2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Towards Efficient Trajectory Generation for Ground Robots beyond 2D Environment

Jingping Wang[†1,2], Long Xu[†1,2], Haoran Fu[3], Zehui Meng[4], Chao Xu[1,2], Yanjun Cao[1,2], Ximin Lyu[3] and Fei Gao[1,2]

***Abstract*— With the development of robotics, ground robots are no longer limited to planar motion. Passive height variation due to complex terrain and active height control provided by special structures on robots require a more general navigation planning framework beyond 2D. Existing methods rarely considers both simultaneously, limiting the capabilities and applications of ground robots. In this paper, we proposed an optimization-based planning framework for ground robots considering both active and passive height changes on the z-axis. The proposed planner first constructs a penalty field for chassis motion constraints defined in $\mathbb{R}^3$ such that the optimal solution space of the trajectory is continuous, resulting in a high-quality smooth chassis trajectory. Also, by constructing custom constraints in the z-axis direction, it is possible to plan trajectories for different types of ground robots which have z-axis degree of freedom. We performed simulations and real-world experiments to verify the efficiency and trajectory quality of our algorithm.**

I. Introduction

Navigation is an increasingly important task with applications in autonomous systems such as robotics and autonomous driving. Among different types of robots, ground robots are seemed to doomed to be constrained on the ground, where the terrain shape should be taken into account. Furthermore, with the advancement in mechanism, sensing, and designs, the application scenarios for ground robots continue to be enriched. Many ground robots are now capable to work beyond a flat plane. Specifically, there are two situations shown in Fig.1 where it is necessary to consider the change in the height of the robot.

1) The ground is not flat, it may be rugged terrain or facilities with multi-layered structures. In this case, the ground robot's chassis height will passively undulate with the terrain (Fig.1a).
2) The robot has degrees of freedom perpendicular to the horizontal direction, such as wheeled bipedal robots [1], [2] and mobile manipulator robots [3], [4]. In this case, the ground robot can actively change the height of certain parts (Fig.1b).

This work is supported by the Technology Cooperation Project (TC20220507025) from the Application Innovation Laboratory, Huawei Technologies Co., Itd.

†Indicates equal contribution.

[1]State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China. (*Corresponding author: Fei Gao*)

[2]Huzhou Institute of Zhejiang University, Huzhou 313000, China.

[3]School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518107, China.

[4]Applicaion Innovation Laboratory, Huawei Technologies Co., Ltd.

E-mail: {gaolon, cxu, and fgaoaa}@zju.edu.cn

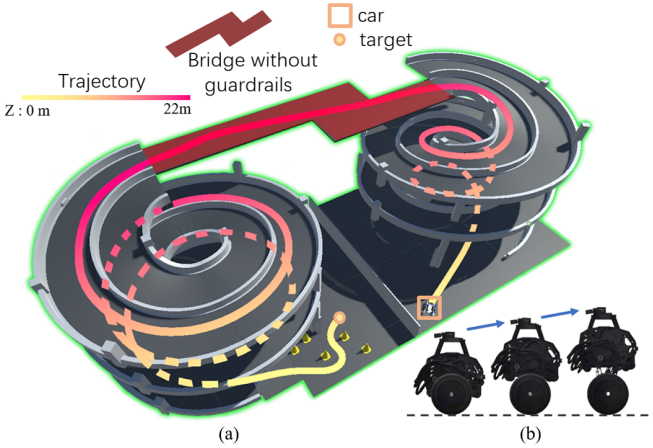


Fig. 1: Two cases where the z-axis height of the ground robot changes. In Fig.(a), the robot must pass through a complex overpass to reach the target, and the height of the chassis is passively varied by terrain constraints. Fig.(b) shows a wheeled bipedal robot that can change the height of its head to accomplish more complex tasks.

However, dealing with the above situations in autonomous navigation is non-trivial. Traditional works [5]–[16] relieve this by assuming that the robot work on a flat surface, which largely simplifies the robot's work scenarios and restricts the motion capabilities. Some works [17]–[19] consider complex terrain but ignore the active height change of ground robots. The inability to well handle the passive and active height variation will greatly limit the capabilities of ground robots, further limiting the applications.

To compensate for the shortcomings of existing methods, in this paper, we design an optimization-based planning framework for ground robots to generate trajectories considering both active and passive height changes.

The proposed algorithm is composed of a pre-analysis procedure and a gradient-based Spatio-temporal trajectory optimizer. Firstly, we filter the original point cloud to provide a search space for the front end of the trajectory generation considering the robot's size. We then construct a penalty field defined in $\mathbb{R}^3$ which embodies the characteristics of the terrain shape. Finally, we optimize the trajectory with smoothness, collision, dynamical feasibility, terrain traversability, and other customized cost terms. Our algorithm can cope with non-planar environments and supports active z-axis degree of freedom. We perform comprehensive tests in simulation and the real world to validate our method.

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10160330

Contributions of this paper are:

1) We propose a novel field construction algorithm in $\mathbb{R}^3$, which describes the level of danger for a robot trying to get a foothold in space.
2) We propose an optimization-based planning framework for ground robots, which can generate trajectories beyond 2D space efficiently.
3) We open source our software[1] for the reference of the community.

II. Related Works

A. Navigation in 2D Cartesian space

Navigation in 2D Cartesian space for ground robots can be translated into elegant mathematical problems, spawning numerous practical studies. Methods can be divided into sampling-based methods and optimization-based methods. Sampling-based methods [5]–[8] are widely used for robot motion planning. The probabilistic planners [9], [10] represented by Rapidly-exploring Random Tree (RRT) [11] obtain a feasible path by expanding the state tree rooted at the starting node. Some improved approaches [12], [13] were later proposed. However, sampling-based methods confront a dilemma between computation consumption and trajectory quality which limits the direct application in realistic settings. Optimization-based methods [14]–[16] usually construct the planning problem as a mathematical optimization problem and solved using numerical optimization algorithms. Commonly, the safety constraints are made by safe motion corridor or Euclidean Signed Distance Functions (ESDF) [20].

B. Navigation considering the terrain shape

Trajectories computed with the assumption of flat terrain may turn out to be highly suboptimal or even unfeasible when mapped onto the 3D terrain surface. To incorporate the 3D shape of the terrain in motion planning of ground robots, several works have been proposed.

Krüsi et al. presented a practical approach [17] to global motion planning and terrain assessment for ground robots in generic 3D environments, which assesses terrain geometry and traversability on demand during motion planning. It consists of sampling-based initial trajectory generation, followed by precise local optimization according to a custom cost measure to directly generate a feasible path connecting two states in the state space of a ground robot's chassis. However, simultaneous terrain assessment during path searching seriously reduces computational efficiency, and the initial RRT-Connect [21] trajectory sampling results in long path search times when dealing with large or complex scenes. Additionally, the final trajectory is biased to longer due to no prior global evaluation.

To solve the above problems, some approaches perform pre-evaluation of global point clouds based on tensor voting [22], improving planning efficiency and trajectory quality. [19] described a solution to robot navigation on 3D surfaces, which utilized an improved A* algorithm for path searching, significantly increasing efficiency. But this approach is only applicable to special robots with adsorption ability. [18] uses sparse tensor voting and accelerates the process with GPU. After building a connectivity graph between global point clouds by the concept of k-Nearest-Neighbor (k-NN) and constructing a Riemannian metric, it searches for trajectories towards the flattest direction. Nevertheless, due to the lack of enlightenment, the Dijkstra algorithm it uses is inefficient.

[1] https://github.com/ZJU-FAST-Lab/3D2M-planner

III. Point Cloud Pre-processing

Inflating point clouds is a typical approach in motion planning to ensure safety. However, this approach is not suitable for our work, because some point clouds in the environment represent traverasable areas like the ground, and cannot be simply considered as obstacles. In fact, in the process of optimization-based trajectory generation, the final trajectory must be safer than the initial trajectory after the optimization process due to the safety constraints. Therefore, the key is to ensure that the initial trajectory is in the robot's C-space. In this work, the initial trajectory is provided by a front-end path searcher.

To this end, we proposed a more concise approach to ensure the safety of the initial trajectory. This approach directly processes the original point cloud and is named Valid Ground Filter (VGF). The raw point cloud records the spatial location information of the object surface, but not all parts of the object surface are available for the robot to stand on. We defined criteria for the "valid" points in the point cloud as follows:

1) The point is on the upper surface of an object.
2) The local area near the point is flat enough which means the robot does not collide or topple.

Denote the original point cloud by M and the point cloud after VGF by M^s. M^s is defined as follow:

$$M^s = \{p \in M | D = K(p,r), \forall p^d \in D, p_z^d \leq p_z \ or \ p_z^d > p_z, \arctan\left(\frac{p_z - p_z^d}{\sqrt{(p_x - p_x^d)^2 + (p_y - p_y^d)^2}}\right) < \theta)\}, \quad (1)$$

where D is the neighboring points of p within radius r. In practice, the value of r should be greater than the minimum ball radius that can wrap around the robot, and θ is the maximum tilt angle of the surface on which the robot can stand. The pipeline for iteratively processing each point is stated in Algorithm 1.

Fig.2 illustrates the process of VGF and the result in a demo scenario. After VGF, the points in M^s all represent the valid surface on which the robot can stand. The path searching process will be performed on M^s to obtain a safe path considering the size of the robot.

IV. Safety Penalty Field

For aerial robots, safety constraints can easily be added to trajectory optimization using ESDF [23] or safe flight

Algorithm 1: Valid Ground Filter

Input: M, r, θ
Output: M^s
begin
  $M^s \leftarrow \emptyset$;
  for ***each*** $p \in M$ **do**
    $D_p = GET\_NEIGHBORS(p, r)$;
    for ***each*** $p^d \in D_p$ **do**
      if $p_z^d < p_z$ **then**
        $\alpha = \arctan\left(\frac{p_z - p_z^d}{\sqrt{(p_x - p_x^d)^2 + (p_y - p_y^d)^2}}\right)$;
        if $\alpha < \theta$ **then**
          $M^s \leftarrow M^s \cup p$;
  return M^s;

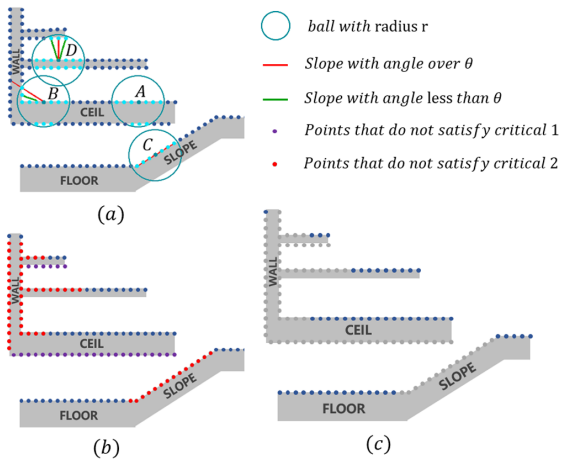


Fig. 2: The Valid Ground Filter Processing. To facilitate this we show side views. Figure (a) illustrates the specific process for four points: According to the criteria, point A is feasible. Point B is too close to the lateral obstacle. Point D is too close to the obstacle above. And the plane where point C is located is too inclined. Figure (b) shows the determination of the points after completing a traversal. Figure (c) shows the final results of VGF.

corridors [24] so that trajectories do not collide with obstacles. However, since the ground robot is constrained on the ground, it is not enough for safety constraints to contain only collision-free term (for example, a trajectory that makes a vehicle run off a cliff meets the collision-free constraint but is extremely dangerous).

In this paper, the safety constraint ensures that no hazards occur while the ground robot is walking along the trajectory, and we define hazards as follows:

1) Collision between robot and object.
2) The robot standing in a too tilted position.
3) The robot undergoes large positional changes (e.g. falling from the edge of a step and drastic change in pose).

To avoid situation 1), we also use the ESDF map defined in the grid map which pushes a 3-dimensional trajectory away from objects in any direction. Moreover, the penalty function based on ESDF is defined in $\mathbb{R}^3$, which is convenient for the optimization problem.

To avoid situations 2) and 3), the estimation of robot posture is necessary. We evaluated the posture of the robot in various places by fitting the local plane using RANSAC [25]. Specifically, estimates were made in the position of each 3D grid, considering that the penalty function should be defined in $\mathbb{R}^3$. However, point clouds exist only close to the object. So we use a *Downward Projection Strategy*:

$p = [x, y, z]^T$ is a point in space, and $H : \mathbb{R}^3 \mapsto \mathbb{R}^3 \times SO(2)$ represents the result of the robot pose estimation at position p. We define $p^{'} = [x, y, z^{'}]^T$ the point that is projected downward onto the surface of the nearest object from p.

$$H(p) = H(p^{'}) = [p_c, \phi]^T \tag{2}$$

The value of H is the RANSAC fit to the local point cloud at $p^{'}$. Where $p_c = [x_c, y_c, z_c]^T$ is the COM(center of mass) of the local plane, ϕ is the angle between the local plane normal vector and $[0, 0, 1]^T$. To avoid the errors caused by the wrong direction of normal vector, $\phi = min(\phi, \pi - \phi)$.

Obviously, for a given position p , a larger ϕ is more detrimental to the robot. And 2-norm of the gradient $||\nabla H(p)||_2$ reflects the sharpness of the robot's positional change when moving near p. Therefore, we defined a travel penalty function $S : \mathbb{R}^3 \mapsto \mathbb{R}$:

$$S(p) = \lambda_f H^2_{(4)}(p) + \lambda_m(\sqrt{||\nabla H(p)||_2}), \tag{3}$$

where λ_f and λ_m are weights.

Thanks to the downward projection strategy, the travel penalty function S is defined in the entire $\mathbb{R}^3$ but there are only horizontal gradients existing on a continuous terrain. In addition, to ensure a certain safety margin, we performed equidistant diffusion of function S, denoted by S'. Since the map is represented by a grid-map, the diffusion process also proceeds discretely. Assume that t is the diffusion distance, construct the matrix $A \in \mathbb{R}^{m \times m}$ which meets the following conditions:

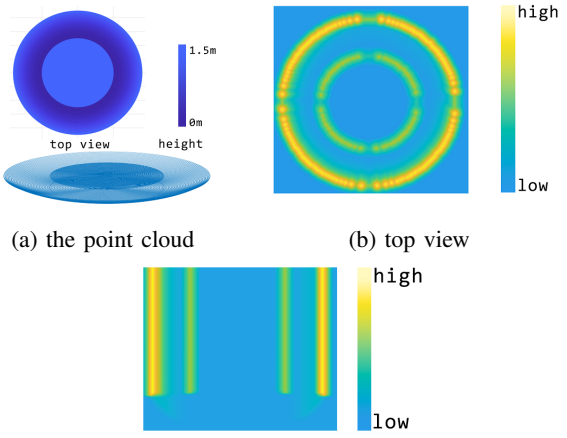
1) $m = 2\lceil t/r_g \rceil + 1$, where r_g is the resolution of the gridmap.
2) $A_{i,j} = \max\{0, 1 - r_g \frac{\sqrt{|i - \frac{m-1}{2}|^2 + |j - \frac{m-1}{2}|^2}}{t}\}$.

Then, we can get the S' by performing a convolution operation on the values of the function S at each horizontal layer with A as the convolution kernel.

Fig.3 shows the result of function S after distribution in a demonstrative environment.

V. Trajectory Generation

In this section, we present our method to incorporate safety and dynamical feasibility constraints into our optimization process. In addition, custom tasks for different types of ground robots can be achieved by constructing custom constraint terms. To begin with, we introduce a computationally



(a) the point cloud (b) top view

(c) side view

Fig. 3: Distribution of function S in a demonstrative environment, which describes some of the properties of the safety penalty field proposed in Sec.IV. Fig.(a) shows the point cloud map of the example environment which includes multi-level structures and sloping floors. Fig.(b) and Fig.(c) show the distribution of the diffused function S in $\mathbb{R}^3$ space from different perspectives. Color represents the value of S.

efficient feasibility check method that applies to a wide range of constraints. Then this method is used in a constrained trajectory optimization process.

A. Path Searching

We use a front-end path searcher to provide the initial trajectory. Since smoothness and dynamic feasibility are considered in the back-end optimization, the path searching becomes relatively simple and only the variation of the robot's position needs to be considered.

In this work, we use A* [26] algorithm to search the path on the grid-map. Thanks to the VGF process, unsafe areas after considering the robot size have been excluded from the search space. In addition, the fact that the robot motion is constrained on the ground surface reduces the number of cells to be searched, and only the cells extremely close to the upper surface need to be considered.

B. Trajectory Representation

In this work, we adopt $\mathfrak{T}_{\mathbf{MINCO}}$ [27] for our trajectory representation, which is a minimum control effort polynomial trajectory class defined as

$$\begin{aligned}\mathfrak{T}_{\mathbf{MINCO}} = \{p(t) : [0, T_\Sigma] \to \mathbb{R}^m | \mathbf{c} = \mathcal{M}(\mathbf{q}, \mathbf{T}),\\ \mathbf{q} \in \mathbb{R}^{m(M-1)}, \mathbf{T} \in \mathbb{R}^M_{>0}\},\end{aligned} \tag{4}$$

where $p(t)$ is an m-dimensional M-piece polynomial trajectory with degree $N = 2s - 1$, s is the order of the relevant integrator chain, $\mathbf{c} = (\mathbf{c}_1^T, ..., \mathbf{c}_M^T)^T \in \mathbb{R}^{2Ms \times m}$is the polynomial coefficient, $\mathbf{q} = (\mathbf{q}_1, ..., \mathbf{q}_{M-1})$ is the intermediate waypoints and $\mathbf{T} = (T_1, T_2, ..., T_M)^T$ is the time allocated for all pieces, $\mathcal{M}(\mathbf{q}, \mathbf{T})$ is the parameter mapping constructed from Theorem 2 in [27], and $T_\Sigma = \sum_{i=1}^M T_i$ is the total time.

An m-dimensional M-piece trajectory is defined as

$$p(t) = p_i(t - t_{i-1}) \quad \forall t \in [t_{i-1}, t_i), \tag{5}$$

and the i^{th} piece trajectory is represented by a $N = 5$ degree polynomial:

$$p_i(t) = \mathbf{c}_i^T \beta(t) \quad \forall t \in [0, T_i), \tag{6}$$

where $\mathbf{c}_i \in \mathbb{R}^{(N+1)\times m}$ is the coefficient matrix, $\beta(t) = [1, t, ..., t^N]^T$ is the natural basis, and $T_i = t_i - t_{i-1}$ is the time allocation for the i^{th} piece. $\mathfrak{T}_{\mathbf{MINCO}}$ is uniquely determined by $(\mathbf{q}, \mathbf{T})$. And the parameter mapping $\mathbf{c} = \mathcal{M}(\mathbf{q}, \mathbf{T})$ can convert trajectory representations $(\mathbf{c}, \mathbf{T})$ to $(\mathbf{q}, \mathbf{T})$, which allows any second-order continuous cost function $J(\mathbf{c}, \mathbf{T})$ to be represented by $H(\mathbf{q}, \mathbf{T}) = J(\mathcal{M}(\mathbf{q}, \mathbf{T}), \mathbf{T})$. Hence, we can get $\partial H/\partial \mathbf{q}$ and $\partial H/\partial \mathbf{T}$ from $\partial J/\partial \mathbf{c}$ and $\partial J/\partial \mathbf{T}$ easily.

To cope with the constraints $\mathcal{G}(p(t), ..., p^{(3)}(t)) \preceq \mathbf{0}$, such as safety assurance and dynamical feasibility, we discretize each piece of the trajectory as κ_i *Constraint Points* [27] $\tilde{p}_{i,j} = p_i((j/\kappa_i) \cdot T_i), j = 0, 1, ..., \kappa_i - 1$. Then the discrete constraints can be transformed into weighted penalty terms, which can form the cost function $J(\mathbf{c}, \mathbf{T})$ by addition.

C. Optimization Problem Formulation

We formulate the beyond 2D trajectory generation for ground robots as an unconstrained optimization problem:

$$\min_{\mathbf{c},\mathbf{T}} J(\mathbf{c}, \mathbf{T}) = \lambda_s J_s + \lambda_t J_t + \lambda_m J_m + \lambda_d J_d, \tag{7}$$

where $\lambda_k, k = \{s, t, m, d\}$ are the weights of each cost function.

1) Safety Assurance J_s: We use the method proposed in [23] to maintain an ESDF from point clouds. Combining it with the safety penalty field we designed in Sec.IV, we can keep the robot moving in the traversable area, the cost function and gradients are:

$$\mathcal{G}_s(\tilde{p}_{i,j}) = \begin{cases} 0, & S'(\tilde{p}_{i,j}) \le s_{thr}, \\ S'(\tilde{p}_{i,j}) - s_{thr}, & S'(\tilde{p}_{i,j}) > s_{thr}, \end{cases} \tag{8}$$

$$\mathcal{G}_c(\tilde{p}_{i,j}) = \begin{cases} d_{thr} - d(\tilde{p}_{i,j}), & d(\tilde{p}_{i,j}) < d_{thr}, \\ 0, & d(\tilde{p}_{i,j}) \ge d_{thr}, \end{cases} \tag{9}$$

$$J_s = \sum_{i=1}^{M} \sum_{j=0}^{\kappa_i - 1} \mathcal{C}(\mathcal{G}_n(\tilde{p}_{i,j})) \cdot \frac{T_i}{\kappa_i}, \tag{10}$$

$$\frac{\partial J_s}{\partial \mathbf{c}_i} = 3 \sum_{j=0}^{\kappa_i - 1} \mathcal{Q}(\mathcal{G}_n(\tilde{p}_{i,j})) \cdot \beta(\frac{j}{\kappa_i} T_i) \cdot (\nabla S_d^T) \cdot \frac{T_i}{\kappa_i}, \tag{11}$$

$$\frac{\partial J_s}{\partial T_i} = \sum_{j=0}^{\kappa_i - 1} \frac{\mathcal{Q}(\mathcal{G}_n(\tilde{p}_{i,j}))}{\kappa_i} \cdot [\mathcal{G}_n(\tilde{p}_{i,j}) + \frac{3j}{\kappa_i} \cdot T_i \cdot (\nabla S_d^T) \tilde{\mathbf{v}}_{i,j}], \tag{12}$$

where $\mathcal{G}_n = \mathcal{G}_s + \mathcal{G}_d$, $\nabla S_d^T = \nabla S'^T - \nabla d^T$, s_{thr} is the safety threshold, $\nabla S'$ is the gradient of S' in $\tilde{p}_{i,j}$, d_{thr} is the collision threshold, $d(\tilde{p}_{i,j})$ is the distance between $\tilde{p}_{i,j}$ and the closet obstacle, ∇d is the gradient of ESDF in $\tilde{p}_{i,j}$. $\mathcal{C}(\cdot) = \max\{\cdot, 0\}^3$ is the cubic penalty, $\mathcal{Q}(\cdot) = \max\{\cdot, 0\}^2$ is the quadratic penalty.

2) Total Time J_t: We minimize the total time $J_t = \sum_{i=1}^M T_i$ to improve the aggressiveness of the trajectory, the gradients are $\partial J_t/\partial \mathbf{c} = \mathbf{0}$ and $\partial J_t/\partial \mathbf{T} = \mathbf{1}$.

3) Smoothness Cost J_m, Dynamical Feasibility J_d: To ensure that the trajectory is smooth enough and can be executed by the agent, we choose the third order of the i^{th} piece trajectory (jerk) as control input, optimize the integration of it and limit the maximum value of velocity and acceleration in the same way as in [28]. Readers can refer to [28] for more details about the cost function and gradients.

VI. Results

A. Implementation details

To validate the performance of our method in real-world applications, we deploy it on a direct-drive self-balancing wheeled bipedal robot called Diablo[2](Fig.1b). All computations are performed by an onboard computer with an Intel Core i7-8550U CPU, which is shown in the right side of Fig.4a. We utilize Fast-lio2 [29] for robust LiDAR-based localization and point cloud map generation. Furthermore, a MPC controller [30] with position and velocity feedback is fitted to the robot for trajectory tracking. The unconstrained optimization problem is solved by an open-source library LBFGS-Lite[3]. It is worth noting that the height of Diablo h_d can be varied in the interval $[h_{min}, h_{max}]$, as shown in Fig.1b. We choose $h = h_{min}$ in point cloud pre-processing and add a custom penalty to constrain this additional degree of freedom:

$$J_h = \sum_{i=1}^{M} \int_0^{T_i} ||p_{i(3)} - h_G(p_i) - h_S(p_i)||^2 dt, \qquad (13)$$

where h_G is the height of ground and h_S is the most suitable height calculated according to the floor height and ceiling height.

In simulation, a desktop PC with an Intel i7-12700 CPU and Nvidia GeForce RTX2060 GPU is used.

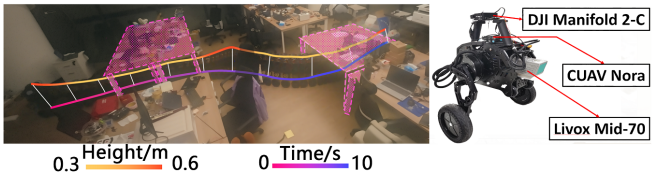
B. Real-World Experiments

We present several experiments in both indoor and outdoor cluttered environments. In the indoor experiment, as shown in Fig.4a, we deliberately keep Diablo at a high altitude to test whether the added z-axis degrees of freedom could help the planner cope with such complex single-level indoor scenarios, thus Diablo needs to crouch down and pass under the table to reach the target.

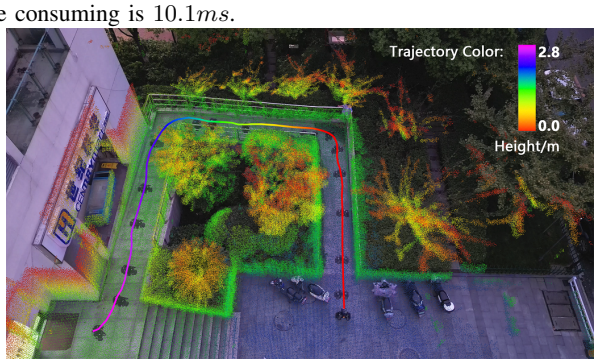
In the outdoor environment, Diablo is demanded to come up to a higher plane, as demonstrated in Fig.4b. There are two ways for humans to ascend this plane, by stairs or through barrier-free access, while for Diablo, there is only one way of the latter. In this test, the point clouds at the stairs are filtered out by the VGF because their normals are too horizontal, allowing the planner to "intelligently" select barrier-free access to generate trajectories. This test also proves that the proposed algorithm can efficiently solve the problems of motion planning in a multi-layered environment.

[2]https://diablo.directdrive.com/

[3]https://github.com/ZJU-FAST-Lab/LBFGS-Lite



(a) Indoor Experiment and Hardware Settings of Diablo: Diablo crouched down and passed under two tables marked as pink dot grid, the lower line is the result of the projection of the trajectory (upper line) onto the ground. The trajectory length is $6.9m$, planning time consuming is $10.1ms$.



(b) Outdoor Experiment: Diablo is demanded to come up to a higher plane via the barrier-free access, the trajectory(rainbow colored line) length is $26.48m$, planning time consuming is $59.89ms$.

Fig. 4: Real-World Experiments

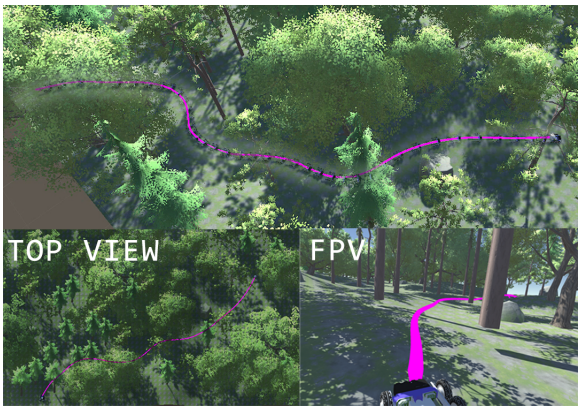


Fig. 5: A four-wheeled differential robot navigating autonomously through the uneven forest.

C. Simulation Experiments

In order to testify the effectiveness of our method in more extensive, rugged and complex environments, we built an open-source simulation environment based on Unity[4]. There are some typical scenarios, including uneven forests, criss-crossing viaducts, multi-level underground parking garages, etc. We simulate a four-wheeled differential robot driving in different environments. Similar to real-world experiments, we use the MPC controller mentioned in Sec.VI-B for trajectory tracking. Fig.5 shows the robot navigating autonomously through one of the simulation scenarios — the uneven forest. More details about the simulation environment are given in

[4]https://unity.com/

TABLE I: Methods Qualitative Comparison

Method	Z-axis freedom	Velocity information	Continuity
Proposed	**Support**	✔	**4-order**
Liu's [18]	No support	✘	0-order
Krüsi's [17]	No support	✘	**4-order**

TABLE II: Methods Quantitative Comparison

Scenario	Method	$t_a(s)$	$len(m)$	$\kappa_m(m^{-1})$
Uneven Forest	Proposed	**0.27**	**85.75**	**0.042**
	Liu's [18]	2.43	86.90	0.174
	Krüsi's [17]	0.92	90.69	0.136
Abandoned Station	Proposed	**1.28**	**62.75**	**0.076**
	Liu's [18]	50.7	72.90	0.252
	Krüsi's [17]	2.9	86.33	0.104

the attached multimedia.

D. Benchmark Comparisons

In this section, we compare the proposed planning algorithm with Liu's method [18] and Krüsi's method [17]. We first qualitatively compared such methods and the results are in Table I. It's clear that only our method support z-axis freedom, which is why Diablo could squat through the table in the indoor experiment. Besides, although Krüsi's method can generate 4^{th} order continuous trajectory, due to the fact that the trajectory is parameterized by the mileage, no information about the robot velocity and acceleration on the trajectory can be obtained. Liu's method only connects points on the point cloud without generating new waypoints, resulting in poor continuity and smoothness of the trajectory, as is evident in the comparison of the mean curvature mentioned in the next paragraph and the visualization of the trajectory in Fig.6.

Further, to quantitatively measure the performance of the trajectory, some generic quantitative evaluation metrics are used to comparing, including terrain assessment time consuming t_a, trajectory length len and mean curvature of the trajectory κ_m which reflects the smoothness of the trajectory. We chose two scenes in our simulation environment, one is the uneven forest mentioned in Sec.VI-C, and the other is an abandoned station with multi-layer structures, as shown in Fig.6. All parameters are finely tuned for the best performance of each compared method.

More than two thousand comparison tests are performed in each case with random starting and ending states. The result is summarized in Table II. It states that the terrain assessment time of Liu's method is much longer than that of the other two. Despite the acceleration by parallel computing with GPUs, it takes a long time to transfer a large number of point clouds between GPUs and CPUs, and the large number of nearest neighbor searches required to construct the connectivity graph with the concept of k-NN also causes the terrain assessment in his method to be extremely time-consuming. Krüsi's method uses RRT-Connect as the front end, but in the optimization based on RRT* [31], the termination condition that takes into account the efficiency leads to insufficient iterations of the algorithm, thus there is a higher probability that the final trajectory is not topologically optimal, resulting in a longer trajectory length, as shown in the abandoned station scenario in Fig.6.

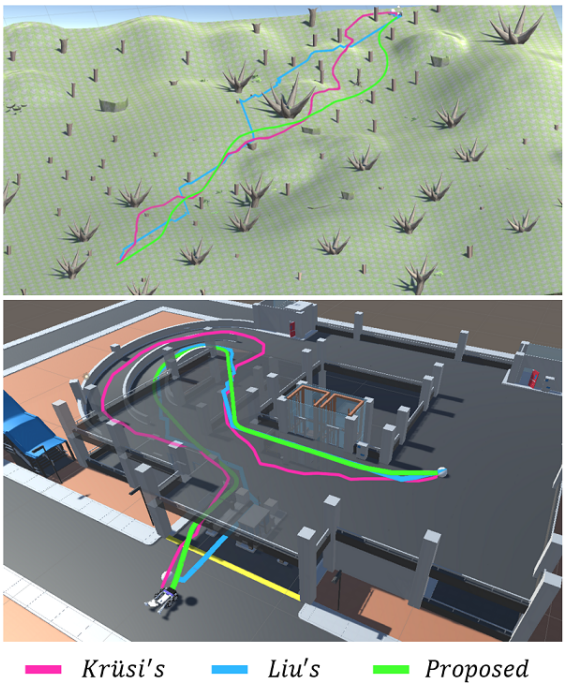


Fig. 6: Trajectory visualization in uneven forest (top) and abandoned station (bottom).

In Table II, our method achieves better performance in terms of terrain assessment time consuming t_a, trajectory length len and mean curvature of the trajectory κ_m, which shows that the terrain assessment algorithm we propose is simpler and more effective, and the trajectories generated based on the proposed safety penalty field and planning framework achieve higher quality.

VII. Conclusion

In this paper, we presented a construction method of a field function defined in $\mathbb{R}^3$ to constrain the motion of ground robot. Using the field function, an optimization-based planning algorithm is proposed for ground robots beyond 2D environment. Real-world experiments and simulation benchmark comparisons validate the efficiency and quality of our method. However, there is still space for improvement for our method. Although generating trajectories beyond 2D environment, the 3D grid map is still somewhat redundant, causing additional memory usage. Moreover, the offline processing limits some online applications.

In the future, our research will focus on the improvement of memory utilization while ensuring the navigation capability beyond 2D environment for ground robots. Besides, to further exploit the advantages of the efficiency of our algorithm, local re-planning will be taken into consideration and adapted to the dynamic environments.

REFERENCES

[1] V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Küng, D. Mannhart, C. Pfister, M. Vierneisel, *et al.*, "Ascento: A two-wheeled jumping robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7515–7521.

[2] C. Zhang, T. Liu, S. Song, and M. Q.-H. Meng, "System design and balance control of a bipedal leg-wheeled robot," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 1869–1874.

[3] P. Štibinger, G. Broughton, F. Majer, Z. Rozsypálek, A. Wang, K. Jindal, A. Zhou, D. Thakur, G. Loianno, T. Krajník, *et al.*, "Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2595–2602, 2021.

[4] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of the research platform of a domestic mobile manipulator utilized for international competition and field test," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7675–7682.

[5] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[6] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1879–1884.

[7] M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3071–3077.

[8] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4889–4895.

[9] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on rrt," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5622–5627.

[10] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2775–2781.

[11] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[12] 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.

[13] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[14] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[15] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, 2019.

[16] W. Ding, L. Zhang, J. Chen, and S. Shen, "Epsilon: An efficient planning system for automated vehicles in highly interactive environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1118–1138, 2021.

[17] 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.

[18] M. Liu, "Robotic online path planning on point cloud," *IEEE transactions on cybernetics*, vol. 46, no. 5, pp. 1217–1228, 2015.

[19] E. Stumm, A. Breitenmoser, F. Pomerleau, C. Pradalier, and R. Siegwart, "Tensor-voting-based navigation for robotic inspection of 3d surfaces using lidar point clouds," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1465–1488, 2012.

[20] H. D. P. Felzenszwalb P F, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[21] 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.

[22] G. Medioni, C.-K. Tang, and M.-S. Lee, "Tensor voting: Theory and applications," in *Proceedings of RFIA*, vol. 2000, 2000.

[23] 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.

[24] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," vol. 4, no. 2, pp. 100–107, 1968.

[27] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, 2022.

[28] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for $\mathrm{SE}(3)$ planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.

[29] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, 2022.

[30] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE Journal*, vol. 39, no. 2, pp. 262–287, 1993.

[31] 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.