue: examined cell, red: anchor points for the plane, purple: search range for the anchor points, yellow: surface box

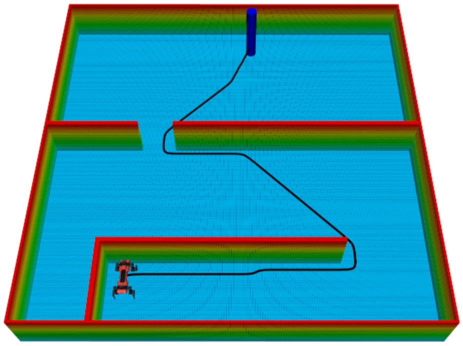

Fig. 5: Simple artificial map to test the baseline of the planner.

introduce a cache that stores information about the presence of sufficient surface for each voxel. This cache value is directly stored efficiently within an additional OpenVDB grid structure which is co-aligned with the coordinates of the actual map. If there is no node present in the ground cache the surface check is performed on demand and the result is subsequently added to the VDB-grid. When encountering the same cell again, the cached value can be used instead of performing the expensive surface check and slope detection calculations. This reduces the complexity of the entire ground check pipeline to a simple $O(1)$ grid lookup.

III. Experiments

In this section, we present the performance evaluation of our proposed three-dimensional global path planner. All experiments were performed on a consumer laptop containing an Intel Core i7-6820HQ with 2.70GHz, with 32 GB RAM, running Ubuntu 20.04. The algorithm was tested on both artificial and real-world scenarios, each presenting different challenges in difficult terrain. Each map was generated with a voxel resolution of 7 cm. To validate our approach we compared it against various existing path planning algorithms. For lower dimensions, this includes a two-dimensional A* taken from the ROS Navigation Stack, as well as the ART-Planner [12] which operates on a 2.5D gridmap [25]. For both the used occupancy grid and the gridmap a downward projection of the generated three-dimensional VDB-map was performed to operate on similar data. To compare our presented approach against other 3D planners, we implemented an efficient map interface that can be used in conjunction with the OMPL library[26]. Here the sampler responsible for finding new valid positions uses the same collision and surface check as described in Section II. This way we were able to compare it to multiple state-of-the-art sampling-based planners. In this work, a PRM, RRT, RRT-connect, and RRT*-connect were used. The non-optimizing planners (i.e., PRM, RRT, and RRT-connect) were all set up to calculate ten solutions to the path planning problem and merge them to gain an optimized plan. The RRT*-connect on the other hand optimizes the found path for a fixed amount of time. Therefore, each of its runtime entries

TABLE I: Planner results for the baseline test

Baseline	Planning Time	Path Length	Node Expansions
2D A*	8.41 ms	20.74 m	-
ART-Planner	276.45 ms	20.52 m	-
3D A*	215.82 ms	20.57 m	133898
RRT	104.32 ms	21.13 m	52358
RRT-connect	97.57 ms	21.07 m	58158
RRT*-connect	5028.55 ms	20.44 m	5071209
PRM	437.28 ms	20.60 m	499779

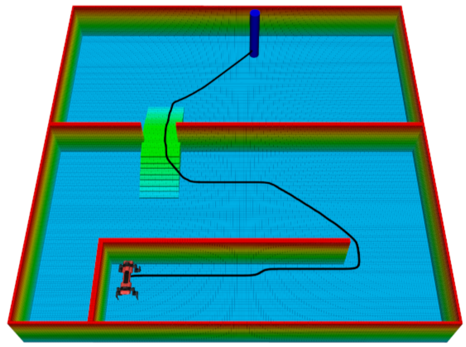

Fig. 6: Artificial map which contains multiple walls and a ramp connecting both rooms.

is close to the defined five-second planning time. For the 3D planning algorithms, the aforementioned preprocessing was necessary to generate viable results. The pre-processing time ranged between 63 and 488 ms, depending on the scale and complexity of the environment. In the following tables presenting the runtime results, the planning time excludes this pre-processing time. The planning time presented in the following tables excludes this pre-processing time. The initial scenarios are performed on artificial maps and are meant as a baseline test of the presented algorithm and the underlying data structure. The first map shown in Fig. 5 represents a small slalom parkour through different rooms without any critical obstacles in the way to verify the basic path planning capabilities. The results of the baseline test are presented in Table I. Obviously, due to the reduced two-dimensional search space the 2D A* performs best of all evaluated planners in the context of runtime. Furthermore, it is, same as the 3D derivative, guaranteed to return an optimal path. It should be noted here, that the small discrepancies between the path length of optimal results visible in each table are due to differences in heuristics and planner concepts (e.g. allowing any-angle planning). In general, in this easy test case, each planner performs well, even though some sampling-based planners generate a longer way due to badly sampled points. The second scenario shown in Fig. 6 is based on the same environment but includes a small ramp in between two rooms. The results are shown in Table II. As the 2D planner projects each obstacle onto its planar footprint the intersection becomes impassible for it. The 2.5D planner on the other hand can still handle non-planar, single-layer obstacles and is able to find a way through the environment.

In the third scenario depicted in Fig. 7 we reuse the map

TABLE II: Planner results for the ramp test case

Ramp	Planning Time	Path Length	Node Expansions
2D A*	-.-	-.-	-
ART-Planner	225.55 ms	23.55 m	-
3D A*	241.23 ms	21.38 m	139404
RRT	159.44 ms	21.71 m	74984
RRT-connect	176.80 ms	21.93 m	86638
RRT*-connect	5024.31 ms	21.07 m	3908630
PRM	732.43 ms	21.29 m	829664

TABLE III: Planner results for the unknown area test case

Unknown Area	Planning Time	Path Length	Node Expansions
2D A*	-.-	-.-	-
ART-Planner	242.27 ms	26.23 m	-
3D A*	1118.01 ms	21.17 m	448201
RRT	-.-	-.-	-
RRT-connect	-.-	-.-	-
RRT*-connect	-.-	-.-	-
PRM	-.-	-.-	-

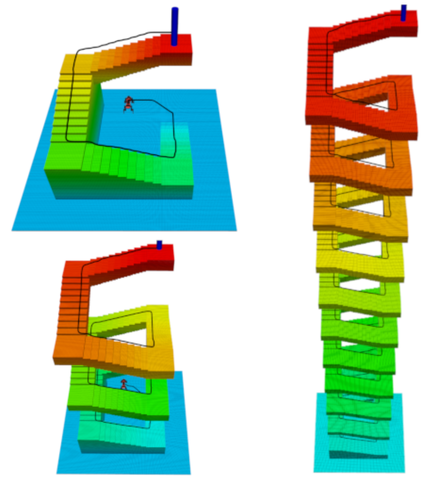

Fig. 8: Three artificial staircases, with one, three, and ten windings

again. However, this time an area of unknown ground is added right after the ramp to showcase the capability to also plan through non-explored areas of a map. The resulting planning time and path length in Table III show, that only the ART-Planner and our presented 3D A* approach were able to find a viable way through the parkour. The sampling-based planners on the other hand were not able to compute a path due to their sampling nature. If a sample would be allowed to persist over an unknown space, the algorithm would also sample points in the unknown space above the map. As a result, the sampling-based planner would generate a path that would just jump over the walls. Therefore, using unknown ground was prohibited to the sampler resulting in no viable path found. Runtime-wise, the 2.5D planner exceeds our presented approach, since in unknown space the number of expanded nodes the A* has to consider increases drastically. However, our approach was able to generate an optimal path with respect to the path length. In the fourth scenario depicted in Fig. 8 the ability of our planner to handle multiple floors is shown. The maps consist of artificial staircases with increasing complexity, including a one-, three- and ten-story staircase. It demonstrates the capability of our implementation to plan a valid path through multi-level areas to the surface above while avoiding obstacles and ledges with sufficient clearance for a safe and collision-free traversal. Since a multi-level staircase can no longer be projected onto a 2.5D elevation map. Therefore, the ART-Planner was, same as the 2D Planner, no longer applicable. The sampling-based planner on the other hand performed well. However, it is evident from the results of Table IV that with increasing size the performance of the sampling-based approach deteriorates. This is mostly due to the increased size of the sampling space from which it picks randomly new points. This combined with the relatively small amount of valid surfaces on top of the stairs might lead to a multitude of invalid samples. After validating, that the planner performs well in artificially generated environments, the next part of the evaluation aims at evaluating its performance on real-world data. For this, we considered two highly different environments that showcase the ability of our planner to handle challenging outdoor terrain. The first of the two real-world scenarios was recorded on a search and rescue testing facility of the THW in AAchern. The map was generated by an ANYmal-C robot using a LIDAR and multiple RGB-D sensors. As can be seen in Fig. 1 the generated path had to overcome multiple inclines and travel through overgrown environments. Even though the area was generally passable without much care for the third planning dimension, the ART-Planner was not able to find a viable path through the environment. This was mostly due to overhanging pushes and trees which were projected down and in turn prevented the planner from finding a viable path. All other three-dimensional planners on the other hand were able to calculate a path through the environment. In this case the A* performed way better since the way contained multiple small passageways. These passages, similar to small doorways, are often a challenge for sampling-based approaches since they require enough samples to fall within these small spaces. As a result, the

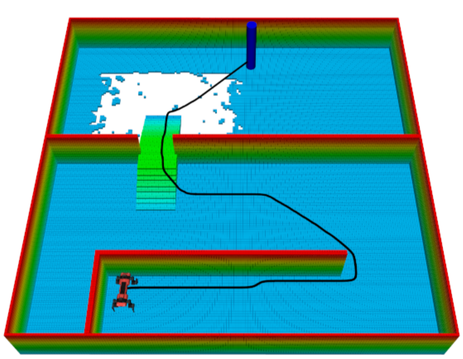

Fig. 7: Similar map as used during the first test cases. This time, though, an area of unknown space was included after the ramp pathway.

TABLE IV: Planner results for the multiple staircase test

1 Floor	Planning Time	Path Length	Node Expansions
2D A*	-.-	-.-	-
ART-Planner	-.-	-.-	-
3D A*	439.92 ms	21.53 m	228497
RRT	659.22 ms	21.37 m	187682
RRT-connect	460.34 ms	21.45 m	137771
RRT*-connect	5028.86 ms	20.65 m	2554886
PRM	937.28 ms	21.11 m	960376
3 Floor	**Planning Time**	**Path Length**	**Node Expansions**
2D A*	-.-	-.-	-
ART-Planner	-.-	-.-	-
3D A*	891.37 ms	58.34 m	390042
RRT	3892.80 ms	61.98 m	556580
RRT-connect	2282.39 ms	62.19 m	438881
RRT*-connect	5033.70 ms	56.66 m	1879636
PRM	2393.43 ms	57.95 m	1939449
10 Floor	**Planning Time**	**Path Length**	**Node Expansions**
2D A*	-.-	-.-	-
ART-Planner	-.-	-.-	-
3D A*	2616.05 ms	207.07 m	943532
RRT	44405.41 ms	231.54 m	3753338
RRT-connect	30804.19 ms	231.86 m	3506627
RRT*-connect	5029.34 ms	205.81 m	1300827
PRM	11304.90 ms	209.29 m	7087769

TABLE V: Planner results for the first outdoor test

Aachern	Planning Time	Path Length	Node Expansions
2D A*	-.-	-.-	-
ART-Planner	-.-	-.-	-
3D A*	1388.07 ms	55.05 m	558430
RRT	6497.10 ms	56.66 m	1003178
RRT-connect	4294.15 ms	57.92 m	636973
RRT*-connect	5022.80 ms	55.26 m	1904943
PRM	5314.67 ms	56.06 m	2484131

sampler has to perform way more ground checks compared to the A*. The map of the last scenario was taken during an analog planetary exploration mission. It was recorded on a Spot robot in the Tabernas Desert in southern Spain using a LIDAR and multiple RGB-D cameras as well. This environment provided multiple craters, mountains, dried river beds and generally rough, difficult-to-pass, rocky terrain. Here again, all three-dimensional planners performed well. In this case, runtime-wise, the sampling-based planners had the advantage since they were able to sample over a huge free-standing terrain. However, from the path length, it is evident, that sampling-based planners, given the chance, will produce non-optimal paths. Reasoning from this it is a trade-off between motion and planning time which planner should be preferred. Overall the results show that the planning time increases with the complexity of the scenario and the planning distance. It is evident, that the planning time directly corresponds to the amount of expanded nodes, since here the costly ground checks are performed. Due to the high cache hit rate of over $> 90\%$ as seen in Table VII a huge amount of redundant ground checks could be prevented leading to a faster planning time.

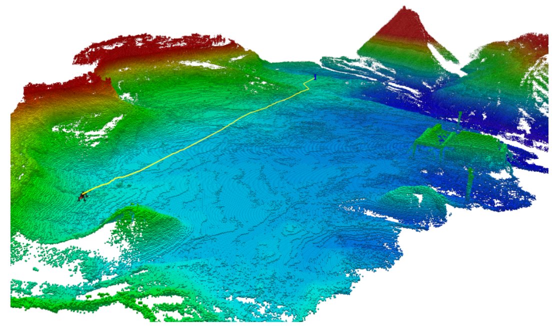

Fig. 9: Map of an area of the Tabernas Desert in southern Spain.

TABLE VI: Planner results for the second outdoor test

Tabernas	Planning Time	Path Length	Node Expansions
2D A*	-.-	-.-	-
ART-Planner	-.-	-.-	-
3D A*	2275.65 ms	60.86 m	430461
RRT	1300.59 ms	62.42 m	293689
RRT-connect	1138.18 ms	62.57 m	212862
RRT*-connect	5035.11 ms	61.29 m	2780266
PRM	1359.46 ms	62.29 m	778263

IV. Conclusion and Future Works

In this work we presented a novel global planning approach based on an adjusted A*, to generate paths utilizing the full three-dimensional space of the environment. This enables the robot to traverse even the most challenging terrain, thus leading to a larger autonomy in various environments. We extended the algorithm with several additional adjustments to address the movement restrictions of autonomous walking robots One limitation of the presented approach is that, due to the increased search space, the calculation of the global path in a 3D environment takes more time than comparable 2D or 2.5D planners. Despite this, our approach enables robots to generate paths that are not possible with traditional 2D planners. To mitigate the increased computational cost, the presented algorithm operates directly on a fast OpenVDB data structure, which provides constant time random access. Furthermore, we presented preprocessing steps and an efficient caching strategy to further improve the computation time during path planning. Key insights of this study are:

- **Efficient 3D Path Planning:** Leveraging the additional third dimension during the planning process enabled

TABLE VII: Ground cache results for all test case

Unknown Area	Ground Checks	Cache Hits	Cache Hit Rate
Baseline	335968	311216	92.63%
Ramp	351334	325265	92.58%
Unknown Area	966521	916115	94.78%
Stair 1 Floor	595693	548625	92.09%
Stair 3 Floor	1040283	954833	91.79%
Stair 10 Floor	2577979	2356077	91.39%
Aachern	1338101	1242874	92.88%
Tabernas	953164	877415	92.05%

the robot to calculate previously infeasible paths. This enables the robot to traverse even the most challenging terrain thus leading to a larger autonomy in various environments. By utilizing the efficient VDB-Map combined with various optimizations we were able to achieve reasonable planning times in large-scale outdoor environments.
- **Consideration of Ground Robot Restrictions:** By applying additional surface checks during the planning process, we were able to address the motion restrictions of ground robots in three-dimensional environments. It was also shown that the developed concepts can be applied to alternative sampling-based planning approaches.
- **Ground Caching:** Applying an efficient caching scheme of surface check results within a VDB-grid improves the performance considerably. This enables the algorithm to reduce the complex re-calculation of ground checks to a simple $O(1)$ cache lookup.

The presented approach shows great potential for autonomous navigation of mobile robots in challenging terrain. However, future work should focus on further decreasing the runtime of the global path planner. Recently the utilized mapping framework was extended by a submapping approach [27] to better handle loop closures during the mapping process. A future improvement for the planner would be to adapt it to this new structure. As short paths can be calculated relatively quickly, it might be viable to improve the runtime by planning only on the smaller submaps instead of a single monolithic map. Additionally, the planning time could be reduced by improving the handling of unknown space during the planning process.

REFERENCES

[1] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985.

[2] M. Herbert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proceedings, 1989 International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1989, pp. 997–1002.

[3] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "ROBOT-CENTRIC ELEVATION MAPPING WITH UNCERTAINTY ESTIMATES," in *Mobile Service Robotics*. WORLD SCIENTIFIC, Aug. 2014, pp. 433–440.

[4] G. Heppner, A. Roennau, J. Oberländer, and S. Klemm, "Lauropesix legged walking robot for planetary exploration participating in the spacebot cup." in *WS on Advanced Space Technologies for Robotics and Automation 2.13*, 2015, pp. 69–76.

[5] 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*. IEEE, 2006, pp. 2276–2282.

[6] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[7] 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, Apr. 2013.

[8] D. De Gregorio and L. Di Stefano, "Skimap: An efficient mapping framework for robot navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2569–2576.

[9] S. Macenski, D. Tsai, and M. Feinberg, "Spatio-temporal voxel layer: A view on robot perception for the dynamic world," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, p. 1729881420910530, 2020.

[10] M. G. Besselmann, L. Puck, L. Steffen, A. Roennau, and R. Dillmann, "VDB-Mapping: A High Resolution and Real-Time Capable 3D Mapping Framework for Versatile Mobile Robots," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. Lyon, France: IEEE, Aug. 2021, pp. 448–454. [Online]. Available: https://ieeexplore.ieee.org/document/9551430/

[11] K. Museth, "VDB: High-resolution sparse volumes with dynamic topology," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–22, June 2013.

[12] 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 2021)*, 2021.

[13] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, Y. Xia, *et al.*, "Survey of robot 3d path planning algorithms," *Journal of Control Science and Engineering*, vol. 2016, 2016.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[15] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan," *Algorithmic and computational robotics*, pp. 303–307, 2001.

[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[17] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[18] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "Rrt*-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[19] 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.

[20] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE international conference on robotics and automation*. IEEE, 1994, pp. 3310–3317.

[21] T. Khuswendi, H. Hindersah, and W. Adiprawita, "Uav path planning using potential field and modified receding horizon a* 3d algorithm," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*. IEEE, 2011, pp. 1–6.

[22] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3d path planning and execution for search and rescue ground robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 722–727.

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

[24] Y. Sun, X. Ran, G. Zhang, H. Xu, and X. Wang, "Auv 3d path planning based on the improved hierarchical deep q network," *Journal of marine science and engineering*, vol. 8, no. 2, p. 145, 2020.

[25] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525

[26] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[27] M. G. Besselmann, A. Roennau, and R. Dillmann, "Vdb-submapping: Efficient handling of retroactive pose changes during large-scale volumetric mapping," in *2023 20th International Conference on Ubiquitous Robots (UR)*. IEEE, 2023, pp. 327–332.