2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

An Efficient Trajectory Planner for Car-like Robots on Uneven Terrain

Long Xu[1,2], Kaixin Chai[2,3], Zhichao Han[1,2], Hong Liu[4],
Chao Xu[1,2], Yanjun Cao[2], and Fei Gao[1,2]

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341558

***Abstract*— Autonomous navigation of ground robots on uneven terrain is being considered in more and more tasks. However, uneven terrain will bring two problems to motion planning: how to assess the traversability of the terrain and how to cope with the dynamics model of the robot associated with the terrain. The trajectories generated by existing methods are often too conservative or cannot be tracked well by the controller since the second problem is not well solved. In this paper, we propose *terrain pose mapping* to describe the impact of terrain on the robot. With this mapping, we can obtain the $SE(3)$ state of the robot on uneven terrain for a given state in $SE(2)$. Then, based on it, we present a trajectory optimization framework for car-like robots on uneven terrain that can consider both of the above problems. The trajectories generated by our method conform to the dynamics model of the system without being overly conservative and yet able to be tracked well by the controller. We perform simulations and real-world experiments to validate the efficiency and trajectory quality of our algorithm.**

I. Introduction

An increasing number of tasks require ground robots to navigate autonomously on uneven terrain, such as forest rescue, wilderness exploration, mining transportation, etc. As well as localization, mapping, and control, motion planning is a crucial part of autonomous navigation systems. Existing 2D indoor navigation techniques for ground robots are relatively mature, and there are many open-source, practical motion planning algorithms [1]–[3]. However, none of them can be directly adapted to uneven terrain, since most of them ignore the essential fact that some properties of the terrain (such as height and curvature) will change with spatial location. Considering this fact mainly brings two new problems for motion planning:

1) How do we measure and consider the traversability of terrain in motion planning?

For example, in general, we want the robot to travel on flat terrain instead of steep areas; even for similarly traversable terrain, one with firm, gentle soil while the other filled with rough gravel, we prefer the robot to travel on the former since the latter is more rugged and driving on it may be detrimental to other modules of the robot. Thus, the slope and roughness of the terrain should be considered in the planner.

2) How should we deal with the dynamics model of the robot associated with the terrain?

As shown in the bottom right corner of Fig. 1, suppose the car-like robot is driving on a sloping surface. The throttle required for the robot to reach the same acceleration in the body frame is different for uphill and downhill due to the presence of gravity. If the planner ignores the terrain in the dynamics model, the planned trajectory may be infeasible for the robot. Thus, it is necessary to handle the dynamics coupled with the terrain so that the controller can better track the trajectory.

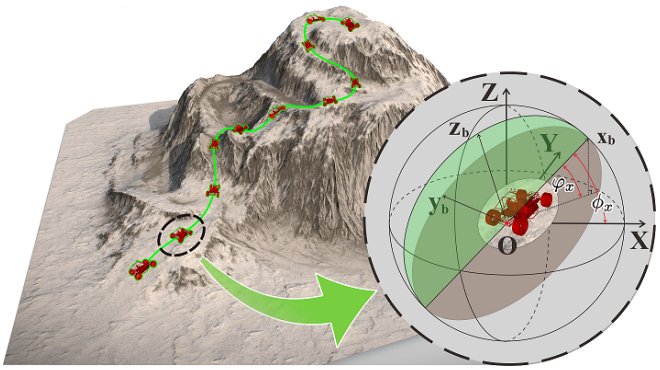


Fig. 1: A car-like robot driving on uneven terrain, the green line indicates the trajectory optimized by the proposed method.

An effective planner for robots on uneven terrain should consider both of these problems. The first problem is related to the safety of the robot, and the second problem is related to the executability of the trajectory. Attributed to the nonlinearity of the terrain, the rising dimensionality of the problem due to the high-dimensional robot state space, and the coupling of the robot dynamics model with the changing terrain, most existing work [4]–[9] cannot address the second problem well. As a result, the trajectories generated by these methods are often too conservative or cannot be tracked well by the controller.

To compensate for the shortcomings of existing methods due to the above factors, in this paper, through a mapping for describing the interaction between the terrain and the robot, we design an efficient optimization-based planning framework for car-like robots, which allows to considering both

This work was supported by the Fundamental Research Funds for the Central Universities and the National Natural Science Foundation of China under grant no. 62003299.

[1]State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China.
[2]Huzhou Institute of Zhejiang University, Huzhou 313000, China.
[3]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, Shaanxi, China.
[4]School of information and electrical engineering, Hangzhou City University, Hangzhou 310015, China.
Corresponding authors: Hong Liu, Fei Gao. E-mail: {gaolon, fgaoaa}@zju.edu.cn, liuhong@zucc.edu.cn

the first and the second problem effectively. The trajectories generated by our method conform to the dynamics model of the system without being overly conservative and yet able to be tracked well by the controller.

We model the motion planning problem of car-like robots on uneven terrain as an optimal control problem. Then, we propose *terrain pose mapping* to describe the impact of terrain on the robot. With this mapping, we can obtain the $SE(3)$ state of the robot on uneven terrain for a given state in $SE(2)$. Also, we will show an algorithm that approximately constructs this mapping. Finally, we use piecewise polynomials to represent the trajectory and simplify the problem to a nonlinear constrained optimization problem containing constraints of non-holonomic dynamics, curvature, etc, which can be solved using numerical optimization algorithms. Besides, We perform comprehensive tests in simulation and real world to validate our method. Contributions of this paper are:

1) We propose *terrain pose mapping* to describe the impact of terrain on the robot and present an efficient algorithm to construct it approximately.
2) We propose an optimization-based planning framework for car-like robots on uneven terrain, which allows considering terrain curvature and dynamics of the robot.
3) We open source our software[1] for the reference of the community.

II. Related Works

A. Motion Planning for Car-like Robots

The popularity of autonomous driving has extensively promoted the research of motion planning for car-like robots. Existing approaches can be roughly divided into sampling-based and optimization-based methods. The former is represented by RRT* [10] and its variants, which ensure global optimality with sufficiently dense sampling. For the non-holonomic constraint of car-like robots, many algorithms that consider the nonlinear dynamics of the robot while sampling have been proposed [11]–[14], enhancing the efficiency of sampling-based planners in more scenarios. Although sampling-based methods can avoid local minima in non-convex environments, they confront a dilemma between computation overhead and trajectory quality.

Optimization-based approaches [15]–[20] usually model the motion planning problem of a car-like robot as an optimal control problem (OCP) and represent the trajectory with discrete state points, then simplify the problem to nonlinear model predictive control (NMPC) problem [18] or simpler quadratically constrained quadratic programming, quadratic programming [15], [16], which can be solved by numerical optimization algorithms.

B. Motion Planning on Uneven Terrain

In recent years, many works are trying to solve the two problems mentioned in Sec.I, and most of them focus on the first one by combining various geometric information [4]–[8], exploiting conditional value-at-risk [21], fusing semantic information [22], [23], etc. For the second problem, most of them either avoid it in planning [5], [8] or consider only the geometric properties of the trajectory [4], [7], leaving the other problems to the controller.

Krüsi et al. [4] presented a practical approach to global motion planning and terrain assessment for car-like robots in generic 3D environments, which assesses the traversability of terrain on demand during motion planning. However, in the face of complex environments, this work often requires a huge number of samples to generate trajectories, which does not guarantee real-time performance. To address the shortcomings of this method, Jian et al. [6] proposed a plane-fitting based uneven terrain navigation framework, which utilizes informed-RRT* [24] as the front end. It refines the path using Gaussian Process Regression [25] and finally generates dynamically feasible local trajectories by solving an NMPC problem. Although this work considers the changing terrain within NMPC, their method is inefficient in generating long trajectories because the complexity of the NMPC problem will rise dramatically as the planning horizon becomes longer. In order to assess the traversability more precisely, Zhang et al. [26] considers the suspension system when the robot is stationary, but it requires an accurate identification of the robot's physical parameters and estimation of landing points of the four tires, which is less efficient.

The work [27] uses surfel to represent the points cloud, incorporating both kinematic and physical constraints of robots, enabling efficient sampling-based planners for challenging navigation on uneven terrain. Nevertheless, the generated trajectory does not contain information about the robot's speed and acceleration with respect to time. Besides, when employing Dubins or Reeds Sheep state spaces, this method takes too long to converge to near-optimal paths. Wang et al. [9] proposed an optimization-based planning framework for ground robots considering both active and passive height changes on the z-axis. The trajectories planned by their method can provide dynamical information related to time and have excellent smoothness, benefiting from the penalty field for chassis motion constraints defined in $\mathbb{R}^3$. Although this method considers the velocity and acceleration of the trajectory in 3D space, the optimized trajectory is often dynamic infeasible or too conservative as the dynamics coupled with the changing terrain is not taken into account.

III. Planning Framework

A general motion planning problem for ground robots on uneven terrain can be expressed as the following optimal control problem:

$$\min_{\mathbf{u}(t)} \int_0^{t_f} \tau(\mathbf{s}(t), \mathbf{s}^{(1)}(t), ..., \mathbf{s}^{(s)}(t), \mathbf{u}(t))dt + \rho(t_f) \quad (1)$$

$$s.t. \quad \dot{\mathbf{s}}(t) = f(\mathbf{s}(t), \mathbf{u}(t)), \quad (2)$$

$$\text{Ter}(\mathbf{s}(t)) = 0, \quad (3)$$

$$C_p(\mathbf{s}(t), \mathbf{s}^{(1)}(t), ..., \mathbf{s}^{(s)}(t), \mathbf{u}(t)) \preceq \mathbf{0}, \quad (4)$$

where $\mathbf{s}$ is the state of the robot, $\mathbf{s}^{(*)}$ is the $*$-th order derivative of $\mathbf{s}$, $\mathbf{u}$ is the control input of the robot, ρ :

[1] https://github.com/ZJU-FAST-Lab/uneven_planner

$[0,\infty) \mapsto [0,\infty)$ is the time regularization. $\tau(*)$ denotes the cost associated with the task, such as energy, risk, etc. Eq.(2) and Eq.(3) denote the state transfer equation and terrain contact constraint, respectively. In this work, we assume that the robot must be in contact with the terrain and its wheels do not slip. Eq.(4), where $C_p(*) : * \mapsto \mathbb{R}^{N_p}$ and $\mathbf{0} = [0, 0, .., 0]^{\mathrm{T}} \in \mathbb{R}^{N_p}$, denote N_p inequality constraints of the robot, including dynamics constraints, traversability constraints, etc.

In this section, we will present how we simplify the state description of a car-like robot on uneven terrain by a mapping $\mathscr{F}$ called *terrain pose mapping*, and give a specific algorithmic procedure for constructing this mapping. Then, we parameterize the state trajectory as piecewise polynomials and give explicit expressions for the control inputs, dynamical variables with the help of this mapping and the state transfer equation. Since $\tau(*)$ may differ from task to task, in this paper, we propose an exemplary cost function that combines terrain curvature and trajectory smoothness. Finally, we will analyze the more specific trajectory optimization problem after these processes, which can be solved using numerical optimization algorithms.

A. Terrain Pose Mapping

Using the simplified bicycle model [28] and describing the state of a car-like robot on uneven terrain by a $SE(3)$ state with the position $\mathbf{p} = [x, y, z]^{\mathrm{T}} \in \mathbb{R}^3$ and the attitude $\mathbf{R} = [\mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b] \in SO(3)$, we can write the robot's model as follows:

$$\dot{\mathbf{p}} = \mathbf{x}_b \cdot v_x, \tag{5}$$

$$\dot{\mathbf{R}} = \mathbf{R}\lfloor \frac{v_x \tan\delta}{L_w} \cdot \mathbf{z}_b \rfloor, \tag{6}$$

where v_x is the velocity along the body axis $\mathbf{x}_b$, δ is the steering angle, L_w is the wheelbase length of the robots, the operation $\lfloor * \rfloor$ takes a vector to a skew-symmetric matrix. It is worth noting that on flat ground, the state of the robot can be represented using only $\mathbf{s}_r = [x, y, \theta]^{\mathrm{T}} \in SE(2)$. However, due to the presence of uneven terrain, we need to use a higher dimensional representation for the state.

In this work, we consider the terrain contact constraint (4) brings about actually a mapping relation called *terrain pose mapping* $\mathscr{F} : SE(2) \mapsto \mathbb{R} \times \mathbb{S}^2_+$, where $\mathbb{S}^2_+ \triangleq \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\|_2 = 1, \mathbf{x} \cdot \mathbf{b}_3 > 0\}, \mathbf{b}_3 = [0, 0, 1]^{\mathrm{T}}$. This means that the state in $SE(2)$, which is originally on flat ground, is given height and attitude, with elements in $\mathbb{R}$ representing the height z, elements in $\mathbb{S}^2_+$ representing the body axis $\mathbf{z}_b$. It is worth noting that we do not use the entire 2D sphere $\mathbb{S}^2 \triangleq \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\|_2 = 1\}$, but $\mathbb{S}^2_+$, because in a common task, we do not want the body axis $\mathbf{z}_b$ to face the lower half-plane, which requires high speed and is not safe.

Let $\mathbf{x}_{yaw} = [\cos\theta, \sin\theta, 0]^{\mathrm{T}}$ denote the direction of yaw angle, the mapping $\mathscr{F}$ can be expressed as two functions:

$$z = f_1(x, y, \theta), \tag{7}$$

$$\mathbf{z}_b = \mathbf{f}_2(x, y, \theta). \tag{8}$$

Using the Z-X-Y Euler angles to represent the attitude of the robot, we can obtain:

$$\mathbf{p} = [x, y, z]^{\mathrm{T}} = [x, y, f_1(x, y, \theta)]^{\mathrm{T}}, \tag{9}$$

$$\mathbf{y}_b = \frac{\mathbf{f}_2(x, y, \theta) \times \mathbf{x}_{yaw}}{\|\mathbf{f}_2(x, y, \theta) \times \mathbf{x}_{yaw}\|}, \tag{10}$$

$$\mathbf{x}_b = \mathbf{y}_b \times \mathbf{f}_2(x, y, \theta). \tag{11}$$

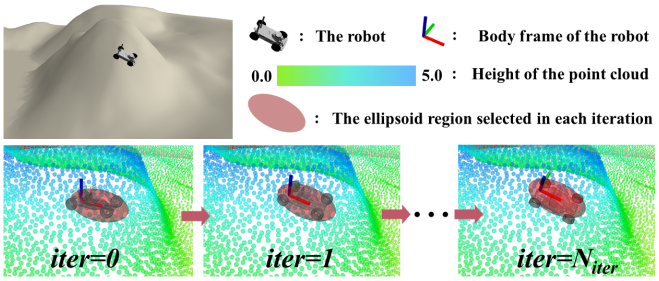
Thus, the terrain contact constraint allows us to still use elements in $SE(2)$ to describe the state of the robot on uneven terrain.

Many methods can be used to construct the mapping $\mathscr{F}$; the most accurate method is to remotely drive the robot onto real terrain, collect accurate localization data and fit it, but this method is ineffective for regions that the robot has not reached. Usually, it is easier for the robot to obtain the point cloud of the environment by some LIDAR-based SLAM algorithms [29]. Thus, in this paper, we propose a simple and efficient method to obtain the mapping $\mathscr{F}$ from the point cloud by using an iterative plane-fitting strategy while considering the size and attitude of the robot. The pipeline for processing each $SE(2)$ state is shown in Algorithm 1.

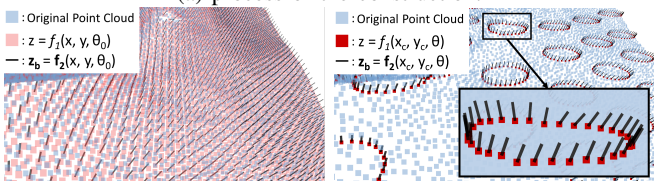
Algorithm 1: Get the result of $\mathscr{F}$ at a $SE(2)$ state

Input: state $\mathbf{s}_r \in SE(2)$, Iteration times N_{iter}, Ellipsoidal parameters (e_x, e_y, e_z)
Output: $\mathbf{z}_b$, z
begin
  $\mathbf{z}_b \leftarrow \mathbf{b}_3$;
  $z \leftarrow$ FindNearestXYPointZ($\mathbf{s}_r$);
  for ***each*** $i \in N_{iter}$ **do**
    $(p_i, R_i) \leftarrow$ CalculateSE3($\mathbf{s}_r, z, \mathbf{z}_b$);
    $G_i \leftarrow$ FindEllipsoidPoints(p_i, R_i, e_x, e_y, e_z);
    $p_{mean} \leftarrow$ GetMeanPosition(G_i);
    $Cov \leftarrow$ ZeroSquareMatrix3();
    for ***each*** $p_j \in G_i$ **do**
      $p_e \leftarrow p_j - p_{mean}$;
      $Cov \leftarrow Cov + p_e p_e^{\mathrm{T}}$;
    $p_{mean} \leftarrow p_{mean}/$NumOf($M_i$);
    $Cov \leftarrow Cov/$NumOf(M_i);
    $\mathbf{z}_b \leftarrow$ GetMinEigenVec(Cov);
    $z \leftarrow p_{mean}$.GetZ();
  return $\mathbf{z}_b$, z;

For a $SE(2)$ state, $\mathbf{z}_b$ is first initialized to $\mathbf{b}_3$, then we search the point cloud for a neighboring point whose (x, y) coordinate is nearest, and use the height of this point as the initial value of z. Thus, we can obtain the corresponding $SE(3)$ state of the robot. Next, the points in an ellipsoidal region related to the robot's size, position, and attitude will be taken out. And so, $\mathbf{z}_b$ will be updated by the eigenvector corresponding to the smallest eigenvalue of their covariance matrix, z will be updated by the average of the heights of the points, which allows us to obtain the new ellipsoid region for the next iteration.



(a) process of the construction.



(b) when $\theta = \theta_0$. (c) when $(x, y) = (x_c, y_c)$.

Fig. 2: The Process and Results of Constructing Mapping $\mathscr{F}$. Figure (a) illustrates the process of Algorithm 1, where the gray vehicles indicate the robot poses obtained in each iteration. Figures (b) and (c) show the different z and $\mathbf{z}_b$ when θ or position (x, y) is fixed, respectively, where the blue points are the original point cloud. The heights of the red points in Figure (b) and Figure (c) indicate the height $z \in \mathbb{R}$. The black line on each point indicates the $\mathbf{z}_b \in \mathbb{S}^2_+$ with the direction pointing from the point to the sky. In Figure (c), black lines and red dot heights on each circle indicate $\mathbf{z}_b$ and z respectively when the position is the center of the circle (x_c, y_c) but $\theta \in SO(2)$ is different. Here, $\mathbf{z}_b$ may vary with θ at the same position (x_c, y_c), since the neighboring area of the robot may be rugged rather than a flat plane.

We discretize $SE(2)$ space into grids and fit $\mathscr{F}$ for the state corresponding to each grid. When the value or gradient corresponding to a $SE(2)$ state is required, we then compute it using trilinear interpolation, where operations on manifold [30] are used for the processing of $SO(2)$ and $\mathbb{S}^2_+$. Fig. 2 illustrates the process and results of constructing mapping $\mathscr{F}$ through the example environments.

B. Trajectory Parameterization

In this paper, we use quintic piecewise polynomials to represent state trajectories. Since the mapping $\mathscr{F}$ allows state space of the robot still be $SE(2)$, each piece of trajectory can be denoted as:

$$x_i(t) = c_{x_i}^{\mathrm{T}} \gamma(t), \quad t \in [0, T_i]; \tag{12}$$
$$y_j(t) = c_{y_j}^{\mathrm{T}} \gamma(t), \quad t \in [0, T_j]; \tag{13}$$
$$\theta_k(t) = c_{\theta_k}^{\mathrm{T}} \gamma(t), \quad t \in [0, T_k]; \tag{14}$$

where $i = 1, 2, ..., N_i; j = 1, 2, ..., N_j; k = 1, 2, ..., N_k$ is the index of piecewise polynomial, $T_*, * = \{i, j, k\}$ is the duration of a piece of the trajectory, $c_* \in \mathbb{R}^6, * = \{x_i, y_j, \theta_k\}$ is the coefficient of polynomial, $\gamma(t) = [1, t, t^2, ..., t^5]^{\mathrm{T}}$ is the natural base.

We also add the constraint that the trajectory is four times continuously differentiable at the segmented points to obtain more continuous trajectories. Thus, let the whole trajectory be $\mathbf{x}(t) = [x(t), y(t), \theta(t)]^{\mathrm{T}}$, $\mathbf{y}_{yaw} = [-\sin\theta, \cos\theta, 0]^{\mathrm{T}}$, combining with Eq.(5) and Eq.(6), we can compute the control inputs and dynamical variables analytically as follows:

$$v_x = \frac{v}{\cos\phi_x}, \tag{15}$$
$$a_x = \frac{a_t}{\cos\phi_x} + g\sin\varphi_x, \tag{16}$$
$$a_y = \frac{a_n}{\cos\phi_y} + g\sin\varphi_y, \tag{17}$$
$$\omega_z = \frac{\omega}{\cos\xi}, \tag{18}$$
$$\kappa = \frac{v_x}{\omega_z}, \tag{19}$$
$$\delta = \arctan(L_w \cdot \kappa), \tag{20}$$

where

$$v = \sqrt{\dot{x}^2 + \dot{y}^2}, \tag{21}$$
$$a_t = \ddot{x}\cos\theta + \ddot{y}\sin\theta, \tag{22}$$
$$a_n = -\ddot{x}\sin\theta + \ddot{y}\cos\theta, \tag{23}$$
$$\omega = \dot{\theta}, \tag{24}$$
$$\cos\phi_x = \mathbf{x}_b^{\mathrm{T}}\mathbf{x}_{yaw}, \quad \cos\phi_y = \mathbf{y}_b^{\mathrm{T}}\mathbf{y}_{yaw}, \tag{25}$$
$$\sin\varphi_x = \mathbf{x}_b^{\mathrm{T}}\mathbf{b}_3, \quad \sin\varphi_y = \mathbf{y}_b^{\mathrm{T}}\mathbf{b}_3, \tag{26}$$
$$\cos\xi = \mathbf{z}_b^{\mathrm{T}}\mathbf{b}_3, \tag{27}$$

$a_x, a_y, \omega_z, \kappa$ denote longitude acceleration, latitude acceleration, angular velocity, and curvature, respectively. It is worth noting that $\cos\phi_x, \cos\phi_y, \sin\varphi_x, \sin\varphi_y, \cos\xi$ are all related to Eq.(8). When $\mathbf{z}_b = \mathbf{b}_3$, both control inputs and dynamical variables degenerate to the case that the ground is flat.

C. Trajectory Optimization

In this paper, we propose the cost function $\tau = \mathbf{j}(t)^{\mathrm{T}}\mathbf{j}(t) + \rho_{\text{ter}} \cdot \sigma(\mathbf{x}(t))$, where $\mathbf{j}(t) = \mathbf{x}^{(3)}(t)$ denotes the jerk of the trajectory, and its square integral represents the smoothness of the trajectory, ρ_{ter} is a constant. $\sigma(\mathbf{x}(t))$ is *Surface Variation* proposed by [31] to approximate the terrain curvature, which can be obtained while calculating $\mathscr{F}$, since $\sigma(\mathbf{x}) = \lambda_0 / \sum_{i=0}^{2} \lambda_i$, where $\lambda_0 \le \lambda_1 \le \lambda_2$ are eigenvalues of the covariance matrix of points in the ellipsoidal region near $\mathbf{x}$.

We formulate the trajectory optimization problem for car-like robots on uneven terrain as:

$$\min_{\mathbf{c}_{xy}, \mathbf{c}_\theta, \mathbf{T}_{xy}, \mathbf{T}_\theta} \int_0^{T_s} (\mathbf{j}(t)^{\mathrm{T}}\mathbf{j}(t) + \rho_{\text{ter}}\sigma(\mathbf{x}(t)))dt + \rho_T T_s \tag{28}$$
$$s.t. \quad \dot{x}\sin\theta - \dot{y}\cos\theta = 0, \tag{29}$$
$$\mathbf{M}_{xy}\mathbf{c}_{xy} = \mathbf{b}_{xy}, \quad \mathbf{M}_\theta\mathbf{c}_\theta = \mathbf{b}_\theta, \tag{30}$$
$$\mathbf{T}_{xy} \succeq \mathbf{0}, \quad \mathbf{T}_\theta \succeq \mathbf{0}, \tag{31}$$
$$v_x^2 - v_{max}^2 \le 0, \tag{32}$$
$$a_x^2 - a_{mlon}^2 \le 0, \tag{33}$$
$$a_y^2 - a_{mlat}^2 \le 0, \tag{34}$$
$$\frac{\omega_z^2}{v_x^2 + \delta_+} - \frac{\tan^2\delta_{max}}{L_w^2} \le 0, \tag{35}$$
$$c_{min} - \cos\xi \le 0, \tag{36}$$
$$\sigma(\mathbf{x}) - \sigma_{max} \le 0, \tag{37}$$

where $\mathbf{c}_{xy} = [[c_{x_1}, c_{y_1}]^{\mathrm{T}}, [c_{x_2}, c_{y_2}]^{\mathrm{T}}, ..., [c_{x_M}, c_{y_M}]^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6M\times 2}$, $\mathbf{c}_\theta = [c_{\theta_1}^{\mathrm{T}}, c_{\theta_2}^{\mathrm{T}}, ..., c_{\theta_\Omega}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6\Omega\times 1}$ are coefficient matrix, M and Ω represent the number of pieces of the piecewise polynomial of x, y and θ, respectively. In this paper, we make the trajectory of theta have more pieces (i.e. $\Omega > M$) to better fit the non-holonomic constraint (29).

$\mathbf{T}_{xy} = [T_{1xy}, T_{2xy}, ..., T_{Mxy}]^{\mathrm{T}} \in \mathbb{R}^M$, $\mathbf{T}_\theta = [T_{1\theta}, T_{2\theta}, ..., T_{\Omega\theta}]^{\mathrm{T}} \in \mathbb{R}^\Omega$ are time vectors, satisfying $\|\mathbf{T}_{xy}\|_1 = \|\mathbf{T}_\theta\|_1 = T_s$. Eq.(30) is combination of the continuity constraint mentioned in last subsection and boundary condition of the trajectory:

$$[\mathbf{x}(0), \mathbf{x}^{(1)}(0), \mathbf{x}^{(2)}(0)] = [\mathbf{x}_{init}, \mathbf{x}_{init}^{(1)}, \mathbf{x}_{init}^{(2)}], \tag{38}$$

$$[\mathbf{x}(T_s), \mathbf{x}^{(1)}(T_s), \mathbf{x}^{(2)}(T_s)] = [\mathbf{x}_{fina}, \mathbf{x}_{fina}^{(1)}, \mathbf{x}_{fina}^{(2)}], \tag{39}$$

so $\mathbf{M}_{xy} \in \mathbb{R}^{(5M+1)\times 6M}$, $\mathbf{M}_\theta \in \mathbb{R}^{(5\Omega+1)\times 6\Omega}$.

Conditions (32)$\sim$(35) are dynamic feasibility constrains, including limitation of longitudinal velocity v_x, longitude acceleration a_x, latitude acceleration a_y, and curvature κ, where $v_{max}^2, a_{mlon}^2, a_{mlat}^2, \delta_{max}$ are constants. δ_+ in condition (35) is a very small positive constant to avoid the zero denominators. Conditions (36) and (37) are limitations of attitude and terrain curvature, where c_{min}, σ_{max} are constants. We consider it unsafe for the robot to have too large ξ or too large terrain curvature.

To deal with Eq.(30), we use the method proposed by work [32], which allows eliminating Eq.(30) and converting the optimization variable $\mathbf{c}_{xy}, \mathbf{c}_\theta$ to the segmentation point positions $\mathbf{q}_{xy} \in \mathbb{R}^{(M-1)\times 2}$ and $\mathbf{q}_\theta \in \mathbb{R}^{\Omega-1}$, thus reducing the dimensionality of the problem. Moreover, we refer to the differential homogeneous mapping [33], using the C^2 function mentioned in work [34] to map each element $T_* \in \mathbb{R}_+$ in $\mathbf{T}_{xy}$ and $\mathbf{T}_\theta$ to $\mathbb{R}$, eliminating the constraints (31). As for the remaining non-holonomic constraint (29) and other inequality constraints (32)$\sim$(37), we discretize each piece of the duration T_{ixy} as K time stamps $\tilde{t}_{ij} = (j/K) \cdot T_{ixy}$, $(i = 1, 2, ..., M_{xy}, j = 0, 1, ..., K-1)$, and impose the remaining constraints on these time stamps, thus the number of constraints of the problem (28) becomes $7KM_{xy}$. We also use this method to discretize $\sigma(\mathbf{x}(t))$ and obtain its integral accumulatively.

Then, we use PHR Augmented Lagrange Multiplier method [35] (PHR-ALM) to solve the simplified problem, which is a method for solving constrained optimization problems, smoothing the dual function by adding quadratic terms, iteratively solving the approximate unconstrained problem and updating the dual variables. Besides, L-BFGS [36] is chosen as the unconstrained optimization algorithm to work with PHR-ALM. It is a quasi-Newton method that can estimate the Hessian matrix from the previous objective function values and gradients with limited memory.

Last but not least, the gradients of the objective function and the constraints need to be calculated explicitly. It is easy to derive them using the chain rule, except for σ and $\cos\phi_x, \cos\phi_y, \sin\varphi_x, \sin\varphi_y, \cos\xi$ with respect to $\mathbf{f}_2$. For σ, since we can obtain it together with $\mathbf{f}_2$, the gradient can be obtained using trilinear interpolation as well; for the others, let $\mathbf{z}_b \triangleq [a, b, c]^{\mathrm{T}}$, $\sin\theta \triangleq s_\theta$, $\cos\theta \triangleq c_\theta$, $r \triangleq c_\theta a + s_\theta b$, $s \triangleq as_\theta - bc_\theta$, we give their gradients with respect to state $\mathbf{x}$ as follows:

$$\nabla_{\mathbf{x}} \cos\phi_x = \nabla_{\mathbf{x}}(\sqrt{1-r^2}) = -r(1-r^2)^{-\frac{1}{2}} \nabla_{\mathbf{x}} r, \tag{40}$$

$$\nabla_{\mathbf{x}} \cos\phi_y = \nabla_{\mathbf{x}}(\frac{c}{\sqrt{1-r^2}}) \tag{41}$$

$$= (1-r^2)^{-\frac{1}{2}} \nabla_{\mathbf{x}} c + r(1-r^2)^{-\frac{3}{2}} c \nabla_{\mathbf{x}} r, \tag{42}$$

$$\nabla_{\mathbf{x}} \sin\varphi_x = \nabla_{\mathbf{x}}(-\frac{rc}{\sqrt{1-r^2}}) \tag{43}$$

$$= -r(1-r^2)^{-\frac{1}{2}} \nabla_{\mathbf{x}} c - (1-r^2)^{-\frac{3}{2}} c \nabla_{\mathbf{x}} r \tag{44}$$

$$\nabla_{\mathbf{x}} \sin\varphi_y = \nabla_{\mathbf{x}}(\frac{s}{\sqrt{1-r^2}}), \tag{45}$$

$$= (1-r^2)^{-\frac{1}{2}} \nabla_{\mathbf{x}} s + r(1-r^2)^{-\frac{3}{2}} s \nabla_{\mathbf{x}} r, \tag{46}$$

$$\nabla_{\mathbf{x}} \cos\xi = \nabla_{\mathbf{x}}(\mathbf{b}_3^{\mathrm{T}} \mathbf{z}_b) = \nabla_{\mathbf{x}} c. \tag{47}$$

In this work, we use the points in a unit circle on the X-Y plane to represent $\mathbf{z}_b \in \mathbb{S}_+^2$, i.e., $a^2 + b^2 < 1, c = \sqrt{1-a^2-b^2}$. Thus, $\nabla_{\mathbf{x}} s, \nabla_{\mathbf{x}} r$ are easily obtained since $\nabla_{\mathbf{x}} a, \nabla_{\mathbf{x}} b$ are obtained by trilinear interpolation. For $\nabla_{\mathbf{x}} c$, due to $c = \sqrt{1-a^2-b^2}$, $\nabla_{\mathbf{x}} c = -(a\nabla_{\mathbf{x}} a + b\nabla_{\mathbf{x}} b)/c$.

IV. Results

A. Implementation details

In order to validate the performance of our method in real-world applications, we deploy it on a car-like robot, as shown in Fig. 3. All computations are performed by an onboard computer NVIDIA Jetson Nano. We utilize NOKOV Motion Capture System[2] for localization. Furthermore, a MPC controller [37] with position feedback is fitted to the robot for trajectory tracking. We adopt the lightweight hybridA* algorithm as the front end of the planner and use Dubins Curve [38] to shoot the end state for earlier termination of the search process. Besides, the implementation of the L-BFGS utilizes an open-source library LBFGS-Lite[3]. All simulations are run on a desktop with an Intel i7-12700 CPU.

B. Real-World Experiments

In the real-world experiments, as seen in Fig. 3, we require the car-like robot to traverse two terrain models made of foam with distinct raised areas, obstacles, and slopes in order to test whether the proposed mapping $\mathscr{F}$ can help the planner generate safe and dynamically feasible trajectories for robots on uneven terrain. Limitation of longitudinal velocity, longitude acceleration, latitude acceleration, steering angle, attitude, and terrain curvature are set to $v_{max} = 0.8m/s^2$, $a_{mlon} = 5.0m/s^2$, $a_{mlat} = 5.0m/s^2$, $\delta_{max} = 0.505$, $c_{min} = 0.86$ and $\sigma_{max} = 0.05$, respectively. Meanwhile, we set the time weight $\rho_T = 500$ and the terrain curvature weight $\rho_{\text{ter}} = 10$ to ensure the aggressiveness of the trajectory. Fig. 6 shows some cases with different starting and ending points in our tests.

Furthermore, to know the variation of relevant variables (e.g., attitude $\hat{\xi}$, forward velocity $\hat{v}_x$, etc.) during the motion

[2] https://en.nokov.com/

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

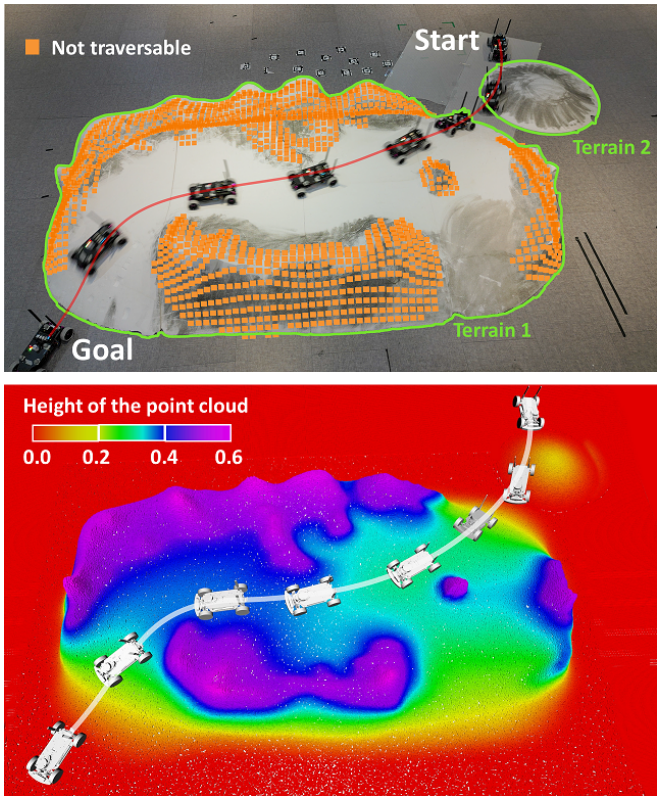


Fig. 3: Real-World Experiments: In this case, the car-like robot needs to go downhill and uphill twice, traverse two terrains, and avoid the not traversable area determined by conditions (36) and (37). The length of the planned trajectory in 3D space (the red curve) is $5.496m$, planning time consuming is $1.603s$. The bottom half of this figure shows the visualization in RViz.

of the robot, we make statistics in Tab. I for the case shown in Fig. 3. As we can see, the robot maintains a relatively high speed and small tracking error throughout to reach the target state without violating the constraints we set. More demonstrations can be found in the attached multimedia.

C. Simulation Experiments

To testify the effectiveness of our method in more extensive and complex environments, we build an open-source simulation environment based on Gazebo[4]. There are some typical uneven terrain, including deserts, mining area, volcanic area, etc. We simulate a car-like robot driving in different environments. For example, in the uneven forest, the robot needs to avoid obvious obstacles such as shrubs and trees; in an extinct volcanic environment, the robot needs to find a flatter entrance when crossing the crater. As for desert and mining area, the robot must maintain a balance between traversability and path length while planning the trajectory to traverse them. Fig. 4 shows the robot navigating autonomously through different simulation scenarios.

[4]https://gazebosim.org/

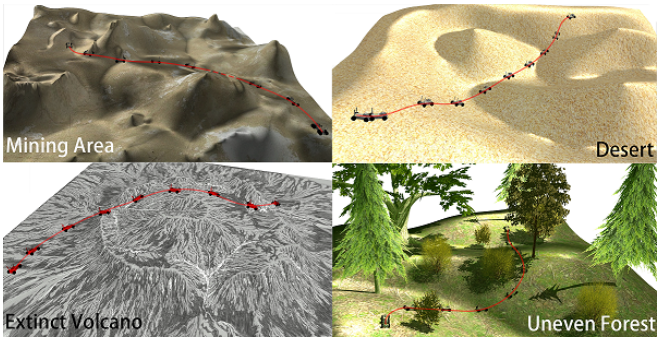


Fig. 4: A car-like robot navigating autonomously through different uneven terrain.

TABLE I: Statistics in Real-World Experiments

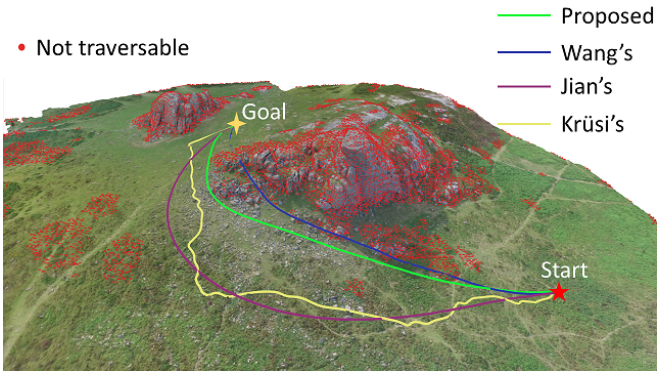
Statistics	Mean	Max	STD.
$\hat{v}_x$ (m/s)	0.684	0.800	0.212
$\hat{\xi}$ (rad)	0.117	0.366	0.108
Tracking Error (m)	0.059	0.109	0.025

D. Benchmark Comparisons

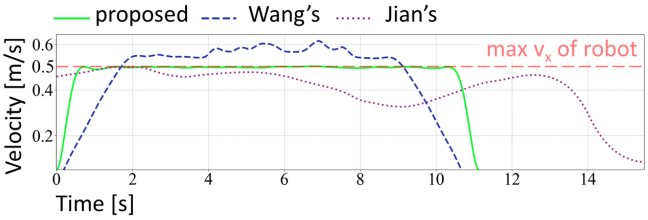
In this subsection, we compare the proposed planning algorithm with Krüsi's [4], Jian's [6] and Wang's [9] methods. We first compared the velocity curve of the trajectory with the last two works. The comparison is conducted on mountainous terrain, as shown in Fig. 5a. In addition, the maximum velocity and longitude acceleration were set to $v_{max} = 0.5m/s$ and $a_{mlon} = 5.0m/s^2$, respectively. As can be seen from the curves in Fig. 5b, our trajectory planner can better utilize the maneuverability of the robot. The trajectory planned by Wang's method [9] is shorter. However, because their method does not consider the terrain contact constraint, it is easy to optimize the trajectory to a not traversable area. Moreover, the generated trajectory violates the constraint of maximum velocity(v_x) of the robot since the dynamics coupled with the changing terrain is not taken into account. Regarding Jian's method [6], although the changing terrain is modeled in the NMPC problem, for two reasons, the trajectory planned by their method is less optimal and has a longer execution time and length. The first reason is that they do not consider the execution time in the optimization. The second one is the insufficient iterations of the sampling-based front end.

To further quantitatively measure the performance of the trajectory with existing methods, some generic evaluation metrics are used to compare, including the mean of planning time consuming (t_p), mean curvature of the trajectory (κ_m) which reflects the smoothness of the trajectory, and mean of tracking error (Tra_{err}). We chose four scenes in our simulation environment mentioned in Sec. IV-C. All parameters are finely tuned for the best performance of each method.

More than one thousand comparison tests are performed in each scene with random starting and ending states. Since the trajectory generated by Krüsi's method [4] does not provide



(a) Trajectory visualization in mountainous terrain.



(b) Comparison of velocity curves of trajectories.

Fig. 5: Trajectory Visualization and Comparison

TABLE II: Benchmark Comparison

Method	$t_p(s)$	$\kappa_m(m^{-1})$	$\text{Tra}_{err}(m)$
proposed	**0.301**	**0.710**	**0.086**
Krüsi's [4]	4.696	1.530	-
Jian's [6]	0.450	0.910	0.101
Wang's [9]	0.960	0.717	0.117

time-related information, we did not count its Tra_{err}. The result is summarized in Table II. It states that mean planning time of Krüsi's method [4] is much longer than that of other three. This is because the maximum curvature constraint of the robot limits the size of the trajectory library used in this method. Wang's method [9] also has a long planning time due to the absence of a better heuristic function for graph search. Our trajectories are smooth and have a small mean curvature since the polynomial is used to represent the trajectory and the norm of its high-order derivatives is minimized. Moreover, thanks to our rational formulation of the problem and optimization strategy providing better continuity of the robot states and their finite-dimensional derivatives, as well as our effective consideration of traversability using terrain curvature, our trajectories are easier to track by the controller.

In Table II, our method achieves better performance in terms of mean of planning time consuming t_p, mean curvature of the trajectory κ_m and mean of tracking error Tra_{err}, which shows that the algorithm we propose is more efficient and effective. The trajectories generated based on the proposed planning framework have higher quality.

V. Conclusion

In this paper, we propose the mapping $\mathscr{F} : SE(2) \mapsto \mathbb{R} \times \mathbb{S}^2_+$ to describe the impact of terrain on the robot and present an efficient algorithm to construct it approximately. Using this mapping to model the motion of a car-like robot on uneven terrain, we present a trajectory optimization framework for car-like robots on uneven terrain that can consider terrain curvature and dynamics of the robot. Real-world experiments and simulation benchmark comparisons validate the efficiency and quality of our method. In fact, more factors can also be considered in this framework in the form of constraints, such as obstacles and user-defined risks.

However, there is still space for improvement in our method. We did not consider the suspension system of the robot, the interaction of the robot's tires with the ground and the possible tire skidding. Also, in more complex scenarios (e.g., weedy land, deserts, muddy land), it is challenging to construct accurate dynamics models of the robot in contact with the terrain, which may require a combination of other tools, such as neural networks, stochastic processes, etc.

In the future, we will extend this algorithm to multi-layer environments and more difficult field environments. Besides, to further exploit the advantages of the efficiency of our method, local re-planning will be considered and adapted to the dynamic environments.

References

[1] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[2] Rösmann, Christoph and Hoffmann, Frank and Bertram, Torsten, "Kinodynamic trajectory optimization and control for car-like robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5681–5686.

[3] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[4] 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.

[5] T. Overbye and S. Saripalli, "Fast local planning and mapping in unknown off-road terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5912–5918.

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

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

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

[9] 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," *arXiv preprint arXiv:2302.03323*, 2023.

[10] 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.

[11] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 5054–5061.

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

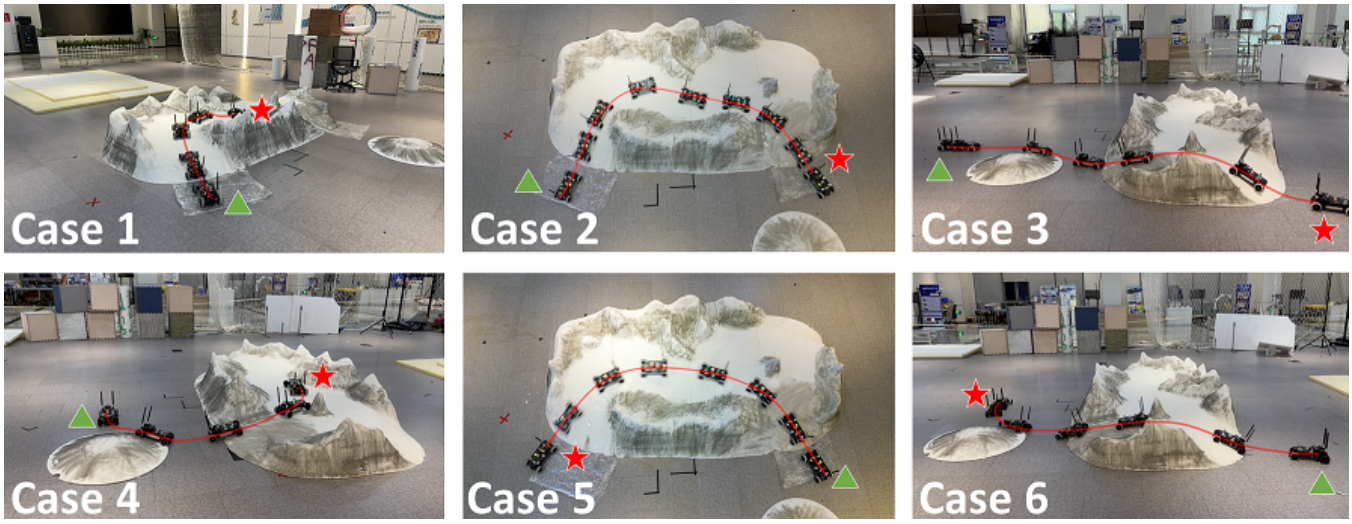


Fig. 6: The motion diagram of car-like robot in the real-world experiments, where the red curves are the planned trajectories. The green triangles indicate the starting points and the red pentagrams indicate the end points.

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

[14] N. Seegmiller, J. Gassaway, E. Johnson, and J. Towler, "The maverick planner: An efficient hierarchical planner for autonomous vehicles in unstructured environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2018–2023.

[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] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 835–842.

[17] J. Zhou, R. He, Y. Wang, S. Jiang, Z. Zhu, J. Hu, J. Miao, and Q. Luo, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 439–446, 2020.

[18] K. Kondak and G. Hommel, "Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 3. IEEE, 2001, pp. 2698–2703.

[19] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

[20] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 970–11 981, 2021.

[21] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A. Agha-mohammadi, "STEP: stochastic traversability evaluation and planning for risk-aware off-road navigation," in *Robotics: Science and Systems*. RSS Foundation, 2021, pp. 1–21.

[22] T. Guan, Z. He, R. Song, D. Manocha, and L. Zhang, "Tns: Terrain traversability mapping and navigation system for autonomous excavators," *Proceedings of Robotics: Science and Systems*, 2022.

[23] 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.

[24] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 2997–3004.

[25] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[26] K. Zhang, Y. Yang, M. Fu, and M. Wang, "Traversability assessment and trajectory planning of unmanned ground vehicles with suspension systems on rough terrain," *Sensors*, vol. 19, no. 20, p. 4372, 2019.

[27] F. Atas, G. Cielniak, and L. Grimstad, "Elevation state-space: Surfel-based navigation in uneven environments for mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5715–5721.

[28] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

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

[30] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[31] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Visualization, 2002. VIS 2002.* IEEE, 2002, pp. 163–170.

[32] 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.

[33] J. Leslie, "On a differential structure for the group of diffeomorphisms," *Topology*, vol. 6, no. 2, pp. 263–271, 1967.

[34] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen, *et al.*, "Differential flatness-based trajectory planning for autonomous vehicles," *arXiv preprint arXiv:2208.13160*, 2022.

[35] R. T. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming," *SIAM Journal on Control*, vol. 12, no. 2, pp. 268–285, 1974.

[36] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

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

[38] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.