This is the Author's Accepted Manuscript version of the paper:
C. Shen, S. Yu, B. I. Epureanu, and T. Ersal, "An efficient global trajectory planner for highly dynamical nonholonomic autonomous vehicles on 3D terrains," IEEE Transactions on Robotics, doi:10.1109/TRO.2023.3344030.

An Efficient Global Trajectory Planner for Highly Dynamical Nonholonomic Autonomous Vehicles on 3D Terrains

Congkai Shen†, Siyuan Yu†, Bogdan I. Epureanu, and Tulga Ersal*

***Abstract*—A novel hierarchical global trajectory planner is presented to allow highly dynamical nonholonomic off-road autonomous vehicles to achieve high mobility on 3D terrains. On complex terrains with uneven topology, designing safe and feasible vehicle trajectories often demands an understanding of the vehicle's dynamical and nonholonomic constraints. Prior research, however, treats the global planning problem as a path planning problem without effectively accounting for topology or dynamical constraints. To address this gap, this paper presents a three-phase trajectory planning algorithm composed of an A*, a rapidly exploring random tree (RRT), and a local trajectory refining (LTR) phase to incorporate dynamical and nonholonomic constraints on uneven terrain. The algorithm is tested in scenarios with randomized terrain fields and obstacles to demonstrate the necessity for all three phases. The algorithm is shown to have lower cost, higher success rate, and higher computational efficiency compared to state-of-the-art methods. The algorithm is then tested by controlling a simulated MRZR vehicle on a 3D terrain along with a local controller, with comparisons to state-of-the-art algorithms. It is demonstrated that the new algorithm is capable of planning dynamically feasible trajectories with lower cost where the state-of-the-art algorithms fail to perform due to neglecting dynamical vehicle limitations.**

***Index Terms*—Off-road, navigation and control, motion planning, obstacle avoidance, sampling algorithm, trajectory planning, hierarchical planning architecture.**

I. Introduction

Scenarios such as military operations, mining, and planetary exploration require autonomous vehicles to have off-road navigation capabilities [1]. Such off-road conditions often suggest an unstructured environment where the flat ground assumption does not hold. Uneven terrain topology could pose serious hazards to vehicle mobility, especially for large, high-speed vehicles. Therefore, to realize their full potential in off-road scenarios, autonomous vehicles need global trajectory planners that efficiently account for the uneven terrain topology, as well as the nonholonomic and dynamical constraints of the vehicle. However, as the literature review below shows, such a planner does not yet exist. This paper aims to fill this gap.

†These authors contributed equally to this work.
C. Shen, S. Yu, B. I. Epureanu and T. Ersal are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI. (email: {cosh, johnysy, epureanu, tersal}@umich.edu
The authors acknowledge the technical and financial support of the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-19-2-0001 U.S. Army DEVCOM Ground Vehicle Systems Center (GVSC) Warren, MI.
DISTRIBUTION A. Approved for public release; distribution unlimited. OPSEC #: 7224
* Corresponding author

A. Background

The ground robotics literature offers various motion planning methods for 3D terrain navigation. With an emphasis on computational performance, graph-search based methods have been developed to provide paths for small robots with computational efficiency. In [2], an A* algorithm is used to find a path in irregular triangular meshes. To rapidly re-plan the path, a D*Lite algorithm was developed [3], which can plan from the end to the starting position to significantly shorten the replanning time. To generate fast and smooth paths using the graph search based technique, a field D* algorithm was implemented in [4] and validated with Mars Exploration Rovers [5]. State lattice based approaches have also demonstrated their real-time applicability in navigating complex 3D terrain environments via considerations for robot dynamics and effective search in discretized spaces [6]–[8]. By using a grid as the representation of path points, this kind of path planner has limited resolution. Therefore, its utility is also limited for highly dynamical vehicles, i.e., where nonholonomic constraints play a significant role in the vehicle behavior.

To explicitly address the limited resolution when planning on 3D terrains, researchers in [9] used an artificial potential field (APF) to plan paths for all-terrain wheeled vehicles and proved that this algorithm can generate near-optimal trajectories if a global search is given as an initial guess. However, it is difficult to take hard constraints into account. Thus, this method inevitably faces the problem of planning paths that traverse into infeasible areas.

Model predictive control (MPC) has been favored due to its ability to consider all kinds of constraints simultaneously. In on-road scenarios, MPC has been successfully applied in extreme driving conditions by planning trajectories instead of only paths [10], [11]. MPC has also been extended to off-road navigation considering deformable terrains [12], 3D topology [13], and vehicle stochastic traversability [14]. Moreover, there exists work combining multiple factors that affect vehicle-terrain interaction, and the controller achieves robustness by continuously learning the inverse kinodynamics of the off-road vehicle [15]. Though the MPC method demonstrates success in this domain of interest, it is typically used as a local planner and controller for short distances rather than a global planner.

Sampling based planning algorithms such as rapidly exploring random trees (RRT) can provide efficient global planning solutions under constraints. In [16], researchers relaxed con-

straints as a cost function to substantially accelerate the solving time by employing quadratic programming on the connection of two nodes. To mitigate confusion stemming from high costs and infeasible outcomes, sampling based methods were employed to compute a global path, incorporating the hard constraints of nonholonomic motion and terrain traversability [17], [18]. Utilizing reinforcement learning to capture the critical kinodynamic constraints, the Terrain-SBMP (sampling based motion planning) approach is created to plan paths for vehicular systems on uneven terrains [19]. To further improve the success rate of RRT based methods, a hierarchical motion planning framework is developed in [20] by biasing the RRT sampling with D*Lite path points. Built upon these works, a three-phase algorithm that considers the nonholonomic constraints of a wheeled vehicle on 3D topology was developed in [21]. However, for extreme cases of 3D topology, planning only a path is not sufficient, as the speed profile can significantly affect vehicle mobility, as well, and needs to be co-optimized together with the path.

To extend the RRT based frameworks to perform trajectory planning, nonholonomic constraints, and dynamical constraints must be considered. To that end, kinodynamic based motion planning has been applied with success to incorporate both constraints in trajectory planning. To improve the reachability analysis (i.e., nonholonomic constraint settings), RRT frameworks combined with a learned policy (RL-RRT) were implemented in [22]. To further consider key dynamical limits in trajectory planning, NoD-RRT utilized the neural network based cost function and nonlinear robot dynamics to plan trajectories [23]. However, these approaches neither have an efficient computational performance nor offer the capability to plan trajectories in 3D terrain.

To alleviate the sub-optimal results sampling-based planning algorithms often yield, researchers have devised a range of techniques for further optimizing the paths [24]. For example, the discontinuity in the original paths can be addressed by utilizing different curves [25]–[27]. To further account for the kinematic constraints, researchers have developed ways to add perturbations to the path points and formulated a gradient descent optimizer [17], [28]. However, these investigations primarily concentrate on path optimization within kinematic constraints and lack the dynamical constraints necessary to address the objective of navigating through highly challenging 3D topologies of this study.

B. Original Contributions

In light of the review above, an important gap is identified in the literature; namely, a global trajectory planner that efficiently accounts for nonholonomic and dynamical constraints on 3D terrains does not exist. Addressing this gap is critical to maximize vehicle mobility.

To address this gap, a novel three-phase algorithm is introduced in this paper inspired by the advantages observed in hierarchical motion planning frameworks. Specifically, the three-phase algorithm comprises the A*, RRT, and local trajectory refining (LTR) phases. A* guides the RRT to sample collision-free trajectories that reach the goal position. Then, a novel formulation of RRT that incorporates nonholonomic and dynamical constraints is developed to plan feasible trajectories. The LTR phase then refines the trajectories to reduce the trajectory cost. The necessity of each phase is analyzed by comparing the planning cost under two kinds of obstacle fields with random terrains. The new planner is then tested in simulation in a hill climbing scenario and a complex field together with a local tracker to demonstrate improved mobility. Finally, the proposed method is benchmarked against state-of-the-art algorithms to demonstrate its lower trajectory cost.

Thus, the original contribution of this work is a novel hierarchical global trajectory planner that considers nonholonomic and dynamical constraints for off-road wheeled vehicles and plans trajectories effectively and efficiently with a high success rate on 3D terrains. The significance of this contribution is that it allows for navigating highly dynamical off-road autonomous vehicles with high mobility on 3D terrains.

A preliminary version of this work was presented at a conference [21], considering only the path planning aspect and with a coarse estimation of nonholonomic constraints. Compared to [21], this paper newly presents: 1) a global trajectory planner instead of only a path planner; 2) improved consideration of kinematic constraints of off-road vehicles; 3) incorporation of dynamical constraints on 3D topology; 4) the development of local trajectory refining for improving trajectory planning cost; and 5) the simulation and evaluation of the newly proposed global trajectory planner in high fidelity simulations against the state-of-the-art formulations.

C. Organization

The rest of the paper is organized as follows. Sec. II describes the nonholonomic and dynamical constraints as well as the neural network based cost function used in the trajectory planner. Sec. III details the topology-aware RRT algorithm. Sec. IV explains the methodologies of the first phase (A*) and the third phase (local trajectory refining) and how the three phases are connected. Together, Sec. II through Sec. VI-A describe the methodology, and their relationship is illustrated in Fig. 1. The results and discussions are presented in Sec. V and VI, where Sec. V shows the computational efficiency and trajectory cost of the planned trajectories, and Sec. VI describes the local controller that is used alongside the trajectory planner and utilizes a high fidelity vehicle simulation to evaluate the utility of the algorithm. Finally, Sec. VII concludes the study.

II. Neural Network Cost Function & Constraints

This section illustrates the core of the topology-aware RRT to efficiently evaluate cost and constraints as shown in Fig. 1. The cost function and nonholonomic and dynamical constraints used in RRT are introduced, where the costs are described in Sec. II-B and the constraints are demonstrated in Sec. II-C and II-D. Both cost and kinematic feasibility are determined through an optimal control problem (OCP), which is presented in Sec. II-A. The OCP section provides training and fitting data for Sec. II-C and II-B.

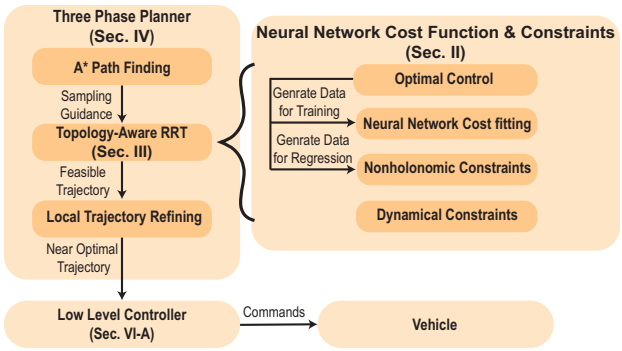


Fig. 1. Schematic of the proposed approach.

A. Optimal Control Problem Setting

The OCP is formulated such that it combines the tracking and planning problems, allowing the consideration of vehicle states ξ and control inputs ζ in the cost function. The OCP is given in a general form as follows:

$$\underset{\xi,\ \zeta,\ t_f}{\text{minimize}} \quad J = \mathcal{T}(t_f) + \int_0^{t_f} \mathcal{I}[\xi(t), \zeta(t)]dt \tag{1}$$

$$\text{subject to} \quad \dot{\xi}(t) = \mathcal{V}[\xi(t), \zeta(t)] \tag{2}$$

$$\xi_{\min} \leq \xi(t) \leq \xi_{\max} \tag{3}$$

$$\zeta_{\min} \leq \zeta(t) \leq \zeta_{\max} \tag{4}$$

$$\mathcal{F}[\xi(t_f)] \leq 0 \tag{5}$$

where J is the total cost function, $\mathcal{T}(\bullet)$ and $\mathcal{I}(\bullet)$ are nonlinear cost functions for total time to goal, states and controls respectively. Eq. (1) is also subject to the constraints in Eq. (2) to Eq. (5). Eq. (2) is the hard constraint for the vehicle dynamics detailed in Sec. II-A1. The state and control bounds are set in Eq. (3) and Eq. (4), respectively. The OCP does not explicitly take obstacle avoidance into account, because obstacles are considered in Sec. III. Eq. (5) is the hard terminal constraint for the vehicle to reach the goal. The expansions of these expressions are detailed below.

1) Vehicle Model: A kinematic bicycle vehicle model is used to predict the vehicle behavior on the local plane, because it is proven to capture the key vehicle nonholonomic constraints in a computationally efficient way [29].

The state and control vectors are defined as follows:

$$\xi := \begin{bmatrix} x \\ y \\ \psi \\ u \end{bmatrix} = \begin{bmatrix} \text{x position of center of mass} \\ \text{y position of center of mass} \\ \text{yaw angle} \\ \text{longitudinal speed} \end{bmatrix} \tag{6}$$

$$\zeta := \begin{bmatrix} a_x \\ \delta_f \end{bmatrix} = \begin{bmatrix} \text{longitudinal acceleration} \\ \text{front tire steering angle} \end{bmatrix} \tag{7}$$

The nonlinear state equation is defined as

$$\dot{\xi} = \begin{bmatrix} u(t)\cos(\psi(t) + \beta(\delta_f)) \\ u(t)\sin(\psi(t) + \beta(\delta_f)) \\ u(t)\frac{\cos(\beta(\delta_f))\tan(\delta_f)}{l_a + l_b} \\ a_x \end{bmatrix} \tag{8}$$

$$\beta(\delta_f) = \tan^{-1}\left(\frac{l_a}{l_a + l_b}\tan(\delta_f)\right) \tag{9}$$

The parameters l_a and l_b are the distances from the front and rear axles to the vehicle center of mass.

2) Cost Function: The cost function J is expressed as follows:

$$J = \quad w_t t_f + \int_{t_0}^{t_f} (w_{\delta_f}\delta_f^2 + w_{a_x}a_x^2)dt + \int_{s_0}^{s_f} w_\kappa \kappa^2 dt \tag{10}$$

The first term minimizes the total time to goal. The second term regulates the control effort. The last term penalizes the curvature κ along the path. The parameters w_t, w_{δ_f}, w_{a_x}, and w_κ are the corresponding weights for each term. As a two-point boundary problem, only stage costs are considered. Adding terminal costs does not affect the result of the optimization. Hence, the terminal costs are ignored in this part. However, when concatenating optimal solutions in the RRT algorithm, the terminal costs become important and are discussed in Sec III.

3) State and Control Bounds: The state and control bounds are set based on safety concerns and mechanical limits of actuators. The state bounds are expanded as:

$$\begin{bmatrix} x_{\min} \\ y_{\min} \\ \psi_{\min} \\ u_{\min} \end{bmatrix} \leq \begin{bmatrix} x \\ y \\ \psi \\ u \end{bmatrix} \leq \begin{bmatrix} x_{\max} \\ y_{\max} \\ \psi_{\max} \\ u_{\max} \end{bmatrix} \tag{11}$$

The control bounds are set similarly as:

$$\begin{bmatrix} a_{x_{\min}} \\ \delta_{f_{\min}} \end{bmatrix} \leq \begin{bmatrix} a_x \\ \delta_f \end{bmatrix} \leq \begin{bmatrix} a_{x_{\max}} \\ \delta_{f_{\max}} \end{bmatrix} \tag{12}$$

4) Terminal State Constraints: Constraints are added on the terminal state to allow the vehicle to reach the desired terminal state. They are expressed as:

$$\begin{bmatrix} x_f - \epsilon_x \\ y_f - \epsilon_y \\ \psi_f - \epsilon_\psi \\ u_f - \epsilon_u \end{bmatrix} \leq \begin{bmatrix} x_f \\ y_f \\ \psi_f \\ u_f \end{bmatrix} \leq \begin{bmatrix} x_f + \epsilon_x \\ y_f + \epsilon_y \\ \psi_f + \epsilon_\psi \\ u_f + \epsilon_u \end{bmatrix} \tag{13}$$

where $\bullet_f$ denotes the terminal state and $\epsilon_\bullet$ is the allowed tolerance for the corresponding state.

5) Solution Strategy: The OCP, which is presented in continuous time in Eq. (1), is solved by transcribing it into a nonlinear programming through NLOptControl [30] and solving using Ipopt [31]. For transcription, 10 collocation points are used with the backward Euler integration scheme.

B. Neural Network Surrogate for Optimal Cost

Though the OCP described in Sec. II-A can be solved rapidly, it is still computationally expensive. Therefore, a neural network surrogate model is introduced to approximate the optimal cost resulting from solving the OCP. The inputs of the neural network are the terminal states of the vehicle used in the OCP. To create the data set for training and validating the neural network, a Latin Hypercube Sampling (LHS) approach that maximizes the distance between each state is used, with the ranges reported in Table I. 2,000,000 samples are generated using LHS and are fed into the OCP. Among the whole set, the ones that are feasible are collected in the data set and are then used to train the neural network.

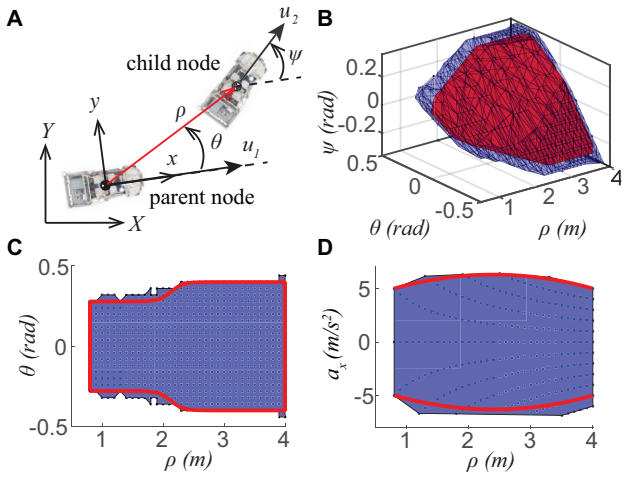


Fig. 2. Figures for nonholonomic constraints. (**A**) The state configuration of parent and child nodes. The global frame is denoted as the $X-Y$ coordinate frame. The $x-y$ coordinate frame represents the vehicle frame, which is based on the pose of the parent node. (**B**) The reachable set of $\theta-\rho-\psi$ with blue volume as the original and the red volume as the regressed reachable set. (**C**) The reachable set is projected onto the $\theta-\rho$ plane. The blue dots are the reachable data set, and the blue area is the region that contains those data. The red solid lines are the regressed boundaries. (**D**) The reachable set for the average acceleration a_x versus the distance ρ.

After the data set is generated through solving the OCP, it is randomly partitioned such that 70% is used for training, 15% is used for validation, and 15% is used for testing. The neural network is trained using Bayesian regularization backpropagation with hyperbolic sigmoid transfer functions. 100 neural networks that have 2 hidden layers, with 16 neurons in the first layer and 12 neurons in the second, are trained and the best one is selected to be used as the cost function for RRT. The neural network is further validated by a new data set generated through a uniform sampling of the kinematic feasibility described in Sec. II-C. The mean accuracy is defined to evaluate the performance of the neural network surrogate model as follows:

$$\frac{1}{N_n}\sum_{i=1}^{i=N_n}\left(1-\frac{|s_i-\tilde{s}_i|}{s_i}\right) \tag{14}$$

where N_n is the total number of new data for further validation, which is set as 3000. s_i is the true cost of the i^{th} data, and $\tilde{s}_i$ is the corresponding cost evaluated by the neural network. The mean accuracy is 96.6%, which fulfills the purpose of being a surrogate evaluation of the optimal cost. By adopting the neural network based cost evaluation, the computational cost is lowered by approximately 10^5 times compared to solving the OCP directly.

C. Nonholonomic Constraints

The trajectory is planned on the local plane originating on the posture of the parent node; i.e., the vehicle motion between the parent and the child node is simplified as a 2D motion. By applying the necessary coordinate transformations, the posture of the parent node is placed at the origin with the heading pointing to the positive x direction as shown in Fig. 2-A. As the position and yaw angle of the parent node become zeros through this coordinate transformation, only five parameters are required to have the full knowledge of the states of the parent and child nodes, which are the speed of the parent node u_1, the distance between the parent and child node ρ, the angle for locating the position of the child node θ, the yaw angle difference between the child and parent node ψ, and the speed of the child node u_2.

TABLE I
UNIFORM SAMPLING SETTINGS

Parameters	Range	Step
ρ	[0.8, 4] (m)	0.1 (m)
θ	[-1.0, 1.0] (rad)	0.04 (rad)
ψ	[-0.35, 0.35] (rad)	0.05 (rad)
u_1	[1, 9] (m/s)	2 (m/s)
u_2-u_1	[1, 9] (m/s)	2 (m/s)

The nonholonomic constraints can be accounted for by solving the optimal control problem at each connection. Even though this solution is reliable, it requires a large computation time. Another way is to create a lookup table for nonholonomic constraints, moving most of the computational burden offline. However, a large lookup table also requires large computational resources for searching, and the states in the lookup table are discrete. To overcome these problems, the neural network is a promising method to replace the lookup table. However, as a black-box method, false positives are hard to exclude, endangering vehicle safety. Based on the above considerations, the chosen method in this study is to use an analytical expression to create a regression model for the conservative estimation of nonholonomic constraints.

To create a regression model for the nonholonomic constraints, i.e., the reachable set, the ground truth is required. The ground truth is the data that is uniformly sampled in five parameters following the range given in Table I and calculated through Sec. II-A to determine feasibility. Because the vehicle model is the kinematic bicycle model, and only the steering angle is bounded, the path's shape is independent of speed. Thus, the reachable set composed of five parameters is separated into two parts: one is for the path's shape as a function of ρ, θ, and ψ, and the other is for the longitudinal acceleration boundary as a function of u_1, u_2, and ρ.

The shape of the reachable set as a function of ρ, θ, and ψ is shown in Fig. 2-B as the blue region. To regress this 3D shape, a projection onto the θ-ρ plane is first studied, as shown in Fig. 2-C. The blue dots are the points that are reachable, and the 2D blue region is the area that covers all feasible data. Based on the shape of the upper and bottom boundaries, the sigmoid function is used for regression as:

$$\theta_{\text{bound}} = \frac{a_1}{1+e^{a_2(\rho-a_3)}}+a_4 \tag{15}$$

where θ_{bound} is the upper or bottom boundary. The parameters a_1, a_2, a_3, and a_4 are optimized for the maximum area inside the 2D blue region. Once the projection is regressed, as shown with the solid red lines in Fig. 2-C, the upper and bottom surfaces of the 3D reachable set are regressed as third-order

polynomial functions of θ and ρ; i.e.,

$$\begin{aligned}\psi_{\text{bound}} =& b_1\rho^3 + b_2\rho^2\theta + b_3\rho\theta^2 + b_4\theta^3 + b_5\rho^2 + \\ & b_6\rho\theta + b_7\theta^2 + b_8\rho + b_9\theta + b_{10}\end{aligned} \tag{16}$$

where ψ_{bound} is the upper or bottom surface, and $b_1, b_2, ..., b_{10}$ are parameters optimized to have the maximum volume without exceeding the 3D blue region. The final regression on the reachable set is shown in Fig. 2-B as the red 3D region.

The existence of the reachable set for u_1, u_2, and ρ is due to the constraints on the acceleration. As a result, it is natural to study the relationship between the acceleration a_x and the direct distance ρ. The acceleration is approximated as the average acceleration by the following expression:

$$a_x = \frac{u_2^2 - u_1^2}{2\rho} \tag{17}$$

The shape of the reachable set for a_x and ρ is shown in Fig. 2-D as the blue region. Ideally, the upper and bottom boundaries should be straight lines. However, the direct distance ρ is not equal to the traveling distance, and thus the approximation of average acceleration is not exact. The bounds for the acceleration at each time do not imply that the average acceleration will follow the same constraints. To regress the relationship between the average acceleration and the direct distance, the parabolic function is used for the upper and bottom boundary as follows:

$$a_{x,\text{bound}} = c_1(\rho - c_2)^2 + c_3 \tag{18}$$

where $a_{x,\text{bound}}$ is the upper or bottom boundary shown in Fig. 2-D, and c_1, c_2, and c_3 are parameters optimized to cover the maximum area inside the blue region. The resulting regressed boundaries are shown as the red solid lines in Fig. 2-D.

D. Dynamical Constraints

The dynamical constraints are used to check whether the given terrain condition can provide enough force to allow the vehicle to keep its motion on the 3D terrain. For this evaluation, the vehicle is simplified as a point mass. The free-body diagrams from the top and side views are shown in Fig. 3.

Given the shape of the path, the curvature in the 3D path is calculated by taking the derivative of the heading with respect to the traveling distance. With the information on terrain shape, the normal and tangential vectors are known, which form the positive x and y axes of the vehicle frame. By projecting the curvature onto those two vectors, the normal and geodesic curvatures, named κ_n and κ_g, are calculated. The normal force N and the maximum allowable friction force f_{max} are calculated by the following equations:

$$F_{c,n} = N - W_z \tag{19}$$

$$N = W_z + F_{c,n} = W_z + mu^2\kappa_n \tag{20}$$

$$f_{\text{max}} = \mu N \tag{21}$$

where W_z is the weight projected in the normal direction, $F_{c,n}$ is the required centripetal force to keep the vehicle's motion, m is the mass of the vehicle, and u is the speed of the vehicle and also the design variable. The coefficient

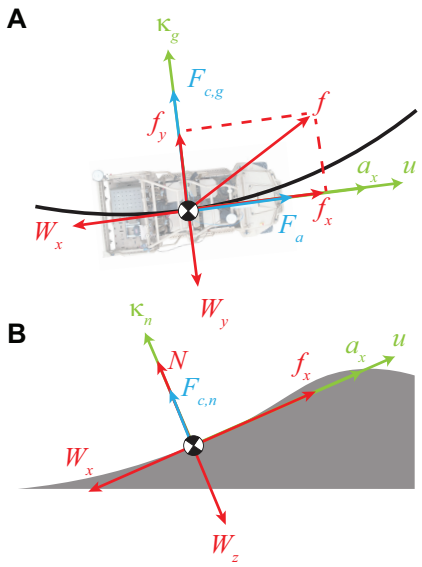


Fig. 3. The free-body diagrams of the vehicle driving on terrain. (**A**) The free-body diagram from the top view (along the normal direction). The black solid line is the traveling path. (**B**) The free-body diagram from the side view (along the lateral direction). The gray-shaded area is the terrain.

of friction μ determines the total friction that the vehicle can utilize when accelerating, braking, or turning. Although knowing its exact value is difficult, recent studies demonstrate that this parameter can be estimated successfully through on-board sensors, off-board sensors, and data driven methods [32]–[34]. Thus, performing such estimations is beyond the scope of this study.

The first dynamical constraint ensures that the vehicle does not lift off the terrain; i.e., $N \geq 0$. Thus, the inequality for u is written as follows:

$$u \leq \begin{cases} \infty, & \text{if } \kappa_n >= 0 \\ \sqrt{\frac{-W_z}{m\kappa_n}}, & \text{otherwise} \end{cases} \tag{22}$$

Based on the above inequality, u is not constrained if the terrain is concave, and is bounded by the maximum speed if the terrain is convex.

The second dynamical constraint checks whether the friction force is large enough to let the vehicle drive along the curved path with the given geodesic curvature κ_g. Suppose the friction force is large enough; then, the forces along the x and y axes are summed to allow the accelerated curvilinear motion shown in Fig. 3-A as follows:

$$\Sigma F_x = f_x - W_x = F_a = ma_x \tag{23}$$

$$\Sigma F_y = f_y - W_y = F_{c,g} = mu^2\kappa_g \tag{24}$$

where f_x and f_y are the longitudinal and lateral portions of friction force, respectively. The total friction force is calculated as follows:

$$f = \sqrt{f_x^2 + f_y^2} \tag{25}$$

By combing Eq. (23)-(25), the second dynamical constraint is

formed by bounding the total friction force by $f_{\max}$ as follows:

$$\begin{aligned} f &\le f_{\max} \\ \sqrt{f_x^2 + f_y^2} &\le \mu N \\ \sqrt{(W_x + ma_x)^2 + (W_y + mu^2\kappa_g)^2} &\le \mu(W_z + mu^2\kappa_n) \end{aligned} \tag{26}$$

The third dynamical constraint checks whether the torque provided by the engine is large enough. The corresponding inequalities are:

$$\begin{aligned} \frac{\tau_{\min}}{R_w} \le & f_x \le \frac{\tau_{\max}}{R_w} \\ \frac{\tau_{\min}}{R_w} \le & W_x + ma_x \le \frac{\tau_{\max}}{R_w} \end{aligned} \tag{27}$$

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum output torque provided by the vehicle engine, and R_w is the wheel radius.

In summary, three dynamical constraints are derived in Eq. (22), Eq. (26), and Eq. (27). They are used to ensure the feasibility of the circular motion corresponding to the normal and geodesic curvatures, respectively.

III. Topology-Aware Rapidly Exploring Random Tree (RRT)

Algorithm 1 RRT

Input: N_s: Number of sampling nodes, ξ_0: Vehicle starting states, ξ_f: Vehicle ending states, $\mathcal{O}$: Obstacle field, μ: Coefficient of friction, $\mathcal{B}$: Map bounding position, $\mathcal{P}$: Terrain point cloud, T_A: A* guidance point.
Output: TP Trajectory Plan.

Main Algorithm:

```
iteration = 0, t = 0, E = ∅
CreateKNNTerrain(P)
while iteration ≤ N_s and t ≤ T_t do:
    ξ_rand = SampleNode(T_A)
    ξ_nearest = FindNearestNode(E, ξ_rand)
    ξ_new = Steer(ξ_rand, ξ_nearest)
    if not Collision(ξ_new, ···) then
        CalculateCost(ξ_new, ξ_nearest)
        RegisterNode(ξ_new, E)
        iteration = iteration + 1
    end if
end while
return TP = ExtractMinimalCostTraj(E)
```

Algorithm 1 describes the proposed RRT formulation for wheeled vehicles that takes 3D terrain topology into account. The algorithm is implemented based on the standard RRT with several modifications made in the functions **CalculateCost**, **FindNearestNode**, **Steer**, and **Collision**. The terrain information is assumed to be gained in the form of point clouds to support a future implementation on a real vehicle. For computational efficiency, the point clouds are stored in the form of a kd-tree to speed up the running time of the K-nearest neighbor search during the planning [35].

In the **CalculateCost** function, the neural net function derived in Sec. II-B is used. The neural net function is derived based on the data generated by the optimal control problem. As mentioned in Sec. II-A, the cost definition only takes the stage cost into account. Even though the path between two nodes is smooth, the concatenated path may not be. To have a smooth speed profile and path shape in the final trajectory, three more terms are added to Eq. (10) as follows:

$$J_{\text{RRT}} = J + w_y(\rho\sin(\theta))^2 + w_\psi\psi^2 + w_u(u_2 - u_1)^2 \tag{28}$$

where J is the cost defined in Eq. (10), and w_y, w_ψ, and w_u are weights for the three additional terms. ρ, θ, ψ, u_1, and u_2 are states shown in Fig. 2-A, which are distance, angle for locating the position, yaw angle, speed of parent node, and the speed of child node. J_{RRT} is the cost defined for RRT planning and the cost used for Local Trajectory Refining (LTR).

In the **FindNearestNode** function, a ball tree data structure is used to speed up the searching procedure. The task of the **FindNearestNode** function is to find the tree node ξ_{nearest} in RRT that has the smallest cost connecting to the sampled tree node ξ_{rand}. For the sake of computational efficiency, the cost definition J_{RRT} is not used for the metric in the ball tree. Instead, an approximated form is used to mimic the neural net cost, and the metric for ball tree $\tilde{J}_{\text{BT}}$ is defined as follows:

$$\begin{aligned} \tilde{J}_{\text{BT}} &= \tilde{J} + w_y(\rho\sin(\theta))^2 + w_\psi\psi^2 + w_u(u_2 - u_1)^2 \\ \tilde{J} &= w_t\frac{2\rho}{u_1 + u_2} + w_{a_x}\frac{(u_2^2 - u_1^2)(u_2 - u_1)}{2\rho} \\ &\quad + w_\kappa\left(\frac{2\theta^2}{\rho} + \frac{2(\psi - \theta)^2}{\rho}\right) \end{aligned} \tag{29}$$

where the first term in $\tilde{J}$ approximates the penalty on traveling time in Eq. (10) by assuming that the motion between parent and child nodes is the uniformly accelerated motion in a straight line. The second term in $\tilde{J}$ approximates the penalty of control effort in acceleration. There is no way to approximate the control effort on steering angle; thus, it is neglected in $\tilde{J}$. This negligence does not significantly influence the value of cost for a small w_{δ_f}. The last term in $\tilde{J}$ approximates the penalty on the path curvature. Intuitively, $\frac{\psi^2}{\rho}$ should be used for approximation. However, it is possible that the vehicle makes an S-turn, making θ non-zero but ψ zero. In that case, the intuitive approximation method underestimates the cost of curvature. To take both θ and ψ into account, the assumption is made that during the first half of the path, the vehicle turns from 0 to θ with a constant curvature, and during the second half of the path, the vehicle turns from θ to ψ.

In the **Steer** function (Algorithm 2), four inputs are given, which are the randomized node through sampling strategy ξ_{rand}, the existing tree node in RRT ξ_{nearest}, the coefficient of friction μ and the topology information of the terrain $\mathcal{P}$. The task of the **Steer** function is to ensure that ξ_{rand} satisfies the nonholonomic constraints and the dynamical constraints derived in Sec. II-C and II-D. If ξ_{rand} violates the constraints, the **Steer** function moves ξ_{rand} inside the region where the constraints are satisfied.

The procedure follows the steps below:

1) Check whether the distance between ξ_{rand} and ξ_{nearest} is

Algorithm 2 Steer Function

Input: ξ_{rand}: Randomized node of vehicle states, ξ_{nearest}: Nearest node of vehicle states, μ: Coefficient of friction, $\mathcal{P}$: Terrain point cloud.
Output: ξ_{new} New node of vehicles states.

Main Algorithm:

ρ = **CalculateDist**(ξ_{rand}, ξ_{nearest})
θ = **GetPolarAngle**(ξ_{rand}, ξ_{nearest})
ψ = **GetYawAngle**(ξ_{rand}, ξ_{nearest})
if $\rho \notin [\rho_{\min}, \rho_{\max}]$ **then**
  $\rho = \rho_{\max}$
end if
$\theta_{\max}$, $\Delta\theta_{\min}$ = **GetPolarAngleBounds**(ρ)
if $\theta \notin [\theta_{\min}, \theta_{\max}]$ **then**
  θ = rand([-$\theta_{\min}$, $\theta_{\max}$])
end if
$\psi_{\min}$, $\psi_{\max}$ = **GetYawAngleBounds**(ρ, θ)
if $\psi \notin [\psi_{\min}, \psi_{\max}]$ **then**
  ψ = rand([$\psi_{\min}$, $\psi_{\max}$])
end if
ξ_{new} = **MoveInNode**(ξ_{nearest}, ξ_{rand}, ρ, θ, ψ)
SpeedCheck = **CheckChildSpeed**(ξ_{new}, ξ_{nearest}, μ, $\mathcal{P}$)
if SpeedCheck **then**
  ξ_{new} = **SetChildSpeed**(ξ_{new}, ξ_{nearest}, μ, $\mathcal{P}$)
end if
return ξ_{new}

larger than the predefined maximum distance $\rho_{\max}$ or smaller than the predefined minimum distance $\rho_{\min}$. If so, move ξ_{rand} to where the distance equals to $\rho_{\max}$. Using this setting, RRT is observed to have better success rate because the tree grows faster under the same setting of N_s.

2) Use the regressed ρ-θ reachable set shown in Fig. 2-B to find the maximum and minimum angle in the polar coordinate system. If the angle θ is outside the feasible region, then the value of θ is uniformly randomized between $\theta_{\min}$ and $\theta_{\max}$.
3) Knowing ρ and θ, the boundaries of yaw angle ψ are determined by the regressed model shown in Fig. 2-C. The modification on the yaw angle ψ follows a similar procedure as the previous step. Moving ρ, θ, and ψ into the feasible region generates a new node as ξ_{new}. Note that θ and ψ determine the pose of the vehicle and they are the critical factors restricting the nonholonomic constraints. By these constraints, the feasible operation region for the vehicle becomes relatively smaller compared to the total sampling space. Direct projection of sampled nodes back to feasible sets on θ and ψ would cause a severe winding phenomenon and result in more unexplored regions of the whole state space.
4) Once the path's shape is guaranteed to be feasible, the next step is to ensure the feasibility of the speed. The function **CheckChildSpeed** is used to check whether the original speed satisfies the constraints regressed in Fig. 2-D and the dynamical constraints in Eq. (22), (26), and (27). If not, the speed in ξ_{new} is uniformly sampled from the feasible region calculated based on the regressed reachable set and the dynamical constraints in **SetChildSpeed** function.

If any step fails, meaning any feasible region in ρ, θ, ψ, or speed has an empty set, the **Steer** function returns nothing, and RRT samples again.

Algorithm 3 Collision Function

Input: ξ_{new}: New node of vehicles states, ξ_{nearest}: Nearest node of vehicle states, $\mathcal{O}$: Obstacle field, μ: Coefficient of friction, $\mathcal{B}$: Map bounding position, $\mathcal{P}$: Terrain point cloud.
Output: flag: Collision check boolean.

Main Algorithm:

BoundaryCheck = **CheckWithinBoundary**(ξ_{new}, $\mathcal{B}$)
FeasibilityCheck = **CheckFeasibility**(ξ_{new}, ξ_{nearest})
SpeedCheck = **CheckChildSpeed**(ξ_{new}, ξ_{nearest}, μ)
ObsCheck = **CheckObsCollisionFree**(ξ_{new}, $\mathcal{O}$)
flag = **not** (BoundaryCheck **and** FeasibilityCheck **and** SpeedCheck **and** ObsCheck)
return flag

The function **Collision** (Algorithm 3) checks whether the newly generated tree node is within the boundaries of the whole map by using the function **CheckWithinBoundary**. It also checks whether the nonholonomic constraints are satisfied by using the function **CheckFeasibility**. The dynamic constraints are evaluated by the **CheckChildSpeed** function. The collision with obstacles is checked by the function **CheckObsCollisionFree**. The function **Collision** returns true if any of those four checks are false.

IV. Three Phase Global Planner (TPP)

The final three-phase global planner (TPP) comprises the three phases of A*, RRT, and LTR. A* provides the sampling guidance for RRT. RRT generates a suboptimal trajectory that obeys the dynamical and nonholonomic constraints as shown in Sec. III. Finally, the LTR refines the trajectory to reach lower cost. This hierarchical planner is expected to have a higher success rate, lower cost, and better computational efficiency in reasonably long scenarios.

The second phase of the algorithm is detailed in Sec. III. Thus, this section first describes the remaining two phases. In particular, the first and third phase of the algorithm are covered in Sec. IV-A and IV-B, respectively. Then, the hyperparameter tuning process is explained, and a planning example is given.

A. Path Finding Algorithm A*

The A* path-finding algorithm [36] is executed to find the shortest path on the 3D terrain. A map with dimensions of $l \times l$ is discretized into an $N_g \times N_g$ grid. The specific values used in this study are summarized in Table II. The grid configuration space is restricted to 8-connectivity and the distance metric is the 3D Euclidean distance as described in [21]. A sketch for the A* result is shown in Fig. 4-A, where the black disks are obstacles, the orange crosses are the points in the A*

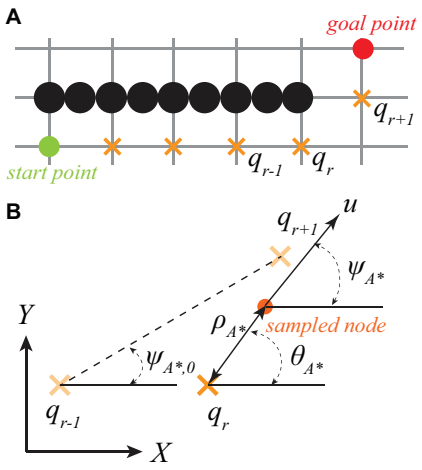


Fig. 4. The figures for A* guidance. (**A**) Illustration of the A* algorithm. The gray lines form the grid in A*. The black disks are obstacles. The orange crosses are the solution points provided by A*. (**B**) Randomization of the RRT node based on the A* solution. The representation of θ_{A^*}, ρ_{A^*}, and ψ_{A^*} is based on the $X-Y$ global frame.

TABLE II
SETTINGS IN A* GUIDANCE

Parameters	Value
l	120 m
N_g	50
$\rho_{\mathrm{A}^*,\min}$	0 m
$\rho_{\mathrm{A}^*,\max}$	15 m
$\theta_{\mathrm{A}^*,\min}$	$-\pi$ rad
$\theta_{\mathrm{A}^*,\max}$	π rad
$u_{\mathrm{A}^*,\min}$	1 m/s
$u_{\mathrm{A}^*,\min}$	9 m/s
σ_ψ	$\frac{\pi}{4}$ rad

path, and the green and red dots are the start and goal points, respectively.

The guidance of A* in the subsequent RRT phase is sequential instead of the uniform randomization used in the literature [20]. Suppose there are a total of N_{A^*} points in the A* path and N_{RRT} nodes for RRT sampling. The i^{th} sampled node is guided by the r^{th} point, q_r, in the A* path, with $r = \lfloor \frac{i}{N_{\mathrm{RRT}}} N_{\mathrm{A}^*} \rfloor + 1$. Once the reference point q_r is located, an RRT node is sampled based on this reference point as shown in Fig. 4-B. The full representation of the posture of a rigid body is composed of six states in the global frame, which are the X, Y, and Z positions and pitch, roll, and yaw angles. However, the Z position and pitch and roll angles are passively determined by the shape of the terrain, which corresponds to the vehicle's steady state. Thus, only three parameters are free to be randomized. The randomization obeys the following distributions:

$$\rho_{\mathrm{A}^*} \sim U(\rho_{\mathrm{A}^*,\min}, \rho_{\mathrm{A}^*,\max}) \tag{30}$$
$$\theta_{\mathrm{A}^*} \sim U(\theta_{\mathrm{A}^*,\min}, \theta_{\mathrm{A}^*,\max}) \tag{31}$$
$$u_x \sim U(u_{\mathrm{A}^*,\min}, u_{\mathrm{A}^*,\max}) \tag{32}$$
$$\psi_{\mathrm{A}^*} \sim N(\psi_{\mathrm{A}^*,0}, \sigma_\psi) \tag{33}$$

where ρ_{A^*} is the distance, and θ_{A^*} is the angle in the polar coordinate system. These two parameters are used to locate the X-Y position of the sampled node. u_x is the longitudinal speed, and ψ_{A^*} is the yaw angle. $\rho_{\mathrm{A}^*,\min}$, $\rho_{\mathrm{A}^*,\max}$, $\theta_{\mathrm{A}^*,\min}$, $\theta_{\mathrm{A}^*,\max}$, $u_{\mathrm{A}^*,\min}$, $u_{\mathrm{A}^*,\max}$, and σ_ψ are the hyperparameters, and their values are shown in Table II. $\psi_{\mathrm{A}^*,0}$ is the angle formed between the previous A* point q_{r-1} and the next A* point q_{r+1}, which is assumed to be the average yaw angle in the current A* point q_r. The value of standard deviation σ_ψ is set to ensure that most sampling of the yaw angle does not form an obtuse angle with the average yaw angle. Following the literature [20], the first three randomizations are uniform so that the vehicle can have an equal possibility of reaching any combination of location and speed within the hyperparameter boundaries. In contrast, due to the nonholonomic constraints, the heading cannot deviate too much from its original value. Hence, using a normal distribution for heading results in more feasible connections. In short, the points in the A* path are sequentially utilized so that RRT can sample around the A* points to increase the possibility of obtaining feasible nodes.

B. Local Trajectory Refining

As identified in [17] and [21], a local path planner can be used to effectively optimize the RRT path. This additional phase demonstrates high efficiency in correcting the winding phenomenon that is inherent in RRT. In [21], the local refinement is done by adding small perturbations to the path points to seek improvements to the cost while obeying the predefined constraints that are already imposed in the previous planning phases. This methodology was only used in path planning algorithms in prior literature. In this work, a similar methodology is applied in the trajectory planner in Algorithm 4.

The LTR begins with the TP, the trajectory that is optimized using RRT in Sec. III and strictly obeys the nonholonomic and dynamical constraints. For each iteration, the LTR algorithm optimizes the trajectory points in random order by shuffling the array indices. Then, a set of candidate nodes of vehicle states are created through perturbing the original trajectory point in the direction of location, speed and yaw angle with 3 options for each, resulting in a total of 27 nodes. The cost of the trajectory is evaluated 27 times for each candidate node and the costs are stored in the CostArray. A new function named **FindBestFeasibleStates** outputs the feasible points with the lowest cost. To do that, each candidate point along with the remaining part of the trajectory needs to pass the **Collision** function, with $\mathcal{B}$ used as a boundary check in **Collision** in Algorithm 3. If the candidate points cannot optimize the trajectory further, the original trajectory is returned.

C. Hyperparameter Tuning

There are three categories for the sampling strategies used during the RRT phase: 1) Sampling at the goal location. 2) Uniform sampling 3) Sampling with A* guidance. The first and second cases represent the general RRT techniques for accelerating trajectory generation and exploring the entire

Algorithm 4 Local Trajectory Refining (LTR)

Input: N_i: Number of maximum iterations, $[\epsilon_d, \epsilon_u, \epsilon_\psi]$: Adjustable values of vehicle position, speed and yaw angle, TP: Planned trajectory from RRT, $\mathcal{O}$: Obstacle field, μ: Coefficient of friction, $\mathcal{B}$: Map bounding position, $\mathcal{P}$: Terrain point cloud.
Output: RTP: Refined trajectory plan.

Main Algorithm:

```
SampleNumber = Length(TP)
OriginalIndexList = 2:SampleNumber
for iteration = 1, ··· N_i do
    ShuffledIndexList = Shuffle(OriginalIndexList)
    for index in ShuffledIndexList do
        Dircs = GetNode(ε_x, ε_u, ε_ψ, SampleNumber)
        ξ_index = TP[index], CostArray = ∅
        for dircs in Dircs do
            ξ_cand = GetNewNode(ξ_index, dircs)
            C = EvaluateCost(TP, ξ_cand)
            CostArray = CostArray ∪ C
        end for
        ξ_best = FindBestFeasibleStates(CostArray, ···)
        TP[index] = ξ_best
    end for
end for
RTP = TP
return RTP
```

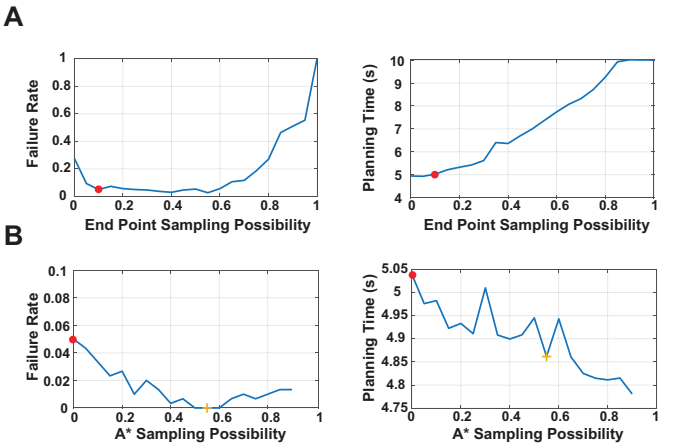


Fig. 5. Figures for hyperparameter tuning. (**A**) The tuning results for RRT without A* guidance. The best option for endpoint sampling is shown as the red dot. (**B**) The tuning results for RRT with A* guidance. In this case, the endpoint sampling probability is set to its optimal value of 0.1. The probability of A* sampling is gradually increased from 0 to 0.9. The red dot is the optimal setting for RRT without A* guidance and the yellow plus sign is the optimal setting for RRT with A* guidance.

state space. The third case is a new strategy introduced in this work. The performance of RRT is influenced by the probability of each case. The hyperparameter-tuning of these three possibilities is conducted to avoid an unfair weakening of the benchmark. The possibilities of the three cases are denoted as p_1, p_2, and p_3, respectively, with $\sum_{i=1}^{3} p_i = 1$.

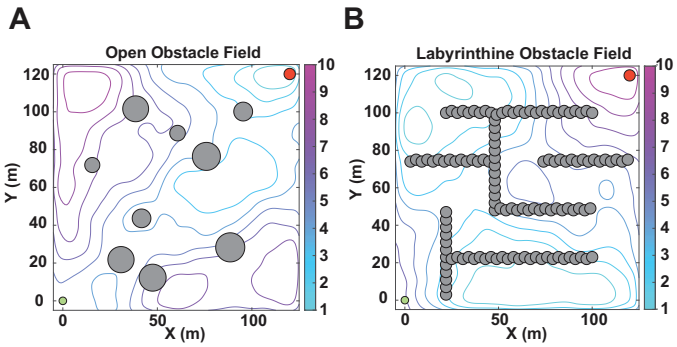


Fig. 6. Figures for randomized scenarios. (**A**) The scenario of an open obstacle field. The gray solid circles are obstacles. The terrain shape is shown as the contour plot. The greed and red circles are the start and goal points, respectively. (**B**) The scenario of a labyrinthine obstacle field.

TABLE III
TERRAIN RANDOMIZATION SETTINGS

Parameters	Values
roughness factor	50
width	120 m
length	120 m
height	10 m
maximum slope	0.289 rad
first filter radius	1 px
second filter radius	2 px

Terrain topology and obstacle field layout play significant roles in characterizing the mobility challenges in 3D terrain navigation. Therefore, scenarios are generated for tuning and testing by randomizing these characteristics.

The terrain topology is generated using midpoint displacement fractals, which can produce realistic terrain heightmaps [37]. Terrain generation is performed in Matlab R2022a, where the terrain shape is defined by roughness factor, width, length, height, and maximum slope. The specific settings used are reported in Table III. The roughness factor is used to define the standard deviation of Gaussian randomization in the process of midpoint displacement fractals. Once the randomization is finished, a Gaussian smoothing filter is applied twice with different filter radii as shown in Table III.

As for the obstacle field, two categories are considered. The first category of obstacle field is an open obstacle field. It features randomly sized obstacles that are dispersed randomly across the terrain as illustrated in Fig. 6-A. This category represents areas with scattered rocks and plants. The second category of obstacle field is a randomly generated labyrinthine obstacle field. It represents a structured obstacle field as illustrated in Fig. 6-B.

The tuning for the benchmark "RRT without A* guidance" only involves p_1. The probability p_1 is exhaustively searched from 0 to 1 with 0.05 increments. For each value of p_1, the planner is tested in 300 scenarios, with 150 labyrinthine obstacle fields and 150 open obstacle fields, each one with a randomized terrain field. The failure rates of the benchmark algorithm and the planning times are shown in Fig. 5-A. The optimal selection for p_1 is based on the prioritization of the

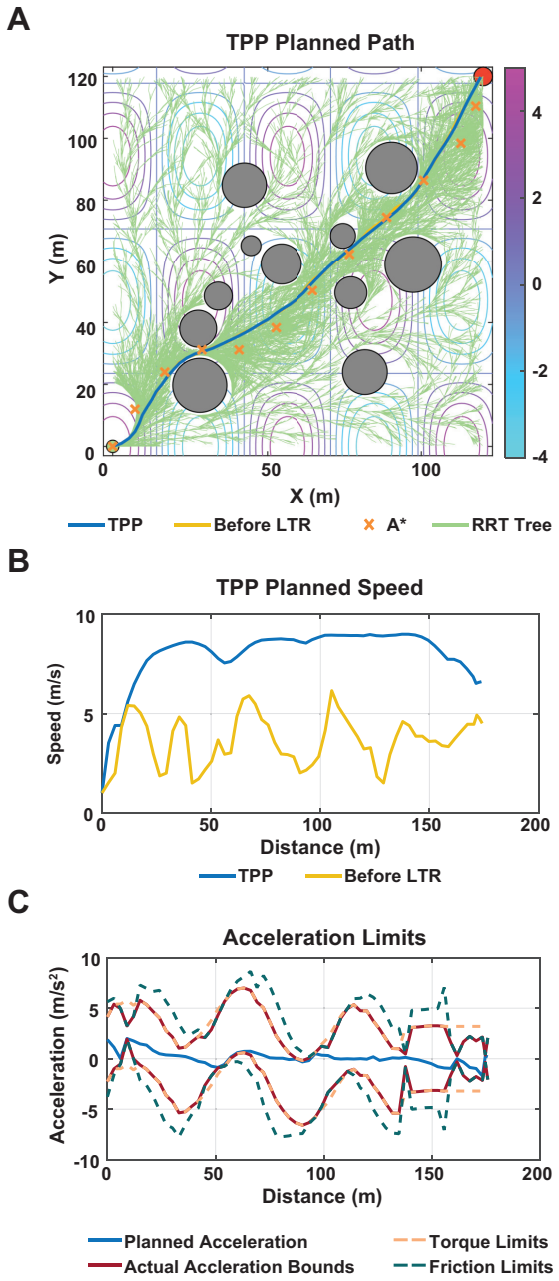


Fig. 7. Example planned trajectory by TPP: (A) The planned path (blue line) from TPP. Green lines are RRT branches, yellow line is the planned path directly from RRT and orange crosses are the A* points for guidance. (B) The speed profile before and after LTR. (C) Illustration that the planned trajectory strictly obeys the dynamical constraints.

minimum failing rate over the minimum planning time. Based on this prioritization, $p_1 = 0.1$ is selected as the best choice in the benchmark.

For a fair comparison, p_1 is kept at 0.1 in the proposed TPP algorithm, and an additional exhaustive search is conducted for p_3 with the same prioritization. The results are shown in Fig. 5-B, and the optimal selection for p_3 is found as 0.55.

D. Planning Example

After hyperparameters are tuned, a typical example demonstrating the planning process in an open obstacle field is shown

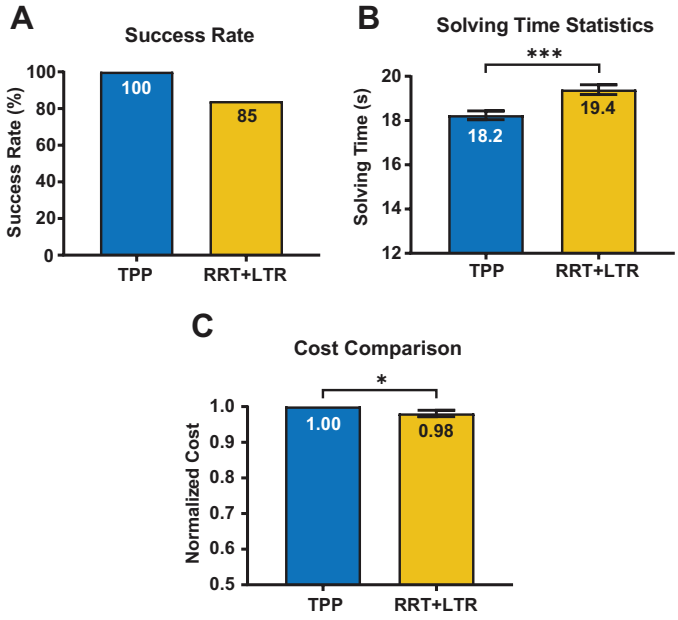


Fig. 8. The statistics for the A* guidance test. (**A**) Success rate. (**B**) Solve time. (**C**) Normalized cost.

in Fig. 7. The terrain field here is set as $Z = 5\cos(\frac{X}{9})\cos(\frac{Y}{15})$. The planned path is plotted in Fig. 7-A along with the candidate tree branches. By design, TPP tends to sample more points around the A* guiding point as seen from the darker green color of those regions. Based on the hyperparameters chosen in Sec. IV-C, there is still a 35% chance that the algorithm uses a uniform random sampling strategy so that the whole operation space is explored. In Fig. 7-B, a comparison of the speed profile is made for the inclusion of LTR. Initially, the speed is low and jittery by the end of the second phase, whereas it is higher and smooth after the LTR phase. As discussed in Sec. IV-B, the algorithm still follows the constraints that are prescribed in RRT. This is demonstrated in Fig. 7-C, where the planned acceleration obeys the dynamical constraints posed by torque and friction limits.

A more in depth analysis of the planning performance is given in Sec. V.

V. Planning Results

The section includes two parts, namely, Sec. V-A and V-B, which demonstrate the necessity of each phase. In all tests, statistical analysis is done through ANOVA with the significance threshold of 0.05. For multiple comparisons, the Tukey posthoc test is adopted for pairwise comparisons. In the bar charts representing the results of these tests, the level of significance is depicted by the number of asterisks; i.e., $*$ indicates $p < 0.05$, $**$ indicates $p < 0.01$, $***$ indicates $p < 0.001$, and $****$ indicates $p < 0.0001$. The results are obtained on a I7-12700K desktop with 32 GB memory.

A. A* Guidance Test

To test the effectiveness of A* guidance, two algorithms are compared in the randomized labyrinthine obstacle field:

(1) the proposed algorithm TPP (i.e., A*+RRT+LTR), and (2) RRT+LTR as the benchmark. The reason for using the labyrinthine obstacle field is to emphasize the advantage that the first phase can provide. The open obstacle field tends to conceal algorithm benefits, as the solution is easy to find without guidance.

Each algorithm is run once in 100 randomized scenarios. The results of the success rate and the solving time statistics are shown in Fig. 8-A and B. The success rate of the proposed algorithm is 100%, whereas the benchmark's success rate is 15% lower. Moreover, with the guidance of A*, the solving time decreases by around 1.2 s and the difference is statistically significant.

The normalized trajectory costs of the two algorithms are also compared to study the effect of biased sampling as shown in Fig. 8-C. The cost values of TPP are used as the baseline for normalization. The biased sampling introduces around 2% cost increase for the proposed algorithm when compared to RRT+LTR, which is acceptable compared to the improvements in success rate and solving time.

These results show the benefits of A* guidance as the increased success rate and decreased solving time. However, the increased normalized cost of the proposed algorithm implies that the biased sampling sacrifices the trajectory cost of RRT. Nevertheless, this shortcoming is deemed minor and thus acceptable.

B. Local Trajectory Refining Test

To test the effectiveness and efficiency of local trajectory refining (LTR), four algorithms are compared in the open obstacle field. All four algorithms are guided by A*. The benchmarks are A*+RRT, A*+RRT*, and A*+RRT*+LTR, and the target algorithm is the proposed framework TPP. Each algorithm is run once in 100 randomized scenarios.

The results of mean normalized cost are shown in Fig. 9-A. The cost values of A*+RRT are chosen as the baseline for normalization. A*+RRT* has a lower cost than A*+RRT by 15% with statistically significant difference. However, after applying the LTR, the costs dropped to around 0.35 without any statistically significant difference between them.

The mean solving times are also compared as shown in Fig. 9-B. The mean solving time of A*+RRT* is more than three times larger than that of A*+RRT, and is 50% larger than that of TPP. The mean solving time of A*+RRT*+LTR is the largest.

In this test, all algorithms succeed on all the testing scenarios; i.e., the success rate is 100% for all algorithms. Therefore, a comparison plot is not provided.

The results show that employing RRT* can provide trajectories with lower cost than using RRT alone, but RRT* still falls short of having a higher trajectory cost than that of TPP. Additionally, using RRT* unavoidably increases the computational burden of the algorithm, making it slower than the TPP. In the TPP framework, using RRT* instead of RRT as the second phase does not decrease the trajectory cost, but approximately doubles the solving time. In this regard, LTR is found to be a more efficient way to generate a lower cost trajectory than RRT*, and TPP is found to be the best combination considering the computational performance and trajectory cost.

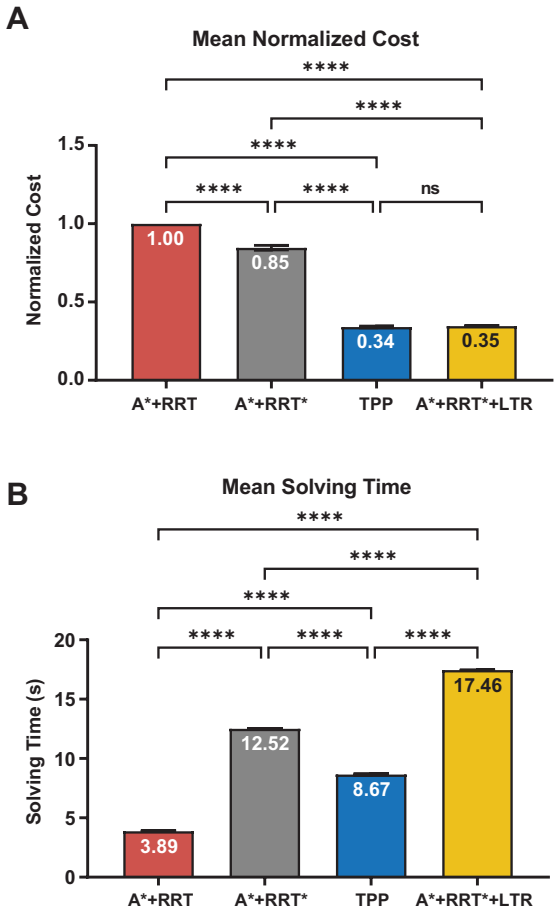


Fig. 9. The statistics for the local trajectory refining test. (**A**) Comparison of normalized cost. (**B**) Comparison of solving time.

VI. Tracking Results

This simulation validates the proposed global planner's capability of planning a dynamically safe trajectory. To do that, a local controller is developed to track the planned trajectories and is described in Sec. VI-A.

The multi-body simulation engine Project Chrono is utilized as the simulator for validation. The vehicle model represents an MRZR 4 powered by a gasoline engine [38]. During the simulation, the global trajectory is planned once at the onset and is then tracked by the local controllers to follow the planned path and speed. The benchmark is the path planning framework presented in [21]. In the benchmark scenarios, because the benchmark path planner does not plan the speed, two speed settings, namely 5 m/s and 9 m/s, are used for tracking, referred as the low-speed and high-speed modes.

The simulation study is divided into three cases. In the first case, two challenging slopes are designed, testing the importance of speed planning. In the second case, a challenging hilly terrain together with a symmetric obstacle field is designed to validate the capability of making complex coordinations between steering and speed in off-road navigation. In the third

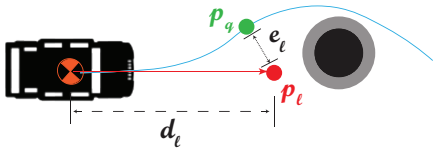


Fig. 10. Diagram for look-ahead PID controller. The blue line is the planned path. The red point p_l is the look-ahead point. The green point p_q is the queried point, which is closest to the look-ahead point on the planned path.

case, the algorithms are tested on 100 randomized labyrinthine obstacle fields, validating the correspondence between the planned cost and the actual cost, and also evaluating the performance of the algorithms through the cost definition in Sec. II-B. In all cases, the coefficient of friction of the terrain field μ is set to 0.6 to mimic the friction forces in sandy loam. These three cases together demonstrate the importance of considering nonholonomic and dynamical constraints for trajectory planning.

A. Local Controller

A look-ahead proportional-integral-derivative (PID) controller is used to steer the vehicle along the planned paths. The formulation of the controller is as follows:

$$\delta_{\text{act}}(t) = \begin{cases} \delta_{\text{PID}}(t), & \text{if } |\delta_{\text{PID}}(t)| < \delta_{\text{act,max}} \\ \delta_{f,\text{max}}\ sign(\delta_{\text{PID}}(t)), & \text{otherwise} \end{cases} \tag{34}$$

$$\delta_{\text{PID}}(t) = K_{p,l} e_l(t) + K_{i,l} \int e_l(t) + K_{d,l} \frac{de_l(t)}{dt} \tag{35}$$

where δ_{act} is defined as the actual front tire steering angle input and δ_{PID} is the output of the PID controller. A maximum steering angle $\delta_{\text{act,max}}$ of $\frac{\pi}{9}$ rad is allowed in accordance with the mechanical steering limits of the vehicle. Here $K_{p,l}, K_{i,l}$ and $K_{d,l}$ are gains of the PID controller. To maintain vehicle stability while avoiding corner-cutting behavior, an adaptive look-ahead distance is used to find the tracking point. The steering controller is illustrated in Fig. 10, where e_l is the error in path tracker, d_l is the look-ahead distance, p_l is the look-ahead point and p_q is the tracking point. The tracking point is governed by:

$$d_l(t) = \begin{cases} d_{l,p}(t), & \text{if } d_{l,\text{min}} < d_{l,p}(t) \\ d_{l,\text{min}}, & \text{otherwise} \end{cases} \tag{36}$$

$$d_{l,p}(t) = u(t) t_l \tag{37}$$

where $u(t)$ is the current speed of the vehicle and $t_l = 1$ s is the look-ahead time. The look-ahead distance is lower bounded by $d_{l,\text{min}} = 1$ m to avoid unstable behavior.

A feedback and feedforward control mechanism as in [39] is adopted for the speed controller. Given the path and the speed profile, the acceleration is derived, and two references are given to the speed controller, namely, the speed reference $u_r(s)$ and the acceleration reference $a_r(s)$, both functions of the traveling distance s. The feedforward part directly calculates the required force to follow the reference acceleration as follows:

$$F_{\text{FFW}} = m a_r + F_{\text{drag}} \tag{38}$$

$$F_{\text{drag}} = W_X \tag{39}$$

where F_{FFW} is the required force in the feedforward part, and F_{drag} is the force that needs to be compensated and equals to the weight projected on the x axis of the vehicle frame.

The feedback part is a proportional controller for the error between the current speed and the reference speed as follows:

$$F_{\text{speedFB}} = k_{\text{speed}}(u - u_r(s)) \tag{40}$$

where k_{speed} is the proportional gain.

With the knowledge of two forces, the total required longitudinal force is estimated as follows:

$$\begin{aligned} F &= F_{\text{FFW}} + F_{\text{speedFB}} \\ &= m a_r + W_X + k_{\text{speed}}(u - u_r(s)) \end{aligned} \tag{41}$$

Given the required longitudinal force, the command on the throttle pedal is calculated. In Project Chrono, a simple powertrain is utilized, and thus the output engine torque is proportional to the throttle pedal command. Thus, the throttle pedal command has a linear relationship with the required longitudinal force as follows.

$$c_{\text{throttle}} = \frac{F}{F_{\text{max}}} \tag{42}$$

where $0 \leq c_{\text{throttle}} \leq 1$ is the throttle pedal command, and F_{max} is the maximum longitudinal force that can be achieved with full throttle.

B. Challenging Slopes

Two challenging slopes are designed with analytical expressions of their height (Z coordinate) as a function of the X coordinate. This case mimics the scenarios when the vehicle is climbing a hill. One slope is expressed as $Z = \frac{3}{1+e^{-1.5 \cdot X}}$, and the other as $Z = \frac{9}{1+e^{-0.3 \cdot X}}$. The first slope is short but steep, which is prone to make the vehicle lift off the terrain at the top of the hill. The second slope is smoother but long, which imposes difficulty in climbing up.

The simulation results of the first slope are shown in Fig. 11-A. Without the consideration of dynamical constraints, the vehicle guided by the benchmark path planner with 9 m/s flies over the terrain at the top of the hill, imposing danger, as all wheels lose contact with the terrain. During this flight period, control over the vehicle is completely lost. The proposed global trajectory planner is aware of the importance of limiting the maximum speed, which is bounded by the maximum centrifugal force that overcomes gravity as described in Eq. (22). Therefore, it can achieve a safe climb. Achieving a similar result with the benchmark planner requires the user to select a suitable speed. In this case, a speed setpoint of 5 m/s gives a satisfactory result, but the benchmark algorithm has no way of designing this speed.

In the second slope, as shown in Fig. 11-B, the benchmark path planner with 5 m/s speed fails due to insufficient momentum to overcome gravity. Even when the throttle is commanded at 100% at the end, the traction force is still

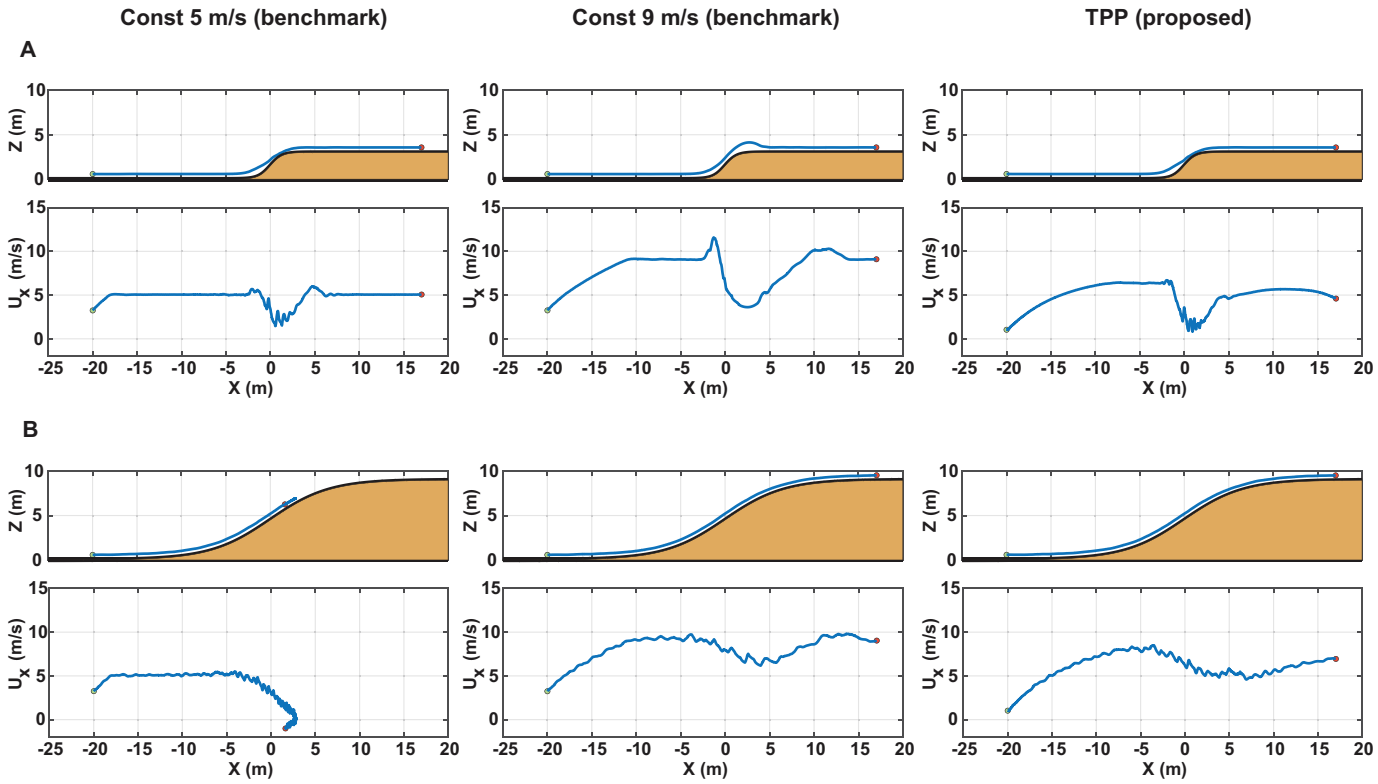


Fig. 11. The tracking results for challenging slopes. (**A**) Results in the short and steep slope. The three columns represent the results of the benchmark path planner with 5 m/s, benchmark path planner with 9 m/s, and the proposed three-phase trajectory planner, respectively. The first row shows the shape of the simulated path in the X-Z plane. The second row shows the profile of the simulated longitudinal speed along the X-axis. (**B**) Results in the long and smooth slope. The configurations are the same with (**A**).

smaller than the gravity force, causing the vehicle to slide back. In contrast, the proposed planner evaluates the required momentum successfully and accelerates more before the hill, preventing the vehicle from losing mobility before reaching the top. A similar result can be obtained with the benchmark planner at 9 m/s speed, but again, the algorithm is not capable of planning this speed on its own.

These results emphasize the importance of speed planning to safely navigate the vehicle in the off-road context and demonstrate the speed planning capability of the TPP framework.

C. Challenging Terrain

The challenging terrain is designed as $Z = -9\sin\left(\frac{X}{15}\right)\sin\left(\frac{Y}{15}\right)$ to represent a hilly topology. An obstacle field is placed to impose navigation challenges and is designed to be symmetric about the line between the start and goal positions to make comparisons between left and right oriented paths fair.

The results of TPP and the benchmark path planner with 5 m/s and 9 m/s speeds are shown in Fig. 12. TPP successfully navigates the vehicle to the goal as shown in Fig. 12-A. The planner recognizes the limitation in maximum available friction force, and thus intentionally turns around the first obstacle, avoiding large bank angles. After the first obstacle, the vehicle navigates to approach the hill with the most gentle slope. The second column of Fig. 12-A plots the actual speed of the vehicle along the actual path with a color map. The color from dark to bright represents the speed from low to high. TPP plans a low speed where the terrain has large bank angles and the vehicle makes large turns, showing its capability to understand the limitation of maximum speed bounded by the maximum centrifugal force along the geodesic curvature that the available friction force can overcome. The third column of Fig. 12-A compares the planned versus tracked speed. With TPP, the vehicle can track the planned speed well due to the consideration of the dynamical limits in the planning.

In Fig. 12-B, using the benchmark planner at 5 m/s, the vehicle is navigated to track along the path that has large bank angles, because the planner lacks consideration of dynamical constraints. On the first convex hill, the vehicle is laterally dragged down by gravity due to insufficient traction, further causing the vehicle to lose mobility before it climbs. As a result, the vehicle loses control at the end of the first convex hill and crashes into an obstacle.

In Fig. 12-C, using the benchmark planner at 9 m/s, the vehicle traverses through the first convex hill, though there are large deviations in path tracking. The vehicle starts losing control when moving down at high speed, and cannot reach the goal.

These results demonstrate the capability of TPP to make a complex coordination of speed and steering for a purposeful exploitation of vehicle dynamics to navigate successfully. The

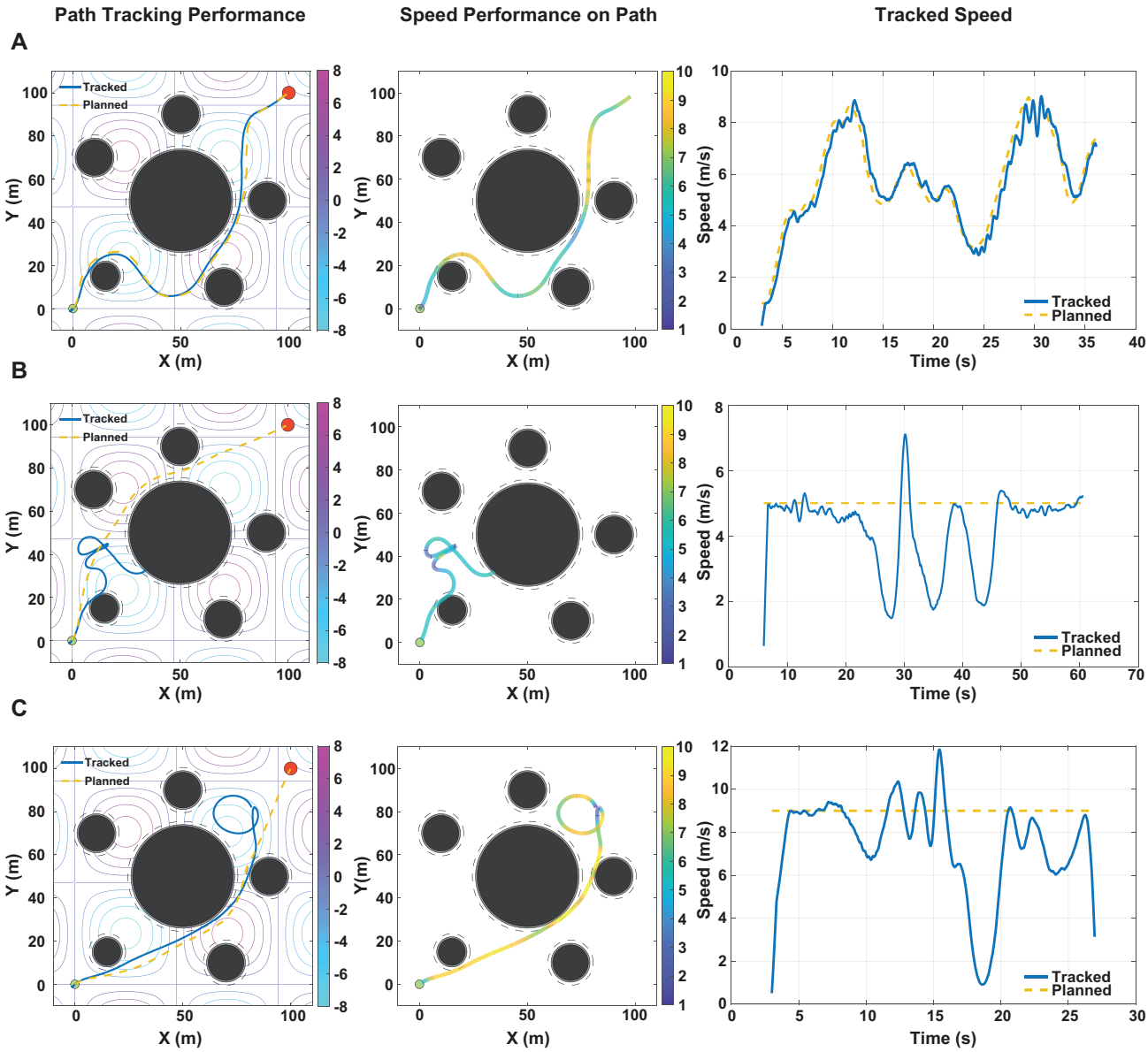


Fig. 12. Tracking results on the challenging terrain with (**A**) the proposed three-phase planner, (**B**) the benchmark path planner with 5 m/s, and (**C**) the benchmark path planner with 9 m/s. For each subplot, the first column shows the result of path tracking performance. The terrain topology is depicted by the contour plot. The black circles are the obstacles. The dashed black rings are the safety margin considering the half-width of the vehicle and the tracking error. The green and red circles are the start and goal points. The second column shows the speed performance. The speed is overlaid on the path with a color map. A lighter color represents a faster speed. The third column shows the tracked speed vs. planned speed.

importance of the capability is underlined by the inability of the benchmark algorithm to achieve similar performance, neither at 5 m/s nor at 9 m/s.

D. Randomized Scenario

The algorithms are tested in 100 randomized scenarios to evaluate their actual cost performance. To find the correlation between the planned cost and actually tracked cost, A*+RRT is used as an additional benchmark.

The statistics are shown in Fig. 13-A. Except for the path planner with 9 m/s, all algorithms have a 100% success rate. When the vehicle makes U-turns, the available friction is not sufficient to let the vehicle track the path with 9 m/s, making the vehicle slide and lose control. This could ultimately result in crashes or complete loss of mobility.

In terms of the actual tracking cost metric, the four groups all have a statistically significant difference between each other. The cost values are normalized based on the cost of the proposed method. The benchmark path planner with 5 m/s has the second lowest cost due to its faster speed than the two phase algorithm, while the benchmark path planner with 9 m/s has the largest value because the vehicle consistently loses control and slides more than the other algorithms. The biggest difference between each algorithm lies in the time-to-

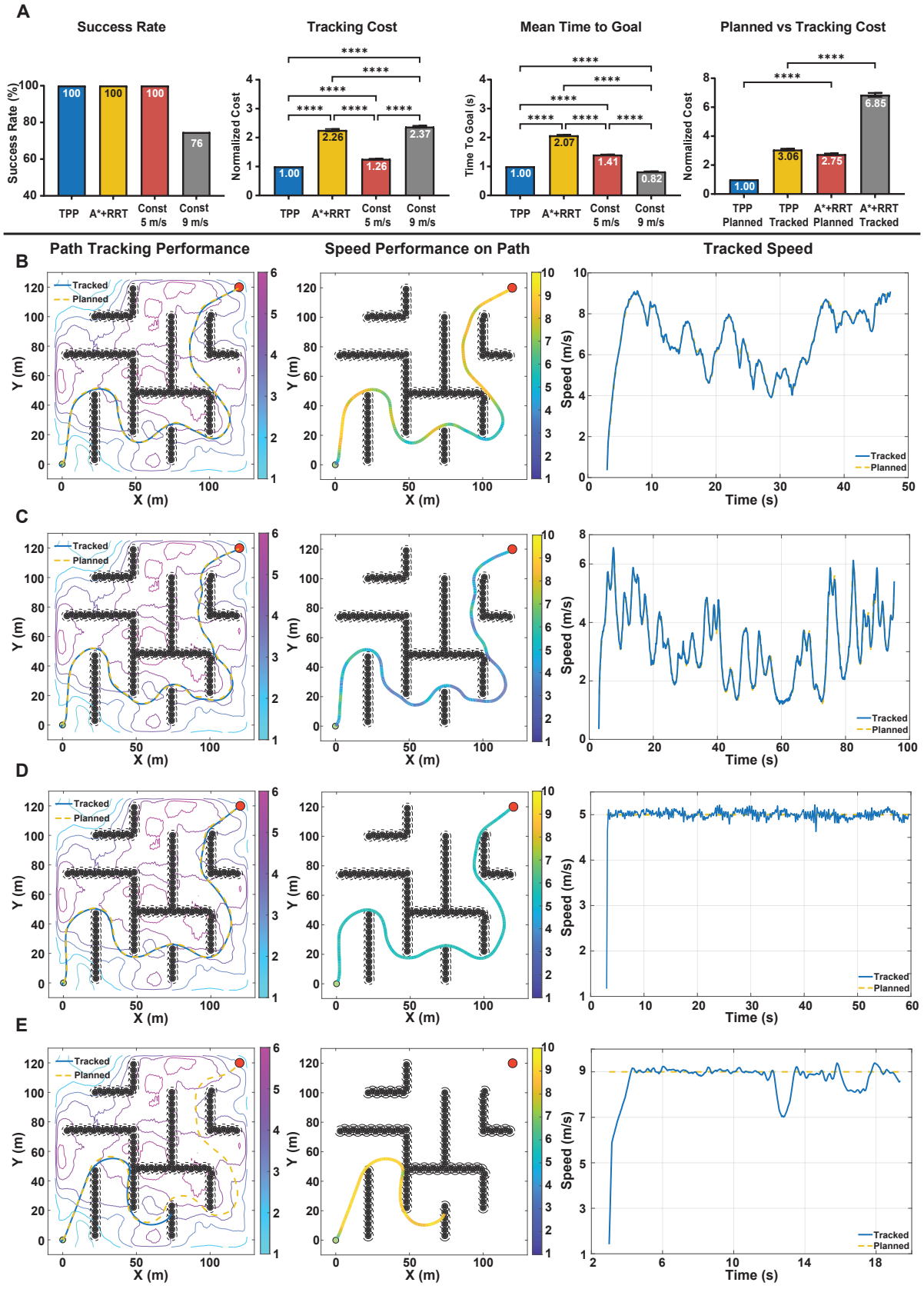


Fig. 13. The results in the randomized scenario. (**A**) Statistics of the tracking results in the randomized scenario: success rate, tracking cost, mean time to goal, and planned vs. tracking cost. (**B - E**) An example randomized scenario. (**B**) Results with the proposed three-phase algorithm. (**C**) Results with A*+RRT. (**D**) Results with benchmark path planner with 5 m/s, and (**E**) Results with the benchmark path planner with 9 m/s.

goal metric, which highlights the speed benefit of TPP. The planner with 9 m/s has the smallest value when it is successful, but it also has the largest failing rate. Thus, considering both safety and mobility, TPP has the best performance. Without the LTR phase, A*+RRT results in the longest traveling time.

To further demonstrate the benefits that are gained from adding LTR, a plot that compares the planned and actual costs is provided. Because the TPP uses LTR to refine the trajectory, it has lower cost. This benefit extends to the actual cost, as the actual cost is significantly lower with TPP compared to A*+RRT.

An example showing the typical performance of each controller is provided through Fig. 13-B to E, representing TPP, A*+RRT, benchmark planner with 5 m/s and benchmark planner with 9 m/s. For each algorithm, a planned path versus tracked path is given in the first figure. The second plot is the speed plot along the path where the color on the path indicates the speed. The last plot is the planned speed versus the actual speed.

TPP plans lower speed when the vehicle encounters large turns and speeds up when the planned path is straight. A*+RRT has lower speed and more speed oscillation compared to TPP due to the higher trajectory cost of the planned trajectory. The vehicle takes 100 s to reach the target, resulting in a higher actual cost than TPP. For the benchmark planner with 5 m/s, the vehicle also succeeds in completing the path. The consideration of only the nonholonomic constraints is sufficient for navigating the vehicle at low speed when the terrain is not challenging. In this case, the vehicle takes 60 s to travel to the goal, which is around 15 s longer than that of TPP. In the benchmark planner with 9 m/s, the vehicle first shows large deviations of the actual path from the planned path and then crashes into the obstacles. This demonstrates the importance of considering dynamical constraints when the vehicle is traveling at high speed. In this case, the consideration of only the nonholonomic constraints is not enough.

Thus, concluding from these results, the benefit of adding LTR is further demonstrated by showing the improved tracking cost as well as the mean time to goal. The ability to take dynamical constraints into account is shown by maintaining a 100% success rate while achieving higher mobility than that of the benchmark planner with 5 m/s speed. The proposed global planner is able to fulfill its task of planning dynamically feasible trajectories while increasing mobility on off-road 3D terrains.

VII. Conclusions

This work develops a novel three-phase algorithm for global trajectory planning on off-road 3D terrains. Its novelty lies in the hierarchical three-phase architecture designed for lower trajectory cost, computational efficiency, and success rate. The time-efficient design considers the nonholonomic and dynamical constraints for vehicle safety and mobility.

The importance of the three-phase framework is demonstrated by showing the contribution of each phase one by one. The contribution of the first phase, namely A* guidance, is shown by comparing the algorithms with and without A*. The results validate that the success rate of finding a feasible trajectory is greatly increased with the sampling guidance. The contribution of the third phase, namely LTR, is demonstrated by comparing the algorithms with and without LTR, and also by replacing RRT with RRT*, which is known for its asymptotic optimality. With statistically significant differences in the planning cost and solving time, it is shown that adding LTR is an efficient way to smooth the trajectory to gain a solution with lower cost. The contribution of the second phase is demonstrated in high-fidelity simulations as accounting for the nonholonomic and dynamic constraints. The proposed algorithm is compared to a state-of-the-art global path planning algorithm. It is shown that the proposed algorithm successfully navigates the vehicle in challenging slopes and terrains, while the benchmark fails in at least one of the tasks. The results show that the constraints included in the proposed planner can successfully avoid losing contact with the terrain or losing control due to the lack of friction force or engine torque, thereby improving vehicle safety. Then, the proposed algorithm is tested in randomized terrain and obstacle fields. Vehicle safety is further validated through a perfect success rate. The ability of the proposed algorithm to plan a trajectory with high mobility is also validated by showing the low tracking cost and short mean time to the goal.

Therefore, the proposed algorithm is an important step toward off-road 3D navigation due to its high success rate and ability to account for vehicle safety and mobility constraints in a computationally efficient way.

Future work involves taking the terrain roughness and traversability into account, such as rock, vegetation, and mud. Adaption to uncertainties due to incomplete or inaccurate terrain information as well as partial observability is expected to make planning more robust. The computational performance can be further improved by applying parallel computing to the third phase. A more sophisticated design of the local controller can be used for better tracking performance. Finally, testing the algorithm experimentally on a physical vehicle is also of interest.

References

[1] L. Bascetta, D. Cucci, G. Magnani, M. Matteucci, D. Osmankovic, and A. Tahirovic, "Towards the implementation of a MPC-based planner on an autonomous all-terrain vehicle," in *Proceedings of Workshop on Robot Motion Planning: Online, Reactive, and in Real-time*, 2012, pp. 1–7.

[2] I. Rekleitis, J.-L. Bedwani, E. Dupuis, T. Lamarche, and P. Allard, "Autonomous over-the-horizon navigation using LIDAR data," *Autonomous Robots*, vol. 34, no. 1-2, pp. 1–18, 2012.

[3] D. Osmankovic, A. Tahirovic, and G. Magnani, "All terrain vehicle path planning based on D* lite and MPC based planning paradigm in discrete space," in *IEEE International Conference on Advanced Intelligent Mechatronics*, 2017, pp. 334–339.

[4] D. Ferguson and A. Stentz, "The Field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments," *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19*, 2005.

[5] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global path planning on board the mars exploration rovers," in *IEEE Aerospace Conference*, 2007, pp. 1–11.

[6] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[7] M. Pivtoraiko, T. M. Howard, I. Nesnas, and A. Kelly, "Field experiments in rover navigation via model-based trajectory generation and nonholonomic motion planning in state lattices," in *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2008, pp. 25–29.

[8] B. Hedegaard, E. Fahnestock, J. Arkin, A. Menon, and T. M. Howard, "Discrete optimization of adaptive state lattices for iterative motion planning on unmanned ground vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 5764–5771.

[9] K. Iagnemma, S. Shimoda, and Z. Shiller, "Near-optimal navigation of high speed mobile robots on uneven terrain," in *Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 4098–4103.

[10] J. Wurts, J. L. Stein, and T. Ersal, "Collision imminent steering on curved roads using one-level nonlinear model predictive control," *IEEE Access*, vol. 9, pp. 39 292–39 302, 2021.

[11] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *American Control Conference*, 2019, pp. 717–722.

[12] J. Dallas, M. P. Cole, P. Jayakumar, and T. Ersal, "Terrain adaptive trajectory planning and tracking on deformable terrains," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 11 255–11 268, 2021.

[13] S. Yu, C. Shen, and T. Ersal, "Nonlinear model predictive planning and control for high-speed autonomous vehicles on 3D terrains," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 412–417, 2021.

[14] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation," *arXiv:2103.02828*, 2021.

[15] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6054–6060, 2021.

[16] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: The 16th International Symposium ISRR*. Springer, 2016, pp. 649–666.

[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. Paton, M. P. Strub, T. Brown, R. J. Greene, J. Lizewski, V. Patel, J. D. Gammell, and I. A. Nesnas, "Navigation on the line: Traversability analysis and path planning for extreme-terrain rappelling rovers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 7034–7041.

[19] T. McMahon, A. Sivaramakrishnan, K. Kedia, E. Granados, and K. E. Bekris, "Terrain-aware learned controllers for sampling-based kinodynamic planning over physically simulated terrains," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 2925–2930.

[20] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," *IEEE International Conference on Robotics and Automation*, pp. 5539–5544, 2013.

[21] C. Shen, S. Yu, and T. Ersal, "A three-phase framework for global path planning for nonholonomic autonomous vehicles on 3D terrains," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 160–165, 2021.

[22] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[23] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation based near-optimal motion planning with kinodynamic constraints using RRT," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8718–8729, 2018.

[24] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, p. 3170, 2018.

[25] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[26] B. Song, G. Tian, and F. Zhou, "A comparison study on path smoothing algorithms for laser robot navigated mobile robot path planning in intelligent space," *Journal of Information and Computational Science*, vol. 7, no. 1, pp. 2943–2950, 2010.

[27] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on Bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science*. IEEE, 2008, pp. 158–166.

[28] T. Overbye and S. Saripalli, "Path optimization for ground vehicles in off-road terrain," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 7708–7714.

[29] C. M. Kang, S.-H. Lee, and C. C. Chung, "Comparative evaluation of dynamic and kinematic vehicle models," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 648–653.

[30] H. Febbo, P. Jayakumar, J. L. Stein, and T. Ersal, "NLOptControl: A modeling language for solving optimal control problems," *arXiv:2003.00142*, 2020.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[32] J. Wurts, J. Dallas, J. L. Stein, and T. Ersal, "Adaptive nonlinear model predictive control for collision imminent steering with uncertain coefficient of friction," in *American Control Conference*, 2020, pp. 4856–4861.

[33] Y. Du, C. Liu, Y. Song, Y. Li, and Y. Shen, "Rapid estimation of road friction for anti-skid autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2461–2470, 2019.

[34] A. M. Ribeiro, A. Moutinho, A. R. Fioravanti, and E. C. de Paiva, "Estimation of tire–road friction for road vehicles: A time delay neural network approach," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 1, p. 4, 2020.

[35] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

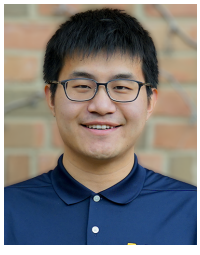
[36] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[37] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Communications of the ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[38] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering*. Springer, 2016, pp. 19–49.

[39] K. Kritayakirana and J. C. Gerdes, "Autonomous vehicle control at the limits of handling," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 4, pp. 271–296, 2012.

Congkai Shen earned his Bachelor of Science in Electrical and Computer Engineering from Shanghai Jiao Tong University, China, in 2020. Subsequently, he completed his Bachelor of Science and Master of Science in Mechanical Engineering at the University of Michigan, Ann Arbor, in 2020 and 2022, respectively. Currently, he is pursuing the Ph.D. degree in Mechanical Engineering at the University of Michigan, Ann Arbor. His research focuses on modeling, system identification, motion planning, and control, in the context of vehicle systems.



Siyuan Yu received his B.S.E in Electrical and Computer Engineering from Shanghai Jiao Tong University, China, in 2020 and the B.S.E and M.S.E in Mechanical Engineering from University of Michigan, Ann Arbor in 2020 and 2022, respectively. He is currently pursuing the Ph.D. degree in Mechanical Engineering at the University of Michigan, Ann Arbor. His research interests include modeling, system identification, motion planning and control, with respect to vehicle systems.

Bogdan I. Epureanu received his Ph.D. in Mechanical Engineering from Duke University in 1999. Currently, he is the Roger L. McCarthy Professor and an Arthur F. Thurnau Professor in the Department of Mechanical Engineering at the University of Michigan and has a courtesy appointment in Electrical Engineering and Computer Science. He is the Director of the Automotive Research Center. His research focuses on nonlinear dynamics of complex systems, such as teaming of autonomous vehicles, enhanced aircraft safety and performance, aeromechanics, early detection of neurodegenerative diseases, forecasting tipping points in disease epidemics and ecology.

Tulga Ersal received the B.S.E. degree from the Istanbul Technical University, Istanbul, Turkey, in 2001, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, MI USA, in 2003 and 2007, respectively, all in mechanical engineering. He is currently an Associate Professor in the Department of Mechanical Engineering, University of Michigan, Ann Arbor. His research interests include modeling, simulation, and control of dynamic systems, with applications to vehicle and energy systems.