sensors

MDPI

Article

Traversability Assessment and Trajectory Planning of Unmanned Ground Vehicles with Suspension Systems on Rough Terrain

Kai Zhang [1], Yi Yang [1,*], Mengyin Fu [1,2] and Meiling Wang [1]

1 School of Automation, Beijing Institute of Technology, Beijing 100081, China; kaizhangbit@gmail.com (K.Z.); fumy@bit.edu.cn (M.F.); wangml@bit.edu.cn (M.W.)
2 School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China
* Correspondence: yang_yi@bit.edu.cn; Tel.: +86-10-6891-3985

Received: 8 August 2019; Accepted: 27 September 2019; Published: 10 October 2019

Abstract: This paper presents a traversability assessment method and a trajectory planning method. They are key features for the navigation of an unmanned ground vehicle (UGV) in a non-planar environment. In this work, a 3D light detection and ranging (LiDAR) sensor is used to obtain the geometric information about a rough terrain surface. For a given SE(2) pose of the vehicle and a specific vehicle model, the SE(3) pose of the vehicle is estimated based on LiDAR points, and then a traversability is computed. The traversability tells the vehicle the effects of its interaction with the rough terrain. Note that the traversability is computed on demand during trajectory planning, so there is not any explicit terrain discretization. The proposed trajectory planner finds an initial path through the non-holonomic A*, which is a modified form of the conventional A* planner. A path is a sequence of poses without timestamps. Then, the initial path is optimized in terms of the traversability, using the method of Lagrange multipliers. The optimization accounts for the model of the vehicle's suspension system. Therefore, the optimized trajectory is dynamically feasible, and the trajectory tracking error is small. The proposed methods were tested in both the simulation and the real-world experiments. The simulation experiments were conducted in a simulator called Gazebo, which uses a physics engine to compute the vehicle motion. The experiments were conducted in various non-planar experiments. The results indicate that the proposed methods could accurately estimate the SE(3) pose of the vehicle. Besides, the trajectory cost of the proposed planner was lower than the trajectory costs of other state-of-the-art trajectory planners.

Keywords: autonomous navigation; mobile robot; unmanned ground vehicle; light detection and ranging sensor; rough terrain

1. Introduction

Recently, ground mobile robots with different functions start to play essential roles in people's daily life. In particular, autonomous navigation in a non-planar environment is an important function. It enables competent operations of a ground mobile robot in many challenging applications, such as surveillance, rescue, and planet exploration. In this search field, two basic and critical problems need to be addressed: traversability assessment and trajectory planning. On the one hand, the traversability tells a robot the effects of its interaction with a rough terrain surface. The terrain information is obtained using a 3D light detection and ranging (LiDAR) sensor and/or a visual sensor. On the other hand, a trajectory planner utilizes a cost function considering the traversability to determine a dynamically feasible motion trajectory. This trajectory permits a robot to move from an initial pose to a goal pose. A pose is made up of the position (x, y, z) and the orientation (roll, pitch, yaw) of the robot.

On an uneven terrain surface, the motion of a ground robot is constrained by the terrain shape. Conventional trajectory planners that assume the terrain to be flat are not applicable, because the trajectories generated by them are hard to be tracked or even non-traversable. These trajectories can cause wheel slip, high roll/pitch angle, and so on. Therefore, it is necessary to quantitatively assess the utility of passing through an uneven terrain surface, and then generate a trajectory based on the utility. This utility is called traversability. Planning based on traversability can reduce the trajectory tracking error in a predictive way at the planning time, which otherwise is reduced using a feedback-based trajectory tracker at the control time.

Among different types of mobile robots, car-like unmanned ground vehicles (UGVs) are rapidly gaining popularity from researchers. A car-like vehicle is often equipped with a suspension system, which is the system of tires, tire air, springs, shock absorbers, and linkages that connects the vehicle to its wheels and allows relative motion between the two. The suspension system has a significant effect on the traversability of the vehicle. However, the suspension system was often neglected in previous work. Therefore, to generate a dynamically feasible trajectory, the proposed approach incorporates the model of the vehicle's suspension system.

The purpose of this work is to navigate a car-like UGV safely and efficiently, with a focus on rough and unstructured terrain. The proposed methods can help an autonomous vehicle to be applied in many challenge applications. For example, the car-like vehicle may be required to dig a hole or unload something on a complex terrain surface. Besides, in the application of teleoperations, the proposed planner can be used to autonomously drive the vehicle to an operator-designated pose. This can reduce the workload of the operator and the telemetry bandwidth.

In summary, the contributions of this paper and the characteristics of the proposed methods are shown as follows:

1. The suspension system of the vehicle is used to reduce the pose estimation error and optimize the trajectory. The optimized trajectory is easy to be tracked by the vehicle in non-planar environments.
2. The traversability is assessed on demand based on original LiDAR points during trajectory planning, without any kind of explicit terrain surface reconstruction or discretization. This feature makes the proposed method efficient in terms of computation and storage.
3. The cost function and the node-expansion rule of the conventional A* are modified to obtain a path satisfying non-holonomic constraints. This path is then optimized by a constraint-aware optimizer based on a custom cost function. The final trajectory is smoother and more traversable than those generated by other state-of-the-art methods.
4. The proposed traversability assessor (or trajectory planner) is general and can be used with any other motion planning method (or traversability assessment method).

The rest of this paper is organized as follows. First, some previous researches about the traversability assessment and trajectory planning in a non-planar environment are reviewed. Then, Section 2 introduces the architecture of the proposed methods briefly. Section 3 describes how to assess the traversability of a ground vehicle based on its suspension system using a 3D LiDAR. Section 4 introduces how to generate and optimize a vehicle trajectory on rough terrain using the assessed traversabilities. Section 5 shows and analyzes the results of some simulation and real-world experiments. Finally, the paper is concluded and a direction for future work is suggested.

1.1. Related Work

The safe and efficient navigation of an unmanned ground vehicle requires the terrain information, which can be exploited to predict the future pose of the vehicle and assess the traversability. The sensors used to obtain the terrain information include visual sensors [1–5], LiDARs [6–13], and so on. Visual sensors can provide various kinds of terrain information, but processing visual data often requires a long computation time. What is more, the performance of visual sensors may vary with the intensity of sunlight [14]. The point clouds from LiDARs usually occupy large storage space, but the performance of a LiDAR-based terrain mapping approach is often relatively stable [15]. The point clouds or the visual data can be converted into traversabilities directly, or be converted into a digital elevation map (DEM) [16] or a 3D grid map [17,18]. A DEM is a 2.5D grid map with each grid value representing the height information about the terrain surface. It is easy to be implemented but cannot model overhanging obstacles, such as bridges. Besides, accurately computing the traversability needs a high resolution DEM, but building and maintaining a high resolution DEM is time-consuming. A 3D grid map is able to reconstruct an environment and suitable for the navigation in a multi-level building [19], but it is inefficient in terms of computation and storage. Compared with the existing methods, the proposed method assesses the vehicle traversability on demand using the original point cloud from a 3D LiDAR. It does not include complex terrain mapping process, so it is more efficient in terms of computation and storage.

A traversability assessor provides a mapping from the terrain maps or the sensor data to the traversabilities. It evaluates the mobility of a ground vehicle. The traversability depends on the pose of the vehicle and the terrain shape. The approaches that assess the traversability based on the terrain maps can be classified into appearance-based methods [20], geometry-based methods [21,22], and learning-based methods [23–25]. The approaches that assess the traversability based on original sensor data are often free of artificial discretization, which is inherent to DEMs, 3D grid maps, or other classical terrain models. For instance, Krüsi et al. propose an approach to determine the traversability of a specific pose without any discretization [26]. This approach can plan a trajectory directly on LiDAR points. Santamaria-Navarro et al. present a method for the large-scale traversability classification of point clouds [27]. In this method, the model for the classification of points is learned from training data using Gaussian processes. However, the existing methods usually ignore the vehicle's suspension system.

Once the traversability is known, a trajectory should be generated. In the context of trajectory planning, researchers have paid much attention to the generation of a collision-free trajectory assuming flat terrain. Common trajectory planning techniques include artificial potential field (APF) methods [28,29], optimization-based methods [30,31], search-based methods [32,33], sampling-based methods [34,35], and so on. However, in a non-planar environment, more complicated cost functions and vehicle models must be used. For example, Amar et al. adapt the trajectory to the rough terrain using the kinematic model of the vehicle [36], and Howard et al. propose to optimize a trajectory by the model-based simulation of a vehicle on the rough terrain [37]. Recently, the trajectory planning methods based on machine learning are rapidly gaining popularity, especially the end-to-end learning methods. The main idea of these methods is to directly learn a mapping from the original sensor data to a trajectory. Learning-based planners have enabled a long-range autonomous navigation using a single stereo camera [38] or a LiDAR [39]. In terms of application, much of the work on trajectory planning in a non-planar environment focused on rovers for planetary exploration [40,41] or military vehicles [42].

The existing traversability assessors and trajectory planners seldom consider the suspension model of the vehicle. They usually take the whole vehicle as a rigid body. However, in a non-planar environment, the traversability of the vehicle largely depends on its suspension system. Therefore, the proposed methods incorporate the vehicle's suspension model in traversability assessment and trajectory planning. This is essential for safe and efficient navigation in a non-planar environment.

2. System Architecture

Figure 1 shows the architecture of the proposed traversability assessor and trajectory planner. The inputs of the system are a start pose and a goal pose. A non-holonomic A* planner generates an initial path, which is then optimized by a trajectory optimizer. Formally, a path is defined as the following mapping: $[0, 1] \rightarrow \mathcal{X}$, and a trajectory is defined as the following mapping: $[0, T] \rightarrow \mathcal{X}$ prescribing the evolution of the state of the vehicle in time, where T is the time instant at which the vehicle reaches the end of the trajectory, and $\mathcal{X}$ is the state space of the vehicle. The state of the vehicle can be defined as the pose (position and orientation) of the vehicle. The final output of the system is a solution trajectory. During the generation and the optimization of the trajectory, the traversability assessor is invoked by both the planner and the optimizer on demand to calculate the traversabilities at required poses. This calculation is based on a suspension model and the point clouds from a 3D LiDAR. Note that the traversability assessment is not treated as a separate and upstream process. It is regarded as an integral part of trajectory planning.

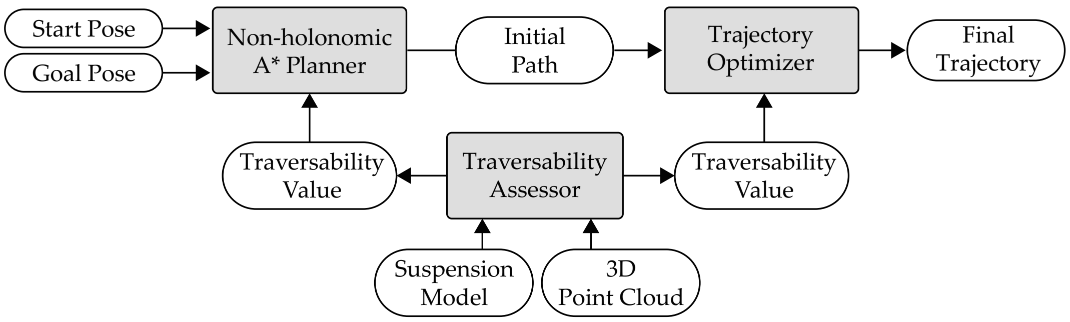


Figure 1. The architecture of the proposed traversability assessor and trajectory planner.

3. Traversability Assessment Using LiDAR

In this section, a method will be introduced to assess the traversability of a vehicle using a 3D LiDAR. The traversability depends on the pose of the vehicle, the terrain roughness, and the height difference. The vehicle pose is estimated based on the point cloud and the suspension system. The overview of this section is shown in Figure 2.

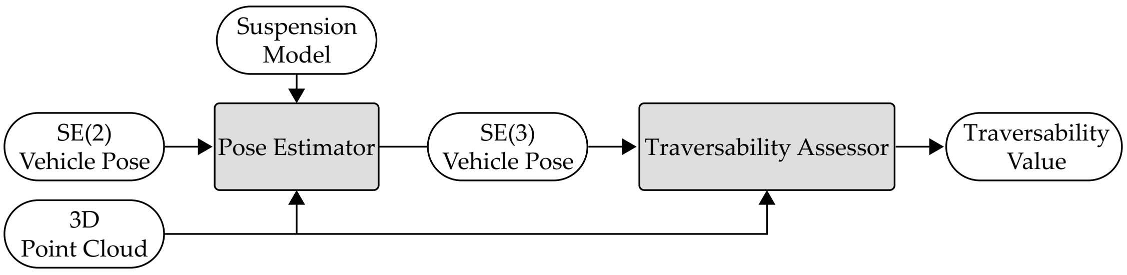


Figure 2. The overview of the traversability assessment approach.

3.1. Light Detection and Ranging

In this work, the 3D LiDAR used to acquire the geometric information of rough terrain is HDL-64E S2 or HDL-32E developed by Velodyne LiDAR, as shown in Figure 3. The manufactory is located at Silicon Valley. The specifications of the LiDAR are summarized in Table 1. The output of the 3D LiDAR is a set of points, called a point cloud. The point cloud is a continuous terrain representation that is free of any artificial discretization. Hence, It is suitable for accurately computing the traversabilities at specific vehicle poses. Besides, it is a by-product of the SLAM module, so we do not spend additional computation time on building and maintaining terrain maps. Next, we will compute the vehicle pose and the traversability based on the point cloud.

(a) HDL-64E S2 (b) HDL-32E

Figure 3. The 3D light detection and ranging (LiDAR) used in this work.

Table 1. The specifications of the LiDARs.

Parameter	**HDL-64E S2**	**HDL-32E**
Distance accuracy	<2 cm	<2 cm
Measurement range	50 m for pavement and 120 m for cars and foliage	70 m
Vertical field of view	$+2.0°$ to $-24.8°$	$+10.7°$ to $-30.7°$
Vertical angular resolution	0.4°	1.33°
Horizontal angular resolution	0.09°	0.16°
# Points per second	1,333,000	700,000

3.2. Pose Estimation

A ground vehicle is always constrained to move on the surface of rough terrain, so the height, roll, and pitch of the vehicle are controlled by the local geometry of the terrain surface and need to be estimated. Formally, pose estimation can be defined as the following function:

$$f_{\mathrm{pe}} : (\mathbb{P}, \overline{s}) \to \widetilde{s} \tag{1}$$

where $\mathbb{P}$ is a point cloud that is constructed before trajectory planning. $\overline{s} = \left[\overline{x} \;\; \overline{y} \;\; \overline{\theta}_z\right]^{\mathrm{T}} \in \mathrm{SE}(2)$ is a query pose, and $\widetilde{s} = \left[\widetilde{x} \;\; \widetilde{y} \;\; \widetilde{z} \;\; \widetilde{\theta}_x \;\; \widetilde{\theta}_y \;\; \widetilde{\theta}_z\right]^{\mathrm{T}} \in \mathrm{SE}(3)$ is the estimated pose on the terrain surface. Note that SE(2) is a state-space whose element is composed of x-coordinate, y-coordinate, and yaw. SE(3) is a state-space whose element is composed of x-coordinate, y-coordinate, z-coordinate, roll, pitch, and yaw. In the following, a pose estimation method developed assuming that the vehicle is static. This approach first calculates a roll and a pitch according to the wheel-terrain interaction, and then estimates the SE(3) pose of the vehicle based on its suspension system. The proposed pose estimation method is based on Point Clouds and vehicle Suspension models, so it is called PC-Sus.

3.2.1. Euler Angle Estimation Based on Wheel-Terrain Interaction

First, the position of the vehicle's wheels on the terrain surface is calculated. For a given query pose $\overline{s} = \left[\overline{x} \;\; \overline{y} \;\; \overline{\theta}_z\right]^{\mathrm{T}}$, the planar coordinates of the right front wheel are calculated as:

$$\begin{bmatrix} x_{\mathrm{rf}} \\ y_{\mathrm{rf}} \end{bmatrix} = \begin{bmatrix} \cos\overline{\theta}_z & \sin\overline{\theta}_z \\ -\sin\overline{\theta}_z & \cos\overline{\theta}_z \end{bmatrix} \begin{bmatrix} W' \,/\, 2 \\ L' \end{bmatrix} + \begin{bmatrix} \overline{x} \\ \overline{y} \end{bmatrix}, \tag{2}$$

where W' and L' are the length of the vehicle's axle and the wheelbase, respectively. Let $\mathbb{P}_{\mathrm{rf}}$ denote the following 3D point set:

$$\mathbb{P}_{\mathrm{rf}} = \left\{ \boldsymbol{p} = [x \;\; y \;\; z]^{\mathrm{T}} \mid (x - x_{\mathrm{rf}})^2 + (y - y_{\mathrm{rf}})^2 < \left(\frac{W_{\mathrm{tire}}}{2}\right)^2, \; \boldsymbol{p} \in \mathbb{P} \right\}, \tag{3}$$

where W_{tire} is the width of the vehicle's tire. $\mathbb{P}_{\mathrm{rf}}$ is the set of the LiDAR points that are near the contact area between the vehicle's tire and the terrain surface. Then, the center of gravity of the points in $\mathbb{P}_{\mathrm{rf}}$ is computed as:

$$\boldsymbol{p}_{\mathrm{average}} = \frac{1}{n} \sum_{\boldsymbol{p} \in \mathbb{P}_{\mathrm{rf}}} \boldsymbol{p} \tag{4}$$

where n is the number of points in $\mathbb{P}_{\mathrm{rf}}$. Let z_{rf} be the z-coordinate of $\boldsymbol{p}_{\mathrm{average}}$. Finally, the position of the right front wheel on the terrain surface can be represented by $\boldsymbol{p}_{\mathrm{rf}} = [x_{\mathrm{rf}} \;\; y_{\mathrm{rf}} \;\; z_{\mathrm{rf}}]^{\mathrm{T}}$. The positions of the right back wheel, the left front wheel, and the left back wheel are represented by $\boldsymbol{p}_{\mathrm{rb}}$, $\boldsymbol{p}_{\mathrm{lf}}$, and $\boldsymbol{p}_{\mathrm{lb}}$ respectively, which are all computed in a similar way.

In this paper, every three wheel positions are defined as a *triple*. For a given triple $\boldsymbol{\alpha} = [\boldsymbol{p}_1 \;\; \boldsymbol{p}_2 \;\; \boldsymbol{p}_3]^{\mathrm{T}}$ ($\boldsymbol{p}_i = [x_i \;\; y_i \;\; z_i]^{\mathrm{T}}$), the normal vector of the plane passing through these three points is:

$$\boldsymbol{n}(\boldsymbol{\alpha}) = (\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1) = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = a\boldsymbol{i} + b\boldsymbol{j} + c\boldsymbol{k} = [a \ \ b \ \ c]^{\mathrm{T}}, \quad (5)$$

where $\boldsymbol{i} = [1 \ 0 \ 0]^{\mathrm{T}}$, $\boldsymbol{j} = [0 \ 1 \ 0]^{\mathrm{T}}$, $\boldsymbol{k} = [0 \ 0 \ 1]^{\mathrm{T}}$, $a = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1)$, $b = (z_2 - z_1)(x_3 - x_1) - (z_3 - z_1)(x_2 - x_1)$, and $c = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)$. Then, the roll $(\theta_x(\boldsymbol{\alpha}))$ and the pitch $(\theta_y(\boldsymbol{\alpha}))$ of the plane that passes through the three points are calculated as:

$$\theta_x(\boldsymbol{n}(\boldsymbol{\alpha})) = \mathrm{sgn}(b) \arccos \frac{c}{\sqrt{b^2 + c^2}} \qquad \theta_y(\boldsymbol{n}(\boldsymbol{\alpha})) = \mathrm{sgn}(a) \arccos \sqrt{\frac{b^2 + c^2}{a^2 + b^2 + c^2}} \quad (6)$$

where sgn is the sign function.

Recall that the wheel positions $\boldsymbol{p}_{\mathrm{rf}}$, $\boldsymbol{p}_{\mathrm{rb}}$, $\boldsymbol{p}_{\mathrm{lf}}$ and $\boldsymbol{p}_{\mathrm{lb}}$ have been computed previously. Let $\mathbb{A}$ be a set of triples: $\{[\boldsymbol{p}_{\mathrm{rf}} \ \boldsymbol{p}_{\mathrm{lf}} \ \boldsymbol{p}_{\mathrm{lb}}]^{\mathrm{T}}, [\boldsymbol{p}_{\mathrm{rf}} \ \boldsymbol{p}_{\mathrm{lf}} \ \boldsymbol{p}_{\mathrm{rb}}]^{\mathrm{T}}, [\boldsymbol{p}_{\mathrm{rf}} \ \boldsymbol{p}_{\mathrm{lb}} \ \boldsymbol{p}_{\mathrm{rb}}]^{\mathrm{T}}, [\boldsymbol{p}_{\mathrm{lf}} \ \boldsymbol{p}_{\mathrm{lb}} \ \boldsymbol{p}_{\mathrm{rb}}]^{\mathrm{T}}\}$. Then, a roll (θ_x^*) and a pitch (θ_y^*) can be calculated as:

$$\begin{aligned} \boldsymbol{\alpha}_x^* &= \underset{\boldsymbol{\alpha} \in \mathbb{A}}{\arg\max} \, |\theta_x(\boldsymbol{n}(\boldsymbol{\alpha}))| & \quad \boldsymbol{\alpha}_y^* &= \underset{\boldsymbol{\alpha} \in \mathbb{A}}{\arg\max} \, |\theta_y(\boldsymbol{n}(\boldsymbol{\alpha}))| \\ \theta_x^* &= \theta_x(\boldsymbol{n}(\boldsymbol{\alpha}_x^*)) & \quad \theta_y^* &= \theta_y(\boldsymbol{n}(\boldsymbol{\alpha}_y^*)) \end{aligned} \quad (7)$$

where $\boldsymbol{\alpha}_x^*$ and $\boldsymbol{\alpha}_y^*$ are the triples that make the roll and the pitch equal to the maximal absolute values, respectively. θ_x^* and θ_y^* are the maximal roll and pitch with signs (positive or negative), respectively. If the vehicle is assumed to be a rigid body without a suspension system, θ_x^* and θ_y^* will be the roll and the pitch of the vehicle, respectively.

3.2.2. Pose Estimation Based on Suspension System

Note that θ_x^* and θ_y^* cannot be used as the roll and the pitch of the vehicle, because the above calculations do not consider the suspension model of the vehicle. The role of the suspension model is a theoretical basis used to compute the roll and the pitch. Without the suspension model, the whole vehicle can only be taken as a rigid body. However, a real vehicle is never a rigid body. Next, the roll and the pitch will be computed based on the suspension model. During the computation, an appropriate suspension model will be chosen based on θ_x^* to estimate the SE(3) pose $(\widehat{s})$ of the vehicle.

There are two well known passive suspension models: the half vehicle model and the full vehicle model, as shown in Figure 4a,b. Formally, the dynamic model of a half vehicle contains four linear differential equations [43]:

$$\begin{aligned} m_{\mathrm{s}}\ddot{z}/2 &= -\mu_{\mathrm{f}}(\dot{z}_{\mathrm{s_{rf}}} - \dot{z}_{\mathrm{u_{rf}}}) - \mu_{\mathrm{b}}(\dot{z}_{\mathrm{s_{rb}}} - \dot{z}_{\mathrm{u_{rb}}}) - k_{\mathrm{f}}(z_{\mathrm{s_{rf}}} - z_{\mathrm{u_{rf}}}) - k_{\mathrm{b}}(z_{\mathrm{s_{rb}}} - z_{\mathrm{u_{rb}}}) \\ I_{\mathrm{y}}\ddot{\theta}_y/2 &= -\mu_{\mathrm{f}}L_{\mathrm{f}}(\dot{z}_{\mathrm{s_{rf}}} - \dot{z}_{\mathrm{u_{rf}}}) + \mu_{\mathrm{b}}L_{\mathrm{b}}(\dot{z}_{\mathrm{s_{rb}}} - \dot{z}_{\mathrm{u_{rb}}}) - k_{\mathrm{f}}L_{\mathrm{f}}(z_{\mathrm{s_{rf}}} - z_{\mathrm{u_{rf}}}) + k_{\mathrm{b}}L_{\mathrm{b}}(z_{\mathrm{s_{rb}}} - z_{\mathrm{u_{rb}}}) \\ m_{\mathrm{u_f}}\ddot{z}_{\mathrm{u_{rf}}} &= \mu_{\mathrm{f}}(\dot{z}_{\mathrm{s_{rf}}} - \dot{z}_{\mathrm{u_{rf}}}) + k_{\mathrm{f}}(z_{\mathrm{s_{rf}}} - z_{\mathrm{u_{rf}}}) + k_{\mathrm{t_f}}(z_{\mathrm{rf}} - z_{\mathrm{u_{rf}}}) \\ m_{\mathrm{u_b}}\ddot{z}_{\mathrm{u_{rb}}} &= \mu_{\mathrm{b}}(\dot{z}_{\mathrm{s_{rb}}} - \dot{z}_{\mathrm{u_{rb}}}) + k_{\mathrm{b}}(z_{\mathrm{s_{rb}}} - z_{\mathrm{u_{rb}}}) + k_{\mathrm{t_b}}(z_{\mathrm{rb}} - z_{\mathrm{u_{rb}}}) \end{aligned} \quad (8)$$

Note that the half vehicle model assumes that the roll of the vehicle is zero. The dynamic model of a full vehicle contains seven linear differential equations [44]:

$$
\begin{aligned}
m_s\ddot{\tilde{z}} =& -\mu_f(\dot{z}_{s_{rf}}-\dot{z}_{u_{rf}})-\mu_f(\dot{z}_{s_{lf}}-\dot{z}_{u_{lf}})-\mu_b(\dot{z}_{s_{rb}}-\dot{z}_{u_{rb}})-\mu_b(\dot{z}_{s_{lb}}-\dot{z}_{u_{lb}})\\
&-k_f(z_{s_{rf}}-z_{u_{rf}})-k_f(z_{s_{lf}}-z_{u_{lf}})-k_b(z_{s_{rb}}-z_{u_{rb}})-k_b(z_{s_{lb}}-z_{u_{lb}})\\
I_x\ddot{\tilde{\theta}}_x =& -\mu_f W_f(\dot{z}_{s_{rf}}-\dot{z}_{u_{rf}})+\mu_f W_f(\dot{z}_{s_{lf}}-\dot{z}_{u_{lf}})-\mu_b W_b(\dot{z}_{s_{rb}}-\dot{z}_{u_{rb}})+\mu_b W_b(\dot{z}_{s_{lb}}-\dot{z}_{u_{lb}})\\
&-k_f W_f(z_{s_{rf}}-z_{u_{rf}})+k_f W_f(z_{s_{lf}}-z_{u_{lf}})-k_b W_b(z_{s_{rb}}-z_{u_{rb}})+k_b W_b(z_{s_{lb}}-z_{u_{lb}})\\
I_y\ddot{\tilde{\theta}}_y =& -\mu_f L_f(\dot{z}_{s_{rf}}-\dot{z}_{u_{rf}})-\mu_f L_f(\dot{z}_{s_{lf}}-\dot{z}_{u_{lf}})+\mu_b L_b(\dot{z}_{s_{rb}}-\dot{z}_{u_{rb}})+\mu_b L_b(\dot{z}_{s_{lb}}-\dot{z}_{u_{lb}})\\
&-k_f L_f(z_{s_{rf}}-z_{u_{rf}})-k_f L_f(z_{s_{lf}}-z_{u_{lf}})+k_b L_b(z_{s_{rb}}-z_{u_{rb}})+k_b L_b(z_{s_{lb}}-z_{u_{lb}})\\
m_{u_f}\ddot{z}_{u_{rf}} =& \mu_f(\dot{z}_{s_{rf}}-\dot{z}_{u_{rf}})+k_f(z_{s_{rf}}-z_{u_{rf}})+k_{t_f}(z_{rf}-z_{u_{rf}})\\
m_{u_f}\ddot{z}_{u_{lf}} =& \mu_f(\dot{z}_{s_{lf}}-\dot{z}_{u_{lf}})+k_f(z_{s_{lf}}-z_{u_{lf}})+k_{t_f}(z_{lf}-z_{u_{lf}})\\
m_{u_b}\ddot{z}_{u_{rb}} =& \mu_b(\dot{z}_{s_{rb}}-\dot{z}_{u_{rb}})+k_b(z_{s_{rb}}-z_{u_{rb}})+k_{t_b}(z_{rb}-z_{u_{rb}})\\
m_{u_b}\ddot{z}_{u_{lb}} =& \mu_b(\dot{z}_{s_{lb}}-\dot{z}_{u_{lb}})+k_b(z_{s_{lb}}-z_{u_{lb}})+k_{t_b}(z_{lb}-z_{u_{lb}})
\end{aligned}
\tag{9}
$$

where $z_{s_{rf}} = W_f\tilde{\theta}_x + L_f\tilde{\theta}_y + \tilde{z}, z_{s_{lf}} = -W_f\tilde{\theta}_x + L_f\tilde{\theta}_y + \tilde{z}, z_{s_{rb}} = W_b\tilde{\theta}_x - L_b\tilde{\theta}_y + \tilde{z}, z_{s_{lb}} = -W_b\tilde{\theta}_x - L_b\tilde{\theta}_y + \tilde{z}$, and z_{rf}, z_{rb}, z_{lf}, z_{lb} are the heights of the vehicle's wheels on the terrain surface. The definitions of the constants in Equations (8) and (9) are shown in Table 2. The values of these constants are obtained by referring to the manufacturer's specifications of the vehicle.

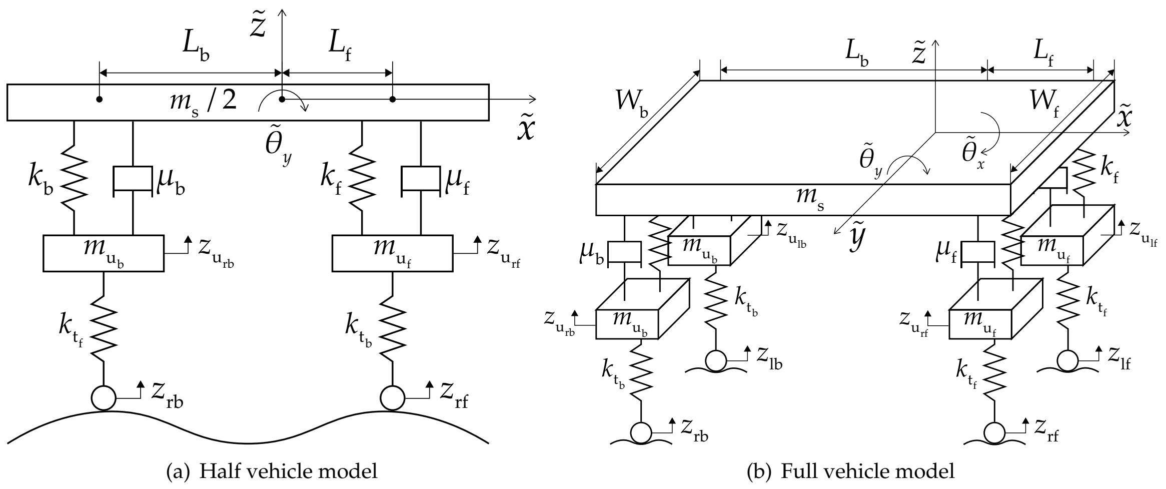


Figure 4. The models of vehicle suspension systems.

Table 2. The definitions of the constants used in the vehicle suspension models.

Notation	**Definition**
m_s	Mass of sprung
m_{u_f}	Mass of front unsprung
m_{u_b}	Mass of back unsprung
I_x	Roll axis moment of inertia
I_y	Pitch axis moment of inertia
k_{t_f}	Stiffness of front tire
k_{t_b}	Stiffness of back tire
k_f	Front suspension spring stiffness
k_b	Back suspension spring stiffness
μ_f	Front suspension damping
μ_b	Back suspension damping
W_f	Width of front sprung
W_b	Width of back sprung
L_f	Length between vehicle front axle and center of gravity of sprung
L_b	Length between vehicle back axle and center of gravity of sprung

Note that during the generation of the initial path, the vehicle is assumed to be static and all the first-order and second-order derivatives in Equations (8) and (9) are equal to zero. Let $\boldsymbol{x} = [\tilde{z}\ \tilde{\theta}_x\ \tilde{\theta}_y\ z_{\mathrm{u_{rf}}}\ z_{\mathrm{u_{lf}}}\ z_{\mathrm{u_{rb}}}\ z_{\mathrm{u_{lb}}}]^{\mathrm{T}}$ denote the *state vector* and $\boldsymbol{w} = [z_{\mathrm{rf}}\ z_{\mathrm{rb}}\ z_{\mathrm{lf}}\ z_{\mathrm{lb}}]^{\mathrm{T}}$ denote the *disturbance vector*. Then, Equation (9) can be written as the following state-transition equation: $\mathbf{0} = \mathbf{A}\boldsymbol{x} + \mathbf{E}\boldsymbol{w}$, where $\mathbf{A}$ and $\mathbf{E}$ are fixed matrices called *state-transition matrix* and *disturbance matrix*, respectively. The solution of this equation is $\boldsymbol{x} = -\mathbf{A}^{-1}\mathbf{E}\boldsymbol{w}$, which can be considered as a mapping $f_{\mathrm{full}} : [z_{\mathrm{rf}}\ z_{\mathrm{rb}}\ z_{\mathrm{lf}}\ z_{\mathrm{lb}}]^{\mathrm{T}} \rightarrow [\tilde{z}\ \tilde{\theta}_x\ \tilde{\theta}_y]^{\mathrm{T}}$. For the half vehicle model, the mapping $f_{\mathrm{half}} : [z_{\mathrm{rf}}\ z_{\mathrm{rb}}]^{\mathrm{T}} \rightarrow [\tilde{z}\ \tilde{\theta}_y]^{\mathrm{T}}$ can be obtained in a similar way. Finally, the SE(3) pose ($\tilde{\boldsymbol{s}}$) of the vehicle is calculated as:

$$\tilde{\boldsymbol{s}} = \left[\tilde{x}\ \tilde{y}\ \tilde{z}\ \tilde{\theta}_x\ \tilde{\theta}_y\ \tilde{\theta}_z\right]^{\mathrm{T}} \quad \text{where} \quad [\tilde{x}\ \tilde{y}\ \tilde{\theta}_z]^{\mathrm{T}} = \bar{s} = [\bar{x}\ \bar{y}\ \bar{\theta}_z]^{\mathrm{T}}$$
$$\text{and} \quad \begin{cases} [\tilde{z}\ \tilde{\theta}_y]^{\mathrm{T}} = f_{\mathrm{half}}([z_{\mathrm{rf}}\ z_{\mathrm{rb}}]^{\mathrm{T}}),\ \tilde{\theta}_x = \theta_x^* & \text{if } |\theta_x^*| \leqslant \theta_x^+, \\ [\tilde{z}\ \tilde{\theta}_x\ \tilde{\theta}_y]^{\mathrm{T}} = f_{\mathrm{full}}([z_{\mathrm{rf}}\ z_{\mathrm{rb}}\ z_{\mathrm{lf}}\ z_{\mathrm{lb}}]^{\mathrm{T}}) & \text{otherwise.} \end{cases} \tag{10}$$

If the roll is nearly zero, the pitch and the z-coordinate are computed using the half vehicle model. Otherwise, the SE(3) pose of the vehicle is computed using the full vehicle model. In summary, given the z-coordinates of the vehicle wheels, the half or full vehicle suspension model will become a linear system of equations with 4 or 7 unknown variables. Then, the roll and the pitch of the vehicle can be calculated via solving this linear system of equations.

3.3. Traversability Computation

In addition to the roll and the pitch calculated by Equation (10), terrain roughness is also necessary in traversability computation. Let $\mathbb{P}_{\mathrm{foot}}$ denote the set of all LiDAR points that are located in the footprint of the vehicle ($\mathbb{P}_{\mathrm{foot}} \subset \mathbb{P}$). Then, the center of gravity ($\boldsymbol{p}_{\mathrm{foot}}$) of the points in $\mathbb{P}_{\mathrm{foot}}$ and the associated covariance matrix are calculated as:

$$\boldsymbol{p}_{\mathrm{foot}} = \frac{1}{n} \sum_{\boldsymbol{p} \in \mathbb{P}_{\mathrm{foot}}} \boldsymbol{p} \tag{11}$$

$$\mathrm{cov}(\boldsymbol{p}_{\mathrm{foot}},\ \boldsymbol{p}_{\mathrm{foot}}) = \frac{1}{n} \sum_{\boldsymbol{p} \in \mathbb{P}_{\mathrm{foot}}} (\boldsymbol{p} - \boldsymbol{p}_{\mathrm{foot}})(\boldsymbol{p} - \boldsymbol{p}_{\mathrm{foot}})^{\mathrm{T}}, \tag{12}$$

where n is the number of points in $\mathbb{P}_{\mathrm{foot}}$. Figure 5 explains how the terrain roughness is computed. The terrain roughness (ρ) is defined as the residual of the fitting plane: $\sqrt{\sum_{j=1}^{n} d_j}$, where d_j represents the distance between ith LiDAR point in $\mathbb{P}_{\mathrm{foot}}$ and the fitting plane. Let λ_{min} be the minimum of the eigenvalues of $\mathrm{cov}(\boldsymbol{p}_{\mathrm{foot}},\ \boldsymbol{p}_{\mathrm{foot}})$. Then, the residual of the fitting plane is:

$$\rho = \sqrt{\lambda_{\mathrm{min}}} \tag{13}$$

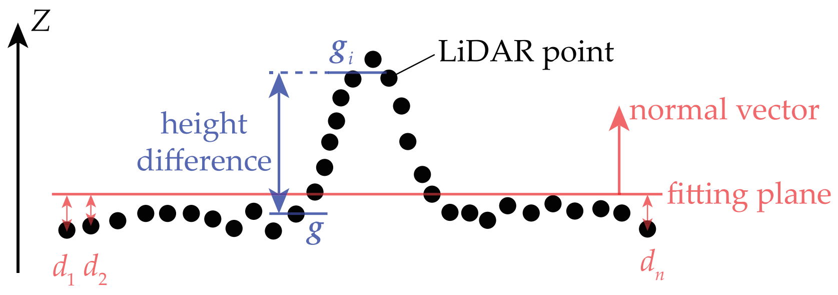


Figure 5. How are the terrain roughness and the height difference computed.

Besides, the height difference need to be computed. As shown in Figure 5, let $\boldsymbol{g}$ denote the grid in which the point $[\widetilde{x}\ \ \widetilde{y}]^{\mathrm{T}}$ located, and let $\mathbb{P}_g$ denote the set of all LiDAR points located in $\boldsymbol{g}$ ($\mathbb{P}_g \subset \mathbb{P}$). Then, the height of $\boldsymbol{g}$ can be calculated as:

$$h_g = \frac{1}{n} \sum_{p \in \mathbb{P}_g} \boldsymbol{p}.z \tag{14}$$

where $\boldsymbol{p}.z$ is the z-coordinate of $\boldsymbol{p}$. Let $\boldsymbol{g}_0,\ \boldsymbol{g}_1,\ \cdots,\ \boldsymbol{g}_7$ denote the eight-connected grids of $\boldsymbol{g}$. Then, the height difference (h_{d}) is defined as: $|h_g - h_{g_i}|$, where $i = \lfloor[(\widetilde{\theta}_z + 22.5) \mod 360]/45\rfloor$ ($\widetilde{\theta}_z \in [0,\ 360)$ and $i \in [0,\ 7]$). The symbols "$\lfloor\ \rfloor$" and "mod" represent the round-down operator and the modulo operator, respectively. In fact, $\boldsymbol{g}_i$ is the grid that is nearest to $\boldsymbol{g}$ in the heading direction of the vehicle.

Finally, the traversability is calculated as:

$$\tau = \begin{cases} 0, & \text{if } |\widetilde{\theta}_x| > \widetilde{\theta}_{x\max} \text{ or } \widetilde{\theta}_y < \widetilde{\theta}_{y\min} \text{ or } \widetilde{\theta}_y > \widetilde{\theta}_{y\max} \text{ or } \rho > \rho_{\max} \text{ or } h_{\mathrm{d}} > h_{\mathrm{d}\max} \\ 1 - \left[w_{\tau_1} \max\left(\dfrac{\widetilde{\theta}_y}{\widetilde{\theta}_{y\min}},\ \dfrac{\widetilde{\theta}_y}{\widetilde{\theta}_{y\max}} \right) + w_{\tau_2} \dfrac{|\widetilde{\theta}_x|}{\widetilde{\theta}_{x\max}} + w_{\tau_3} \dfrac{\rho}{\rho_{\max}} + w_{\tau_4} \dfrac{h_{\mathrm{d}}}{h_{\mathrm{d}\max}} \right], & \text{otherwise} \end{cases} \tag{15}$$

Note that the traversability (Equation (15)) is defined artificially. It is a relative value ranged from 0 to 1. In this work, the definition is that if one of the roll ($\widetilde{\theta}_x$), pitch ($\widetilde{\theta}_y$), terrain roughness (ρ), or height difference (h_{d}) exceeds the respective limit value, the traversability is 0. Otherwise, the traversability is the weighted sum of them, normalized by their respective limit values. Note that a vehicle is usually not front-back symmetrical, so there are two different limit values ($\widetilde{\theta}_{y\min}$ and $\widetilde{\theta}_{y\max}$) for the pitch.

4. Trajectory Planning

In this section, an initial path will first be generated, which is subject to the non-holonomic constraints of a car-like vehicle. Then, this path will be converted into a trajectory, which will be optimized in terms of safety, traversability, time consumption, trajectory smoothness, and so on. Note that the proposed planner and optimizer are called on demand, so the frequency of planning and optimization is not fixed. For example, if the solution trajectory is blocked by a new sensed obstacle, then the planner and optimizer will be called to generate a new trajectory. The overview of this section is shown in Figure 6. The proposed trajectory planner is based on Terrain Shapes and vehicle Suspension models, so it is called TS-Sus.

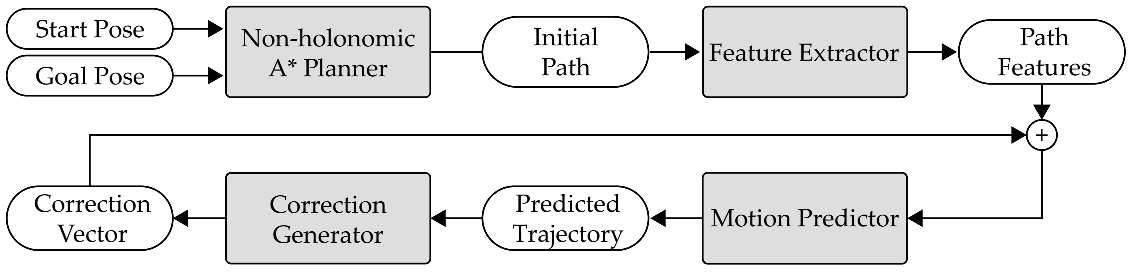


Figure 6. Overview of the trajectory planning approach.

*4.1. Non-Holonomic A**

For a given planning query (a start pose and a goal pose in the SE(2) space), an initial path is first found by a non-holonomic A* planner, which takes into account the non-holonomic constraints, the proximity to an obstacle, and the traversability. Next, the non-holonomic constraints of a car-like vehicle will be introduced, and then the node expansion and the cost function of the non-holonomic A* will be described.

4.1.1. Non-Holonomic Constraints

For a mechanical system, kinematic constraints are described by the relations between the position and the velocity of the system. The kinematic constraints that cannot be integrated to the form containing only the position are called non-holonomic constraints. A system subject to non-holonomic constraints is called a non-holonomic system. A car-like vehicle is a non-holonomic system. It cannot move sideways, and its turning radius is lower bounded. Formally, the non-holonomic constraints that a car-like vehicle (shown in Figure 7) satisfies are written as:

$$\begin{bmatrix} \dot{\overline{x}} \\ \dot{\overline{y}} \\ \dot{\overline{\theta}}_z \end{bmatrix} = \begin{bmatrix} v \cdot \sin \overline{\theta}_z \\ v \cdot \cos \overline{\theta}_z \\ (v \cdot \tan \psi) \;/\; L' \end{bmatrix}, \tag{16}$$

where $[\overline{x} \;\; \overline{y} \;\; \overline{\theta}_z]^{\mathrm{T}}$ is the SE(2) pose of the vehicle, and v is the longitudinal velocity of the vehicle. ψ is the steering angle, and L' is the wheelbase.

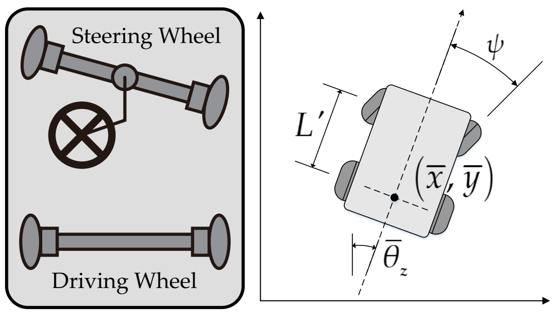
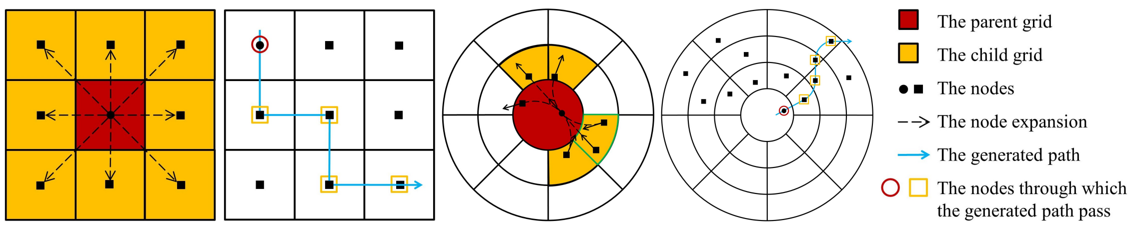


Figure 7. A car-like vehicle and its kinematic model.

4.1.2. Node Expansion

In this section, a non-holonomic A* search is performed to find a path satisfying the non-holonomic constraints. Figure 8 shows the differences between the conventional A* and the non-holonomic A*. On the one hand, the conventional A* treats a car-like vehicle as a point without orientation and only visits grid centers, as shown in Figure 8a. As a result, the path generated by the conventional A* is piecewise-linear, which is a sequence of vehicle positions $[\overline{x}_i \;\; \overline{y}_i]^{\mathrm{T}}$, $i = 1, 2, \cdots, n$ (as shown Figure 8b). On the other hand, the non-holonomic A* considers a car-like vehicle as a vector. The child poses are generated by assuming some different steering angles and a fixed forward/reverse velocity in Equation (16). This method associates vehicle poses with grids, so any continuous pose of the vehicle can be visited. Therefore, the resultant path satisfies the non-holonomic constraints, as shown in Figure 8b. Mathematically, the path generated by the non-holonomic A* is a sequence of SE(2) poses $\overline{\mathbf{s}}_i = [\overline{x}_i \;\; \overline{y}_i \;\; \overline{\theta}_{z_i}]^{\mathrm{T}}$, $i = 1, 2, \cdots, n$, which can be mapped to SE(3) poses using the pose estimation method introduced in Section 3.2.



(a) The node expansion of the conventional A* (b) The node expansion of the non-holonomic A* (c) The path generated by the conventional A* (d) The path generated by the non-holonomic A*

Figure 8. The differences between the conventional A* and the non-holonomic A*.

The shapes of the grids in the conventional A* and the non-holonomic A* are also different. The conventional A* planner and the non-holonomic A* planner search square-shaped grids and sector-shaped grids, respectively. The sector-shaped grids can well represent the characteristics of point cloud data (high resolution for the LiDAR points near the vehicle, and low resolution for the points that are far from the vehicle).

4.1.3. Cost Function

The order of the node expansions in the non-holonomic A* is partly determined by *movement costs*. There are different costs for different types of movements. For example, moving on the uneven ground leads to a greater cost than moving on flat ground, and the cost of moving reverse is greater than that of moving forward. In this work, the cost (f_{m}) of moving from a parent pose ($\widetilde{\boldsymbol{s}}_i \in \mathrm{SE(3)}$) to a child pose ($\widetilde{\boldsymbol{s}}_{i+1} \in \mathrm{SE(3)}$) is computed as:

$$\begin{aligned} f_{\mathrm{m}}\left(\widetilde{\boldsymbol{s}}_{i+1}, \widetilde{\boldsymbol{s}}_i\right) = {} & w_{\mathrm{m}_1} \epsilon_{\mathrm{reverse}} \frac{\mathrm{length}(\widetilde{\boldsymbol{s}}_{i+1}, \widetilde{\boldsymbol{s}}_i)}{N_{\mathrm{m}_1}} + w_{\mathrm{m}_2} \frac{\epsilon_{\mathrm{switch}}}{N_{\mathrm{m}_2}} \\ & + w_{\mathrm{m}_3} \frac{\max(0, d_{\mathrm{omax}} - d_{\mathrm{o}}^{i+1})}{N_{\mathrm{m}_3}} + w_{\mathrm{m}_4} \frac{1 - \tau_{i+1}}{N_{\mathrm{m}_4}}, \end{aligned} \quad d_{\mathrm{o}}^{i+1} \neq 0 \text{ and } \tau_{i+1} \neq 0 \text{ and } \|\widetilde{\boldsymbol{s}}_{i+1} - \widetilde{\boldsymbol{s}}_i\| \leqslant \xi \tag{17}$$

where w_{m_i} $(i = 1, 2, 3, 4)$ is a weight factor (determined empirically) that indicates the importance of the respective cost, and N_{m_i} $(i = 1, 2, 3, 4)$ is a normalization factor that is determined as the largest value of the respective cost. d_{o}^{i+1} is the distance between $\widetilde{\boldsymbol{s}}_{i+1}$ and the obstacle nearest to $\widetilde{\boldsymbol{s}}_{i+1}$. d_{omax}, called *safe distance*, is determined according to the vehicle size and the environment. τ_{i+1} is the traversability of $\widetilde{\boldsymbol{s}}_{i+1}$. ξ is the resolution of the path generated by the non-holonomic A* planner. Besides, $\epsilon_{\mathrm{reverse}}$ and $\epsilon_{\mathrm{switch}}$ are computed as:

$$\epsilon_{\mathrm{reverse}} = 1 + \delta\left(1 + \mathrm{sgn}(v_{i+1})\right) P_{\mathrm{reverse}} \qquad \epsilon_{\mathrm{switch}} = \delta\left(2 - |\mathrm{sgn}(v_{i+1}) - \mathrm{sgn}(v_i)|\right) P_{\mathrm{switch}} \tag{18}$$

where

$$\delta(u) = \begin{cases} 0, & u \neq 0 \\ 1, & u = 0 \end{cases} \qquad \mathrm{sgn}(v) = \begin{cases} -1, & v < 0 \\ 0, & v = 0 \\ 1, & v > 0 \end{cases} \tag{19}$$

and P_{reverse}, P_{switch} are the multiplicative penalty applied to driving reverse and the additive penalty applied to switching direction, respectively. Finally, $\mathrm{length}(\widetilde{\boldsymbol{s}}_{i+1}, \widetilde{\boldsymbol{s}}_i)$ is the length of the path segment connecting the parent pose ($\widetilde{\boldsymbol{s}}_i$) and the child pose ($\widetilde{\boldsymbol{s}}_{i+1}$). It can be calculated as:

$$\mathrm{length}(\widetilde{\boldsymbol{s}}_{i+1}, \widetilde{\boldsymbol{s}}_i) = \begin{cases} \Delta\widetilde{\theta}_z r_{\mathrm{turn}}, & \Delta\widetilde{\theta}_z \neq 0 \\ (\Delta\widetilde{x}^2 + \Delta\widetilde{y}^2 + \Delta\widetilde{z}^2)^{1/2}, & \Delta\widetilde{\theta}_z = 0 \end{cases} \tag{20}$$

where r_{turn} is the maximal turning radius of the vehicle. $\Delta\widetilde{x}$ is computed as $\widetilde{x}_{i+1} - \widetilde{x}_i$. $\Delta\widetilde{y}$, $\Delta\widetilde{z}$, and $\Delta\widetilde{\theta}_z$ are all computed in a similar way. During the non-holonomic A* search, the cost (f_g, called *cost-to-come*) of moving from the start pose ($\widetilde{\boldsymbol{s}}_0$) to the current pose ($\widetilde{\boldsymbol{s}}_i$) can be calculated as:

$$f_g\left(\widetilde{\boldsymbol{s}}_i\right) = f_{\mathrm{m}}\left(\widetilde{\boldsymbol{s}}_i, \widetilde{\boldsymbol{s}}_{i-1}\right) + f_g\left(\widetilde{\boldsymbol{s}}_{i-1}\right) = f_{\mathrm{m}}\left(\widetilde{\boldsymbol{s}}_i, \widetilde{\boldsymbol{s}}_{i-1}\right) + f_{\mathrm{m}}\left(\widetilde{\boldsymbol{s}}_{i-1}, \widetilde{\boldsymbol{s}}_{i-2}\right) + \cdots + f_{\mathrm{m}}\left(\widetilde{\boldsymbol{s}}_1, \widetilde{\boldsymbol{s}}_0\right). \tag{21}$$

Note that the pose whose d_{o} or τ equals 0 is not considered in Equation (17), because it is the non-traversable pose that is not expanded by the A* search. The computational complexity of A* is well known: $O(b^d)$, where b is the branching factor of the A* search tree, and d is the depth of the goal node.

4.2. Trajectory Optimization

The non-holonomic A* only accounts for the kinematic model of the ground vehicle. However, the pose of the vehicle is also largely affected by its suspension model in a non-planar environment. Besides, the initial path produced by the non-holonomic A* is usually not smooth enough and worthy of further improvement. Next, the features of the initial path will first be extracted and the trajectory of the vehicle will be predicted. This prediction is based on the extracted features and the suspension model of the vehicle. Then, the predicted trajectory corresponding to the features will be optimized in terms of smoothness, traversability, and so on.

4.2.1. Feature Extraction

The initial path is a sequence of SE(3) poses, which is a high-dimensional vector and hard to be optimized directly. So the features of this path are first extracted to descend its dimension. The feature vector ($\boldsymbol{u}$) is defined as:

$$\boldsymbol{u} = [\boldsymbol{u}_v^{\mathrm{T}} \ \boldsymbol{u}_{\_z}^{\mathrm{T}}]^{\mathrm{T}}, \quad \boldsymbol{u}_v = [v_0 \ a_0 \ v_{\max} \ a_{\mathrm{f}} \ v_{\mathrm{f}} \ t_{\mathrm{f}}]^{\mathrm{T}} \text{ and } \boldsymbol{u}_{\omega_z} = [\omega_z(0) \ \omega_z(\Delta t) \ \omega_z(2\Delta t) \ \cdots \ \omega_z(t_{\mathrm{f}})]^{\mathrm{T}} \quad (22)$$

where $\boldsymbol{u}_v$ determines the parameters of the trapezoid shown in Figure 9, and $\boldsymbol{u}_{\omega_z}$ determines the knot-points of the spline curve shown in Figure 10. v_0, a_0, $v_{\max}$, a_{f}, and v_{f} are the start velocity, the start acceleration, the maximal velocity (or called the traverse velocity), the terminal acceleration, and the terminal velocity. $\omega_z(t)$ is the angular velocity of the vehicle at time instant t.

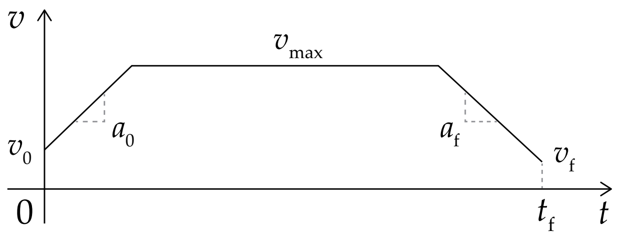


Figure 9. The trapezoidal profile of the longitudinal velocity.

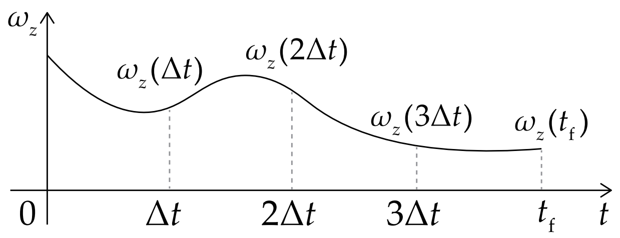


Figure 10. The spline profile of the angular velocity.

The initial path produced by the non-holonomic A* contains no information about the velocity and the acceleration of the vehicle, so v_0, a_0, $v_{\max}$, a_{f} and v_{f} are set manually according to the limitation of the vehicle. t_{f} is the time spent on moving from the start pose to the goal pose, and it can be calculated as:

$$t_{\mathrm{f}} = \left[l_{\mathrm{f}} + \frac{(v_{\max} - v_0)^2}{2a_0} - \frac{(v_{\max} - v_{\mathrm{f}})^2}{2a_{\mathrm{f}}} \right] \Big/ v_{\max} \quad (23)$$

where l_f is the length of the initial path. Once $\boldsymbol{u}_v$ is known, the mapping from the time instant (t) to the traveled distance of the vehicle (l) can be obtained via integration. Besides, for a given initial path, the mapping from l to the yaw of the vehicle ($\widetilde{\theta}_z$) can be obtained via interpolation. Finally, the mapping from t to $\widetilde{\theta}_z$ ($t \in [0,\ t_\mathrm{f}]$) can be obtained. Let K be the dimension of $\boldsymbol{u}_{\omega_z}$ and $\Delta t = t_\mathrm{f}/(K-1)$. Then, $\boldsymbol{u}_{\omega_z}$ can be calculated approximately using forward differences:

$$\omega_z(t) = \frac{\mathrm{d}\widetilde{\theta}_z}{\mathrm{d}t} \approx \frac{\widetilde{\theta}_z(t+e) - \widetilde{\theta}_z(t)}{e} \tag{24}$$

4.2.2. Motion Prediction

Before the feature vector is optimized, it should first be mapped to a vehicle trajectory by a motion prediction method, which contains two steps: motion simulation and pose estimation. Note that in the motion prediction the vehicle is not assumed to be static. This means that the derivatives in Equation (9) can be non-zero, so the pose estimation in the motion prediction is different from that in Section 3.2.

The motion simulation works as follows. Given the SE(3) pose of the vehicle at the current time instant t_i ($t_i \in [0,\ t_\mathrm{f}]$) and a feature vector ($\boldsymbol{u}$), Euler's method is used to calculate the planar position and the yaw at the next time instant t_{i+1}. This calculation is based on the following vehicle motion model:

$$\begin{bmatrix} \dot{\widetilde{x}} \\ \dot{\widetilde{y}} \\ \dot{\widetilde{\theta}}_z \end{bmatrix} = \begin{bmatrix} \cos\widetilde{\theta}_z \cos\widetilde{\theta}_y & \cos\widetilde{\theta}_z \sin\widetilde{\theta}_y \sin\widetilde{\theta}_x - \sin\widetilde{\theta}_z \cos\widetilde{\theta}_x & 0 \\ \sin\widetilde{\theta}_z \cos\widetilde{\theta}_y & \sin\widetilde{\theta}_z \sin\widetilde{\theta}_y \sin\widetilde{\theta}_x + \cos\widetilde{\theta}_z \cos\widetilde{\theta}_x & 0 \\ 0 & 0 & \dfrac{\cos\widetilde{\theta}_x}{\cos\widetilde{\theta}_y} \end{bmatrix} \begin{bmatrix} v \\ 0 \\ \omega_z \end{bmatrix}. \tag{25}$$

The pose estimation works as follows. Given the planar position and the yaw at t_{i+1}, Equations (2)–(4) are used to compute the heights of the four vehicle wheels (z_rf, z_rb, z_lf, z_lb), which are the disturbance inputs of the vehicle's suspension model (shown in Equation (9)). Then, Euler's method is used again to compute the height ($\widetilde{z}$), the roll ($\widetilde{\theta}_x$), and the pitch ($\widetilde{\theta}_y$) of the vehicle at t_{i+1} based on Equation (9). A vehicle trajectory, which is a sequence of SE(3) vehicle poses, is generated when t_i is equal to t_f. Note that this vehicle trajectory is different from the initial path produced by the non-holonomic A*. The poses along the trajectory are all time-indexed, and the trajectory is generated without assuming that the vehicle is static.

4.2.3. Numerical Optimization

To generate an optimal trajectory, the optimizer must adjust the feature vector ($\boldsymbol{u}$) to satisfy the constraint: $\Delta\widetilde{\boldsymbol{s}}_\mathrm{f}(\boldsymbol{u}) = \widetilde{\boldsymbol{s}}_\mathrm{f} - \widetilde{\boldsymbol{s}}(t_\mathrm{f}) = 0$, and minimize a cost function ($f_t(\boldsymbol{u})$), where $\widetilde{\boldsymbol{s}}_\mathrm{f}$ is the goal pose and $\widetilde{\boldsymbol{s}}(t_\mathrm{f})$ is the terminal pose of the trajectory generated by the motion prediction. This is a constrained optimization problem, which can be solved using the *method of Lagrange multipliers*. The basic idea is to convert a constrained problem into a form such that the derivative test of an unconstrained problem can be applied. The Lagrange function ($\mathcal{L}$) is written as:

$$\mathcal{L}(\boldsymbol{u},\ \lambda) = f_t(\boldsymbol{u}) + \lambda^\mathrm{T}\Delta\widetilde{\boldsymbol{s}}_\mathrm{f}(\boldsymbol{u}), \tag{26}$$

where λ is the Lagrange multiplier vector. The necessary conditions for optimality are:

$$\nabla_{\boldsymbol{u},\lambda}\mathcal{L}(\boldsymbol{u},\ \lambda) = 0 \quad \Longleftrightarrow \quad \begin{cases} \dfrac{\partial\mathcal{L}(\boldsymbol{u},\ \lambda)}{\partial\boldsymbol{u}} = \dfrac{\partial f_t(\boldsymbol{u})}{\partial\boldsymbol{u}} + \lambda^\mathrm{T}\dfrac{\partial\Delta\widetilde{\boldsymbol{s}}_\mathrm{f}(\boldsymbol{u})}{\partial\boldsymbol{u}} = \mathbf{0}^\mathrm{T} \\ \\ \dfrac{\partial\mathcal{L}(\boldsymbol{u},\ \lambda)}{\partial\lambda} = \Delta\widetilde{\boldsymbol{s}}_\mathrm{f}(\boldsymbol{u}) = \mathbf{0} \end{cases}. \tag{27}$$

Equation (27) can be solved using Newton's method. The feature vector and the Lagrange multiplier vector are adjusted iteratively until a sufficiently precise value is reached. According to Newton's method, a correction vector for $\boldsymbol{u}$ and λ at each iteration of the optimization is computed as:

$$\begin{bmatrix} \Delta \boldsymbol{u} \\ \Delta \lambda \end{bmatrix} = -\beta \mathbf{H}^{-1} \mathbf{J} = -\beta \begin{bmatrix} \dfrac{\partial^2 \mathcal{L}(\boldsymbol{u}, \lambda)}{\partial \boldsymbol{u}^2} & \dfrac{\partial \Delta \widetilde{\boldsymbol{s}}_{\mathrm{f}}(\boldsymbol{u})}{\partial \boldsymbol{u}}^{\mathrm{T}} \\ \dfrac{\partial \Delta \widetilde{\boldsymbol{s}}_{\mathrm{f}}(\boldsymbol{u})}{\partial \boldsymbol{u}} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \dfrac{\partial \mathcal{L}(\boldsymbol{u}, \lambda)}{\partial \boldsymbol{u}}^{\mathrm{T}} \\ \Delta \widetilde{\boldsymbol{s}}_{\mathrm{f}}(\boldsymbol{u}) \end{bmatrix}, \tag{28}$$

where $\mathbf{H}$ and $\mathbf{J}$ are the Hessian matrix and the Jacobian matrix of the Lagrange function. A step size scaling factor (β) is used to improve numerical stability. In the implementation, forward differences are used to estimate the partial derivatives:

$$\frac{\partial \Delta \widetilde{s}_i(\boldsymbol{u})}{\partial u_j} \approx \frac{\Delta \widetilde{s}_i(u_1, u_2, \cdots, u_j + e, \cdots, u_n) - \Delta \widetilde{s}_i(\boldsymbol{u})}{e} \tag{29}$$

$$\begin{aligned} \frac{\partial^2 \mathcal{L}(\boldsymbol{u})}{\partial u_k \partial u_l} \approx \frac{\Delta^2 \mathcal{L}}{e^2}, \quad \Delta^2 \mathcal{L} = {} & \mathcal{L}(u_1, u_2, \cdots, u_k + e, \cdots, u_l + e, \cdots, u_n) \\ & - \mathcal{L}(u_1, u_2, \cdots, u_k + e, \cdots, u_n) \\ & - \mathcal{L}(u_1, u_2, \cdots, u_l + e, \cdots, u_n) + \mathcal{L}(\boldsymbol{u}). \end{aligned} \tag{30}$$

where $\Delta \widetilde{s}_i$ is the ith element of $\Delta \widetilde{\boldsymbol{s}}_{\mathrm{f}}$, and u_j is the jth element of $\boldsymbol{u}$.

Note that the cost function ($f_{\boldsymbol{t}}(\boldsymbol{u})$) used in the trajectory optimization is different from that used in the non-holonomic A*. The former is used to compute the cost of a trajectory corresponding to $\boldsymbol{u}$. The poses along a trajectory are time-indexed, so $f_{\boldsymbol{t}}(\boldsymbol{u})$ often accounts for more optimization criteria such as velocity, acceleration and time consumption. Next, the optimization criteria that are considered by $f_{\boldsymbol{t}}(\boldsymbol{u})$ will be introduced.

Firstly, the proximity (d_{o}) of the trajectory to an obstacle and the traversability (τ) of the trajectory are included. The cost functions corresponding to them are written as:

$$f_{d_{\mathrm{o}}}(\boldsymbol{u}) := f_{d_{\mathrm{o}}}(f_{\mathrm{mp}}(\boldsymbol{u})) = f_{d_{\mathrm{o}}}(\boldsymbol{t}) = \frac{1}{n} \sum_{i=1}^{n} \max(0,\, d_{\mathrm{omax}} - d_{\mathrm{o}}^{i}) \qquad f_{\tau}(\boldsymbol{u}) = \frac{1}{n} \sum_{i=1}^{n} 1 - \tau_i \tag{31}$$

where f_{mp}: $\boldsymbol{u} \rightarrow \boldsymbol{t}$ is the motion prediction, and $\boldsymbol{t}$ is a vehicle trajectory. n is the number of poses along $\boldsymbol{t}$, and the other notations are the same as the notations in Equation (17). For clarity, $\boldsymbol{t}$ will always be replaced by $\boldsymbol{u}$ in the following definitions of cost functions. That is to say, if a cost is calculated based on the properties of a vehicle trajectory, the feature vector ($\boldsymbol{u}$) will always be mapped to a trajectory ($\boldsymbol{t}$) based on f_{mp}.

Secondly, the velocity and the acceleration should be bounded from above and below according to the manufacturer's specifications of the vehicle. For example, the cost function (f_v) corresponding to the longitudinal velocity is defined as:

$$f_v(\boldsymbol{u}) = \frac{1}{n} \sum_{i=1}^{n} \max(0,\, v_{\mathrm{lim}}^{-} - v_i,\, v_i - v_{\mathrm{lim}}^{+}), \quad v_{\mathrm{lim}}^{-} < 0 \text{ and } v_{\mathrm{lim}}^{+} > 0, \tag{32}$$

where v_i is the longitudinal velocity of the vehicle when it is at the ith pose along the trajectory. v_{lim}^{-} and v_{lim}^{+} are the reversing velocity limit and the forward velocity limit of the vehicle, respectively. In a similar way, the cost functions corresponding to the angular velocity, the longitudinal acceleration and the angular acceleration of the vehicle are defined as $f_{\omega_z}(\boldsymbol{u})$, $f_a(\boldsymbol{u})$, and $f_{\alpha_z}(\boldsymbol{u})$, respectively.

Thirdly, the time consumption (t_f) and the length (l_f) of the trajectory should also be included. The cost functions corresponding to them are $f_{t_\mathrm{f}}(\boldsymbol{u}) = t_\mathrm{f}$ and $f_{l_\mathrm{f}}(\boldsymbol{u}) = l_\mathrm{f}$. l_f is computed by Equation (23) when t_f is known. Fourthly, the trajectory should be optimized in terms of smoothness to reduce unnatural swerves. The cost function (f_σ) corresponding to the trajectory smoothness is defined as:

$$f_\sigma(\boldsymbol{u}) = \frac{1}{n-2} \sum_{i=2}^{n-1} ([\tilde{x}_{i+1} \;\; \tilde{y}_{i+1} \;\; \tilde{z}_{i+1}]^\mathrm{T} - 2[\tilde{x}_i \;\; \tilde{y}_i \;\; \tilde{z}_i]^\mathrm{T} + [\tilde{x}_{i-1} \;\; \tilde{y}_{i-1} \;\; \tilde{z}_{i-1}]^\mathrm{T})^2. \tag{33}$$

Note that the z-coordinate is also considered in the calculation of the smoothness cost. In this way, a trajectory that prevents the vehicle from rising and falling frequently can be obtained. Finally, the total cost function ($f_{\boldsymbol{t}}(\boldsymbol{u})$) is computed as a weighted sum of all the above-mentioned cost functions.

The computational complexity of the trajectory optimization is $O(mn)$, where m is the number of optimization iteration, and n is the number of poses along the trajectory that is optimized. Recall that the computational complexity of A* is $O(b^d)$, so the computational complexity of the proposed approach is $O(b^d + mn)$.

5. Experimental Results and Discussion

This section provides a precise description of the experimental results and discusses them. Moreover, the proposed traversability assessor and trajectory planner are analyzed according to these results. The experiments were conducted in both the simulation and the real world, with the same set of parameter values. The half and full vehicle suspension models used in experiments are shown in Figure 4, Equations (8) and (9). The programs of the proposed approaches are executed on a single core of a 3.2 GHz Intel Core i5-3470 processor.

5.1. Simulation Experiments

In the simulation experiments, virtual terrain surfaces were created using a terrain editor called *EarthSculptor*, and the point clouds of these virtual terrain surfaces were generated by a virtual HDL-64E S2 LiDAR in a simulator called Gazebo, which was run on the Robot Operating System (ROS). Besides, a virtual vehicle with the samespecifications as our real vehicle (an all-terrain vehicle) was created in the Gazebo simulator.

Let $\mathbb{S}$ denote the following set: $\{[\overline{x} \;\; \overline{y} \;\; \overline{\theta}_z]^\mathrm{T} \mid \overline{x}/1 \in \mathbb{Z},\ \overline{y}/1 \in \mathbb{Z},\ \overline{\theta}_z/0.1 \in \mathbb{Z},\ \overline{x} \in [0,\ W_\mathrm{m}],\ \overline{y} \in [0,\ L_\mathrm{m}],\ \overline{\theta}_z \in [0,\ 2\pi)\}$, where W_m and L_m are the width and the length of a terrain surface (in meters), respectively. $\mathbb{Z}$ is the set of all integers. For each terrain surface in Figure 11, all the query poses in $\mathbb{S}$ were mapped to SE(3) poses using the proposed pose estimation method. Then, to acquire the ground truths of the vehicle's height, roll, and pitch, the virtual vehicle was launched at each query pose in $\mathbb{S}$, and the corresponding SE(3) pose was acquired via the ROS message published by the Gazebo simulator. Finally, the errors of our pose estimation method (PC-Sus) could be calculated. Similarly, we computed the errors of a pose estimation method [2] (called DEM-Kin) that is based on Digital Elevation Maps and vehicle Kinematic models, and another pose estimation method [45] (called Kin-GP-VE) that is based on Kinematics, Gaussian Processes, and Vehicle Experiences. Table 3 shows these errors and the improvement in pose estimation using the proposed PC-Sus method. The Root Mean Squared Errors (RMSEs) in roll and pitch estimation were reduced by approximately 35% and 47% over the Kin-GP-VE method and the DEM-Kin method, respectively.

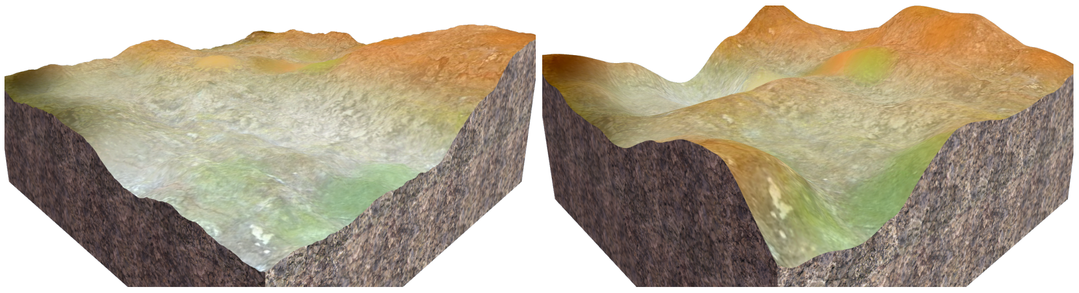


Figure 11. The virtual terrain surfaces used to test the proposed pose estimation method.

Table 3. The errors of the different pose estimation methods.

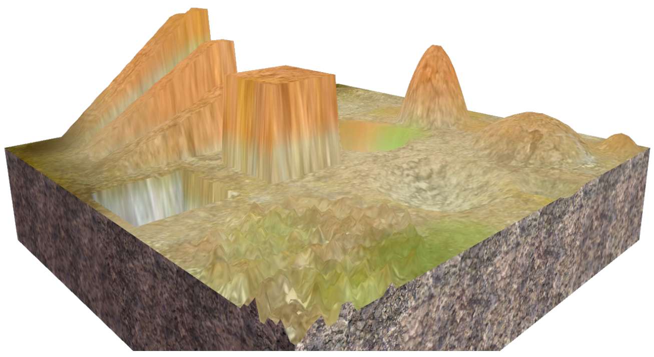
Terrain Surface	Method	RMSE of Roll (Rad)	RMSE of Pitch (Rad)
Figure 11a	DEM-Kin	0.0751	0.0806
	Kin-GP-VE	0.0612	0.0654
	PC-Sus	**0.0389**	**0.0405**
Figure 11b	DEM-Kin	0.0921	0.0984
	Kin-GP-VE	0.0763	0.0817
	PC-Sus	**0.0502**	**0.0535**
Improvement over DEM-Kin		46.85%	47.69%
Improvement over Kin-GP-VE		35.33%	36.30%

Let Θ denote the following set: $\{\bar{\theta}_z \mid \bar{\theta}_z/0.1 \in \mathbb{Z},\ \bar{\theta}_z \in [0,\ 2\pi)\}$. Then, the traversability at $[\bar{x}\ \ \bar{y}]^{\mathrm{T}}$ can be defined as:

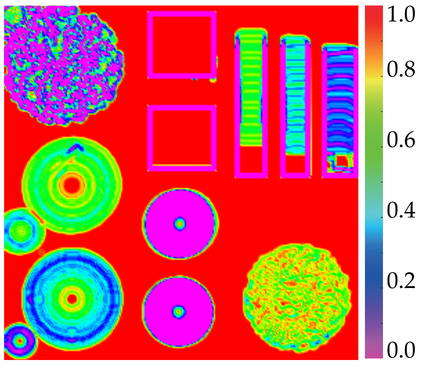
$$\tau_{\overline{xy}} := \max_{\bar{\theta}_z \in \Theta} \tau\left(f_{\mathrm{pe}}\left([\bar{x}\ \ \bar{y}\ \ \bar{\theta}_z]^{\mathrm{T}}\right)\right), \tag{34}$$

where τ is calculated using Equation (15), and f_{pe} is shown in Equation (1). Figure 12 shows the heat map colored by $\tau_{\overline{xy}}$, which is called traversability map. According to the results, the proposed method can evaluate the traversabilities of different terrain shapes, such as ramps, pits, and so on. Note that the traversability maps were constructed only for the purpose of validating the proposed traversability assessor. The traversability map would not be built in trajectory planning. During the process of planning, only the traversabilities of the query poses would be computed.

The proposed trajectory planner was tested on different terrain surfaces. Figure 13 shows the initial paths (red) produced by the non-holonomic A*, the trajectories (green) after optimization, and the real trajectories (blue) of the virtual vehicle. The blue trajectories were generated by making the virtual vehicle track the green trajectories. For clarity, all the above trajectories were drawn on the grayscale traversability maps of the terrain surfaces. Figure 14 shows the z-coordinate, the roll and the pitch of the vehicle along these paths and trajectories. Table 4 shows the performance comparison of these paths and trajectories. Note that d_{o} is $+\infty$ when there is no obstacle in the environments. According to the results, the initial paths had many unnatural swerves (according to Figure 13). Although such paths were drivable, they often led to excessive steering of the vehicle. Compared with the initial paths, the trajectories after optimization were shorter, smoother and had higher traversabilities (according to Figure 14 and Table 4). Therefore, the total cost of the trajectory is reduced after the optimization.

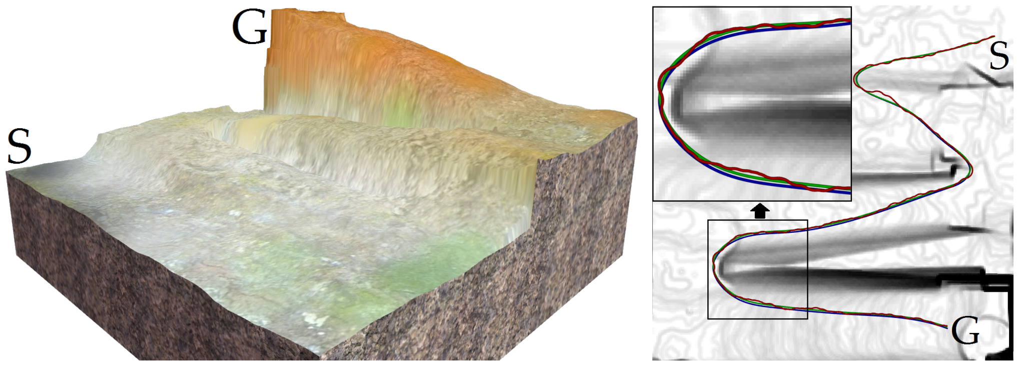


(a) Virtual terrain surface

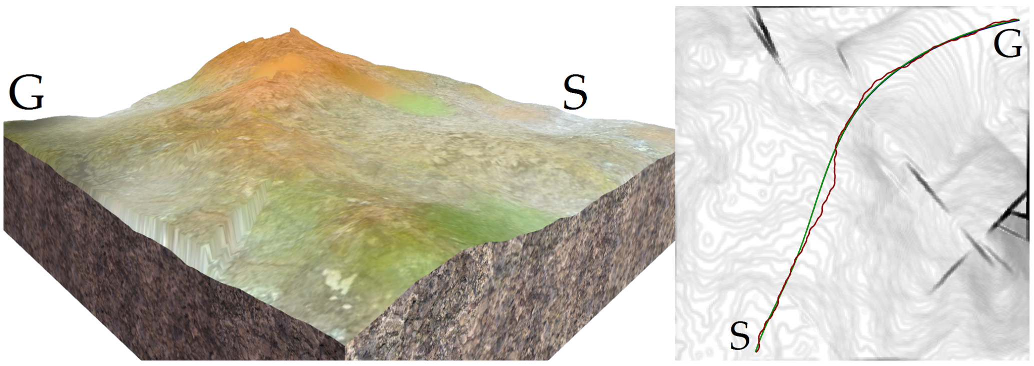


(b) Traversability map colored by $\tau_{\overline{xy}}$

Figure 12. A virtual terrain surface and its traversability map.

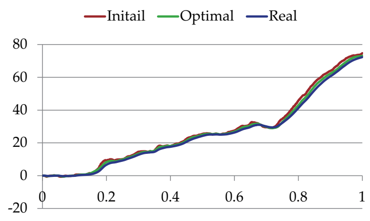


(a) Simulation results in a non-planar environment with steep ramps

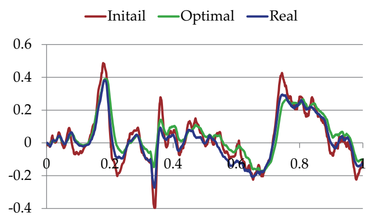


(b) Simulation results in a non-planar environment with gentle ramps

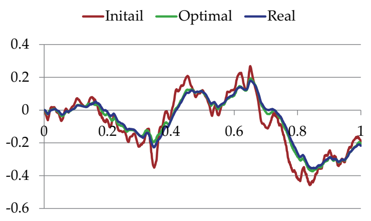
Figure 13. The initial paths (red), the trajectories (green) after optimization, and the real trajectories (blue) of the vehicle on the virtual terrain surfaces. For clarity, the trajectories are drawn on the grayscale traversability maps.



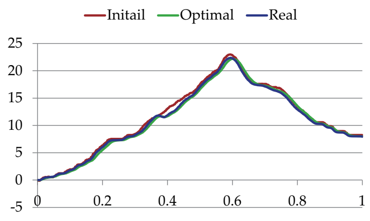
(a) z-coordinate



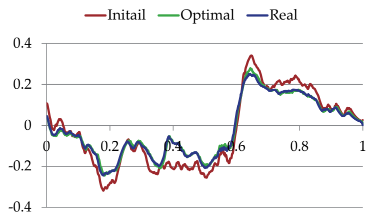
(b) Roll



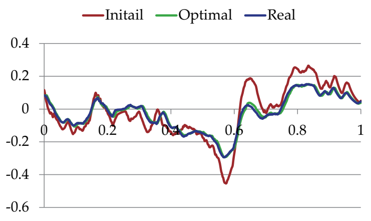
(c) Pitch



(d) z-coordinate



(e) Roll



(f) Pitch

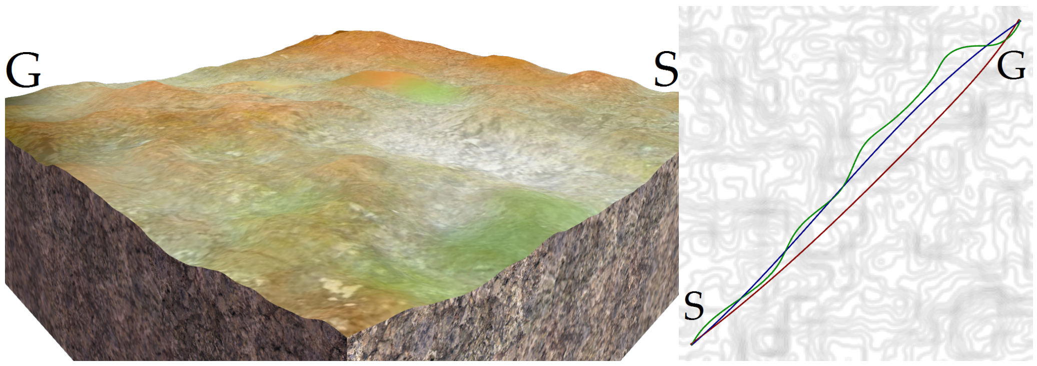
Figure 14. The z-coordinate, the roll and the pitch of the vehicle along the paths and the trajectories. (**a**–**c**) The vehicle state along the trajectories shown in Figure 13a; (**d**–**f**) The vehicle state along the trajectories shown in Figure 13b.

Table 4. The performance comparison of the paths and the trajectories shown in Figure 13. Note that the real trajectories were often different between different runs, so the data of the real trajectories were averaged over 50 runs.

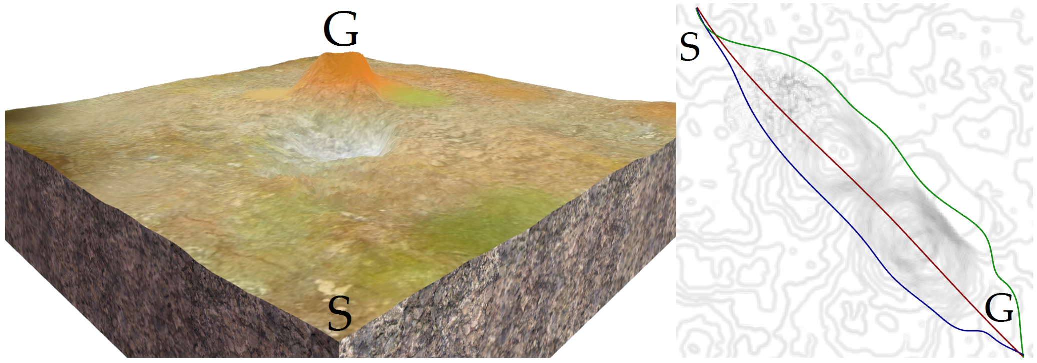
Terrain Surface Shown in Figure 13a						
Path or Trajectory	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**
Initial path	6.09	0.9099	207.09	617.48	0.0111	1.0000
Optimized trajectory	**6.53**	**0.9281**	**200.24**	**596.23**	**1.0000**	**0.2079**
Real trajectory	6.45	0.9267	201.33	598.79	0.9822	0.2190
Terrain Surface Shown in Figure 13b						
Path or Trajectory	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**
Initial path	15.44	0.9329	115.03	340.60	0.4484	1.0000
Optimized trajectory	18.08	**0.9466**	**111.71**	**330.62**	**1.0000**	**0.6461**
Real trajectory	**18.72**	0.9465	111.98	331.88	0.9937	0.6526

Furthermore, Figures 15 and 16, and Table 5 compare the proposed trajectory planner (TS-Sus) with two other state-of-the-art trajectory planners. One [46] (referred to as NoTS) of the two methods does Not account for the Terrain Shape. The other one [26] (referred to as TS-NoSus) takes the Terrain Shape into consideration, but it ignores the Suspension system of the vehicle. In the environment shown in Figure 15a, the TS-NoSus method computed a trajectory that avoided the small terrain undulations. This trajectory was highly traversable (according Figure 16a–c and Table 5) but not smooth (according to Table 5). In fact, the all-terrain vehicle could directly pass through the small terrain undulations without reducing the traversability, due to the shock absorption of its suspension system. The proposed TS-Sus method accounted for the vehicle suspension in the pose estimation, so its trajectory only avoided some large undulations and was smoother than that generated by TS-NoSus.

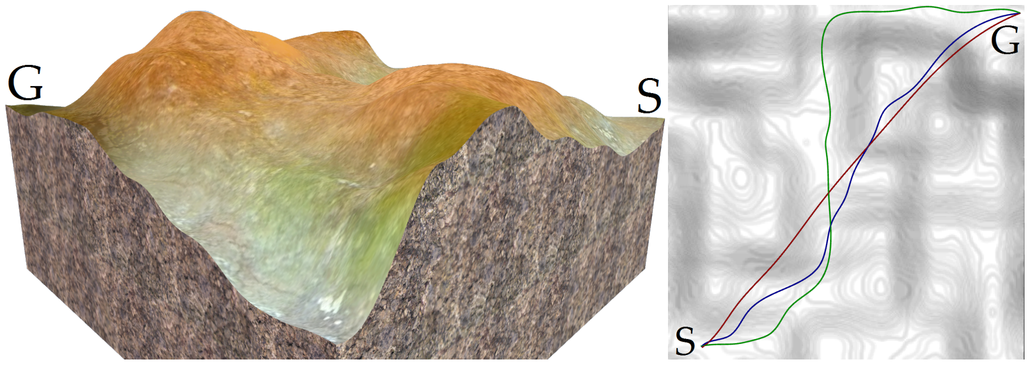
The terrain undulations become larger in the environments shown in Figure 15b,c and there were even non-traversable areas (or called obstacles) in the latter environment. The NoTS method, which did not consider the terrain shape, calculated two nearly "straight" trajectories in these two environments. However, these trajectory were not really straight and longer than the trajectories generated by TS-NoSus and TS-Sus, because the z-coordinate along these "straight" trajectories changed dramatically (according to Figure 16d,g). Recall that the trajectory smoothness was computed using Equation (33). Therefore, the trajectories generated by the NoTS method were not smooth in these two environments. Besides, the TS-Sus trajectories were still shorter and smoother than the TS-NoSus trajectories in these two environments (according to Table 5). In summary, the costs of the TS-Sus trajectories were the lowest, and the runtime of TS-Sus was only a little longer than that of NoTS.



(a) Simulation results in a non-planar environment with small undulations

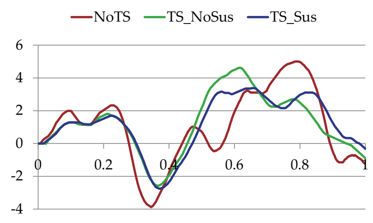


(b) Simulation results in a non-planar environment with medium undulations

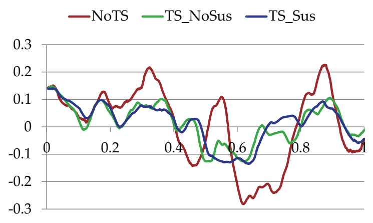


(c) Simulation results in a non-planar environment with large undulations

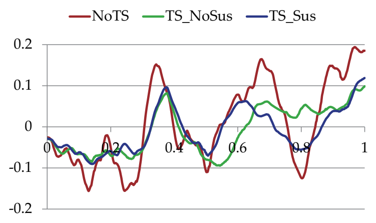
Figure 15. The trajectories generated by NoTS (red), TS-NoSus (green), and TS-Sus (blue).



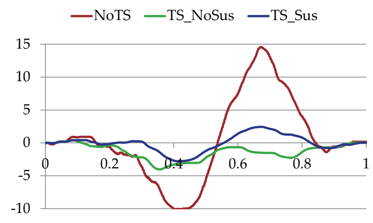
(a) *z*-coordinate



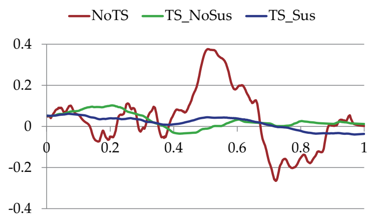
(b) Roll



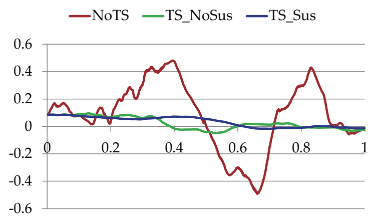
(c) Pitch



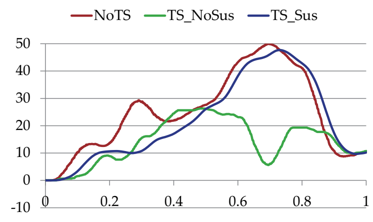
(d) *z*-coordinate



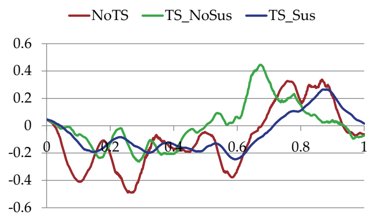
(e) Roll



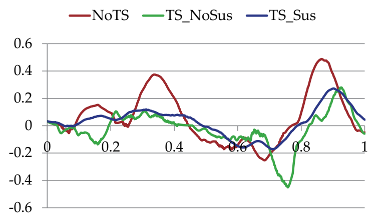
(f) Pitch



(g) *z*-coordinate



(h) Roll



(i) Pitch

Figure 16. The *z*-coordinate, the roll and the pitch of the vehicle along the trajectories. (**a**–**c**) The vehicle state along the trajectories shown in Figure 15a; (**d**–**f**) The vehicle state along the trajectories shown in Figure 15b; (**g**–**i**) The vehicle state along the trajectories shown in Figure 15c.

Table 5. The performance comparison of the trajectories shown in Figure 15 (averaged over 50 runs).

Terrain Surface Shown in Figure 15a							
Method	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**	**Runtime (s)**
NoTS	$+\infty$	0.9318	114.53	336.08	0.7738	0.3533	**3.220**
TS-NoSus	$+\infty$	0.9495	121.22	344.15	0.2050	1.0000	4.861
TS-Sus	$+\infty$	**0.9497**	**112.46**	**332.87**	**1.0000**	**0.3299**	3.338
Terrain Surface Shown in Figure 15b							
Method	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**	**Runtime (s)**
NoTS	$+\infty$	0.2765	119.81	354.92	0.2714	1.0000	**3.385**
TS-NoSus	$+\infty$	0.9708	123.82	366.95	0.5066	0.3973	4.217
TS-Sus	$+\infty$	**0.9795**	**117.97**	**351.42**	**1.0000**	**0.2880**	3.782
Terrain Surface Shown in Figure 15c							
Method	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**	**Runtime (s)**
NoTS	6.08	0.3461	127.96	379.39	0.2966	1.0000	**3.617**
TS-NoSus	11.69	**0.9317**	152.22	452.17	0.7318	0.4407	4.418
TS-Sus	**12.82**	0.9296	**124.65**	**378.46**	**1.0000**	**0.3748**	3.730

5.2. Real-World Experiments

In the real-world experiments, the point clouds of real terrain surfaces were generated by a real HDL-32E LiDAR. An all-terrain vehicle developed by Polaris (shown in Figure 17) was used to test our approaches. Besides, an inertial measurement unit (IMU) was combined with the global positioning system (GPS) to measure the roll and the pitch of the vehicle, which would be used to compute the errors of different pose estimation methods. The IMU and its specifications are shown in Figure 18 and Table 6, respectively. The GPS receiver is equipped with NT1065 "Nomada" and bladeRF as the RF front end. They can simultaneously receive various GNSS satellite signals, including GPS (L1, L2, L3, L5). The IMU is similar to an odometer, whose error increases as time goes on. This accumulated error can be decreased using GPS. The IMU can generate continuous measurements and is more robust in non-planar environments than other odometers. Therefore, the combination of the IMU and the GPS is suitable for obtaining the ground truths of the vehicle's roll and pitch.

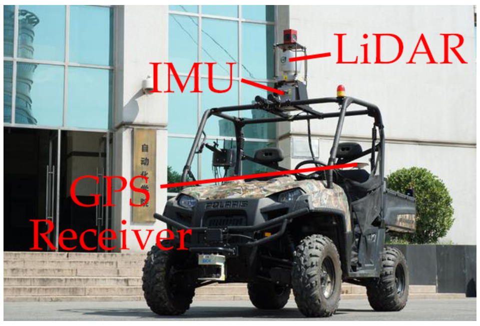


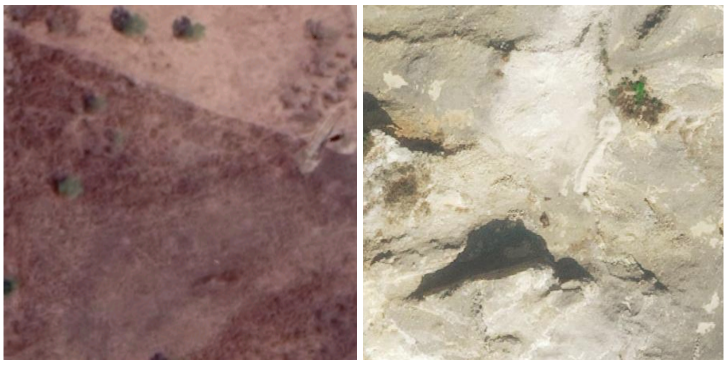
Figure 17. The vehicle used in experiments on rough terrain.

Figure 18. Inertial Measurement Unit (IMU).

Table 6. The specifications of the IMU.

Measurement Range (Roll, Pitch)	Accuracy (Roll, Pitch)	Resolution (Roll, Pitch)	Bandwidth
$-180.0°$ to $180.0°$	$< 0.2°$	$0.01°$	300 Hz

Figure 19a,b show a piece of uneven soil and a piece of rock ground, respectively. The terrain shape of the latter is more complex than that of the former. The vehicle was driven in the above two environments, and the SE(3) pose of the vehicle was recorded at a fixed frequency. Then, different pose estimation algorithms were performed based on the planar positions and the yaws of all the recorded poses. Table 7 shows the errors of the three different pose estimation methods. Typically, the errors are larger when the vehicle runs on a more complex terrain surface. The results show that the proposed method (PC-Sus) outperformed Kin-GP-VE and DEM-Kin (by approximately 36% and 49%).



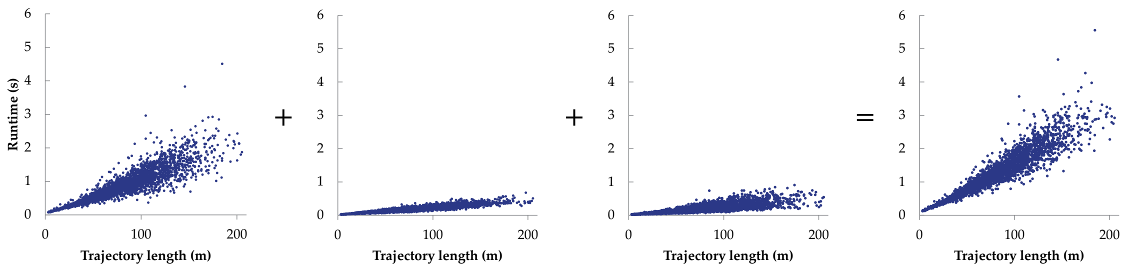
(a) Uneven soil (b) Rock ground

Figure 19. Non-planar environments in the real world.

Table 7. The errors of the different pose estimation methods.

Non-Planar Environment	Method	RMSE of Roll (Rad)	RMSE of Pitch (Rad)
Figure 19a	DEM-Kin	0.0763	0.0819
	Kin-GP-VE	0.0605	0.0640
	PC-Sus	**0.0343**	**0.0367**
Figure 19b	DEM-Kin	0.1104	0.1196
	Kin-GP-VE	0.0879	0.0963
	PC-Sus	**0.0617**	**0.0662**
Improvement over DEM-Kin		49.58%	49.92%
Improvement over Kin-GP-VE		36.56%	36.96%

To analyze the computational complexities of the different stages of the proposed trajectory planning approach, an experiment comprising 3000 different random planning queries (3000 pairs of start and goal poses) was conducted. For this experiment, the point clouds of the non-planar environments shown in Figure 19a,b were used. Each planning query was randomly set in one of these two environments. The results are shown in Figure 20. In all the three stages, the runtime and its variance increase approximately proportionally with the length of the planned trajectory. On average, the non-holonomic A* search is the most computationally expensive.

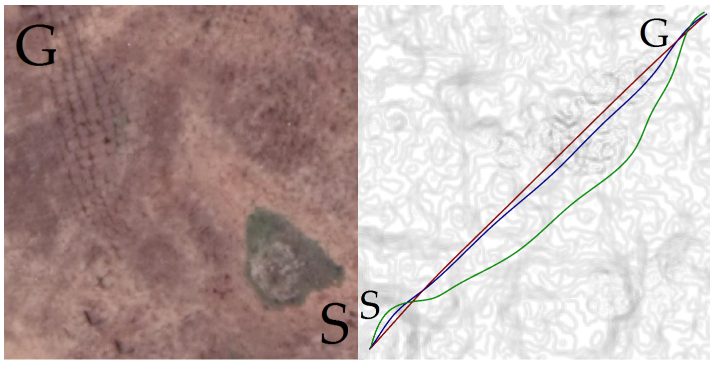


(a) Non-holonomic A* (b) Motion prediction (c) Optimization (d) Total runtime

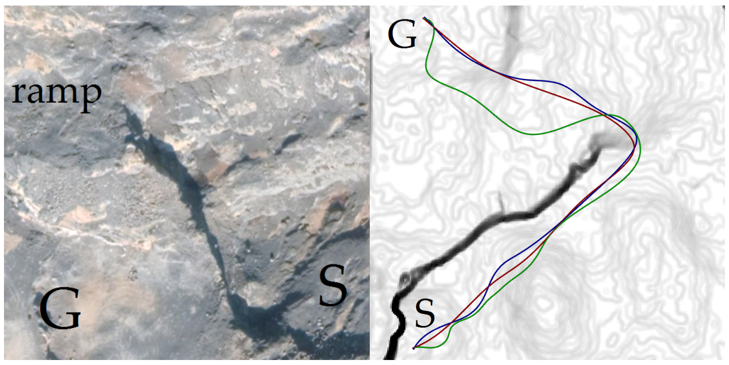
Figure 20. The statistic evaluation of the computational complexities.

Next, three different trajectory planning methods (NoTS, TS-NoSus, and the proposed TS-Sus) were compared in the real-world experiments. Figure 21a shows a piece of soil whose terrain undulations are small. In this environment, the TS-NoSus method computed a trajectory that avoided these small terrain undulations. However, the small terrain undulations often have little impact on the traversability of the vehicle that has a suspension system. Therefore, the propose TS-Sus method generated a trajectory that only avoided some large undulations. As a result, the traversability of the TS-Sus trajectory is nearly the same as that of the TS-NoSus trajectory (according to Figure 22a–c, and Table 8), but the TS-Sus trajectory is much smoother than the TS-NoSus trajectory (according to Table 8).

Figure 21b shows a piece of rock ground that has a ramp and many large terrain undulations. In this environment, the NoTS method computed a trajectory that only avoided the non-traversable areas (or called obstacles). The roll and the pitch along this trajectory were larger than those along the trajectories generated by the other two planners (according to Figure 22e,f). As a result, the traversability of this trajectory was smaller than the traversabilities of the other two trajectories (according to Table 8). In conclusion, the total costs of the trajectories generated by the proposed planner were the lowest in the comparative experiments. The runtime of our planner was a little longer than that of the NoTS planner, because our planner spent more time on processing the information of terrain shapes.

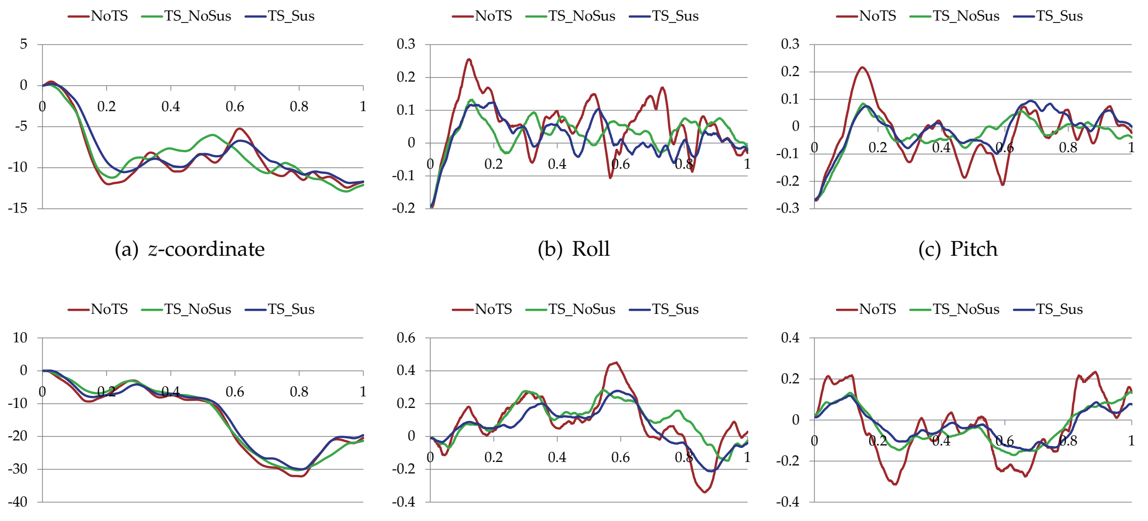


(a) Comparative experiment results on a uneven soil



(b) Comparative experiment results on a rock ground

Figure 21. The trajectories generated by NoTS (red), TS-NoSus (green), and TS-Sus (blue).



(a) z-coordinate (b) Roll (c) Pitch

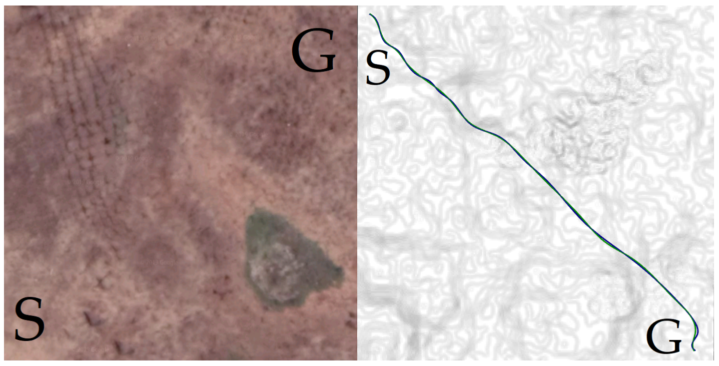
(d) z-coordinate (e) Roll (f) Pitch

Figure 22. The z-coordinate, the roll, and the pitch of the vehicle along the paths and the trajectories. (**a**–**c**) The vehicle state along the trajectories shown in Figure 21a; (**d**–**f**) The vehicle state along the trajectories shown in Figure 21b.

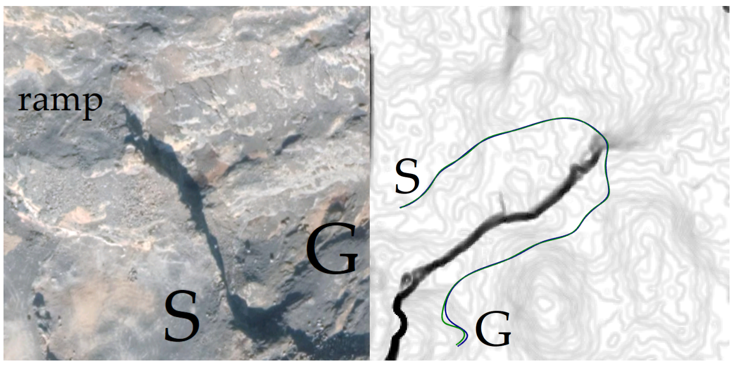
Table 8. The performance comparison of the trajectories shown in Figure 21 (averaged over 50 runs).

Non-Planar Environment Shown in Figure 21a							
Method	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**	**Runtime (s)**
NoTS	$+\infty$	0.9046	60.47	178.57	0.9970	0.6944	**1.489**
TS-NoSus	$+\infty$	**0.9185**	61.81	183.18	0.4490	1.0000	2.334
TS-Sus	$+\infty$	0.9174	**60.04**	**177.68**	**1.0000**	**0.5176**	1.519
Non-Planar Environment Shown in Figure 21b							
Method	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**	**Runtime (s)**
NoTS	13.45	0.5165	**69.09**	**196.83**	0.9982	1.0000	**3.103**
TS-NoSus	14.33	0.9155	78.05	222.62	0.8409	0.7358	4.840
TS-Sus	**14.51**	**0.9165**	70.54	199.08	**1.0000**	**0.6435**	3.174

At last, Figure 23 shows the trajectories (green) after optimization and the real trajectories (blue) of the vehicle in the real world. The blue trajectories were generated by making the vehicle track the green trajectories. Table 9 shows the performance comparison of these trajectories. The results show that there was little difference between the optimized trajectories (green) and the real trajectories (blue). This implies that the optimized trajectories were easy to be tracked by the vehicle on the real rough terrain. The reason is that planning based on the traversability can reduce the trajectory tracking error in a predictive way.



(a) Real-world experiment results on a uneven soil



(b) Real-world experiment results on a rock ground

Figure 23. The trajectories (green) after optimization and the real trajectories (blue) of the vehicle in the real world.

Table 9. The performance comparison of the trajectories shown in Figure 23. Note that the real trajectories were often different between different runs, so the data of the real trajectories were averaged over 50 runs.

Terrain Surface Shown in Figure 23a						
Trajectory	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**
Optimized trajectory	$+\infty$	**0.9165**	**60.93**	**180.28**	**1.0000**	**0.9797**
Real trajectory	$+\infty$	0.9128	61.02	180.65	0.9791	1.0000
Terrain Surface Shown in Figure 23b						
Trajectory	d_o **(m)**	τ	t_f **(s)**	l_f **(m)**	**Trajectory Smoothness**	**Total Cost**
Optimized trajectory	**6.44**	**0.9143**	68.04	194.59	**1.0000**	**0.9816**
Real trajectory	6.29	0.9085	**67.92**	**194.02**	0.9804	1.0000

6. Conclusions

This paper presents a traversability assessment method and a trajectory planning method, which can be applied to generate a safe and efficient trajectory for a car-like vehicle running on a rough terrain surface. The proposed suspension-based traversability calculation enables planning dynamically feasible trajectories on different terrain surfaces. Also, computing the traversability on demand based on original LiDAR points helps us get rid of the explicit terrain reconstruction or discretization. Besides, the proposed traversability assessment method (or trajectory planning method) is a general approach that can be used in any other navigation system of a ground mobile robot.

The proposed method has been tested and analyzed in both the simulation and the real-world experiments. According to the experimental results, the error of our pose estimation method, which accounts for the vehicle suspension model, was smaller than the other methods that ignore the suspension model. In a non-planar environment with various terrain shapes, the proposed traversability assessment method was validated by showing a heat map colored by the traversability. Besides, the different stages of the proposed planner were analyzed in terms of the trajectory costs and the computational complexities. The results indicate that the optimization based on the suspension model could make the trajectory smoother and easier to be tracked. Finally, the proposed planner was compared with some other state-of-the-art planners in various non-planar environments. In these comparative experiments, the proposed planner showed a better generalization ability to compute trajectories with lower costs on both simple and complex terrain surfaces.

In the future, the traversability will be calculated based on not only the geometric information but also the semantic information of rough terrain surfaces. The semantic-based traversability can enable the trajectory planner to navigate a vehicle more intelligently, which needs techniques to perform the semantic segmentation of 3D point clouds from laser sensors or images from visual sensors.

Author Contributions: Conceptualization, K.Z., Y.Y., M.F. and M.W.; Data curation, K.Z.; Formal analysis, K.Z.; Funding acquisition, Y.Y., M.F. and M.W.; Investigation, K.Z.; Methodology, K.Z.; Project administration, K.Z., Y.Y. and M.F.; Resources, Y.Y.; Software, K.Z.; Supervision, Y.Y. and M.F.; Validation, K.Z.; Visualization, K.Z.; Writing—original draft, K.Z.; Writing—review & editing, K.Z., Y.Y., M.F. and M.W.

Funding: The work presented in this paper was supported by the National Natural Science Foundation of China (No. 61473042, No. 61105092, and No. 61173076) and the Program for Changjiang Scholars and Innovative Research Team in University (No. IRT-16R06, No. T2014224). The authors would like to thank the suggestions on academic writing from Wenjie Song.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UGV	Unmanned Ground Vehicle
LiDAR	Light Detection and Ranging
DEM	Digital Elevation Map
ROS	Robot Operating System
GP	Gaussian Process
VE	Vehicle Experience
RMSE	Root Mean Squared Error
TS	Terrain Shape
IMU	Inertial Measurement Unit
GPS	Global Positioning System

References

1. Kelly, A.; Stentz, A. Rough terrain autonomous mobility—Part 2: An active vision, predictive control approach. *Auton. Robots* **1998**, *5*, 163–198. [CrossRef]
2. Lacroix, S.; Mallet, A.; Bonnafous, D.; Bauzil, G.; Fleury, S.; Herrb, M.; Chatila, R. Autonomous rover navigation on unknown terrains: Functions and integration. *Int. J. Robot. Res.* **2002**, *21*, 917–942. [CrossRef]
3. Bai, C.; Guo, J. Uncertainty-Based Vibration/Gyro Composite Planetary Terrain Mapping. *Sensors* **2019**, *19*, 2681. [CrossRef] [PubMed]
4. Zhou, X.; Bai, T.; Gao, Y.; Han, Y. Vision-Based Robot Navigation through Combining Unsupervised Learning and Hierarchical Reinforcement Learning. *Sensors* **2019**, *19*, 1576. [CrossRef] [PubMed]
5. Wellington, C.; Courville, A.C.; Stentz, A. Interacting Markov Random Fields for Simultaneous Terrain Modeling and Obstacle Detection. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 8–11 June 2005; pp. 1–8.

6. Daily, M.; Harris, J.; Keirsey, D.; Olin, D.; Payton, D.; Reiser, K.; Rosenblatt, J.; Tseng, D.; Wong, V. Autonomous cross-country navigation with the ALV. In Proceedings of the 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 24–29 April 1988; pp. 718–726. [CrossRef]
7. Ye, C.; Borenstein, J. A new terrain mapping method for mobile robots obstacle negotiation. In Proceedings of the Unmanned Ground Vehicle Technology, Orlando, FL, USA, 30 September 2003; pp. 52–63.
8. Ren, Z.; Wang, L.; Bi, L. Robust GICP-Based 3D LiDAR SLAM for Underground Mining Environment. *Sensors* **2019**, *19*, 2915. [CrossRef] [PubMed]
9. Peterson, J.; Chaudhry, H.; Abdelatty, K.; Bird, J.; Kochersberger, K. Online Aerial Terrain Mapping for Ground Robot Navigation. *Sensors* **2018**, *18*, 630. [CrossRef]
10. Vlaminck, M.; Luong, H.; Philips, W. Have I Seen This Place Before? A Fast and Robust Loop Detection and Correction Method for 3D Lidar SLAM. *Sensors* **2019**, *19*, 23. [CrossRef]
11. Pang, C.; Zhong, X.; Hu, H.; Tian, J.; Peng, X.; Zeng, J. Adaptive Obstacle Detection for Mobile Robots in Urban Environments Using Downward-Looking 2D LiDAR. *Sensors* **2018**, *18*, 1749. [CrossRef]
12. Castaño, F.; Beruvides, G.; Haber, R.E.; Artuñedo, A. Obstacle Recognition Based on Machine Learning for On-Chip LiDAR Sensors in a Cyber-Physical System. *Sensors* **2017**, *17*, 2109. [CrossRef]
13. Castaño, F.; Beruvides, G.; Villalonga, A.; Haber, R.E. Self-Tuning Method for Increased Obstacle Detection Reliability Based on Internet of Things LiDAR Sensor Models. *Sensors* **2018**, *18*, 1508. [CrossRef]
14. Maimone, M.; Johnson, A.; Cheng, Y.; Willson, R.; Matthies, L. Autonomous Navigation Results from the Mars Exploration Rover (MER) Mission. In Proceedings of the 9th International Symposium on Experimental Robotics, Singapore, 18–21 June 2004; pp. 3–13.
15. Wettergreen, D.; Jonak, D.; Kohanbash, D.; Moreland, S.; Spiker, S.; Teza, J. Field Experiments in Mobility and Navigation with a Lunar Rover Prototype. In Proceedings of the 7th International Conference on Field and Service Robotics, Cambridge, MA, USA, 13 July 2009; pp. 489–498.
16. Pfaff, P.; Triebel, R.; Burgard, W. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *Int. J. Robot. Res.* **2007**, *26*, 217–230. [CrossRef]
17. Stumm, E.; Breitenmoser, A.; Pomerleau, F.; Pradalier, C.; Siegwart, R. Tensor-voting-based navigation for robotic inspection of 3D surfaces using lidar point clouds. *Int. J. Robot. Res.* **2012**, *31*, 1465–1488. [CrossRef]
18. Garrido, S.; Malfaz, M.; Blanco, D. Application of the fast marching method for outdoor motion planning in robotics. *Robot. Auton. Syst.* **2013**, *61*, 106–114. [CrossRef]
19. Kummerle, R.; Hahnel, D.; Dolgov, D.; Thrun, S.; Burgard, W. Autonomous driving in a multi-level parking structure. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3395–3400. [CrossRef]
20. Khan, Y.N.; Komma, P.; Zell, A. High resolution visual terrain classification for outdoor robots. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops, Barcelona, Spain, 6–13 November 2011; pp. 1014–1021. [CrossRef]
21. Singh, S.; Simmons, R.; Smith, T.; Stentz, A.; Verma, V.; Yahja, A.; Schwehr, K. Recent progress in local and global traversability for planetary rovers. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; pp. 1194–1200.
22. Santana, P.; Guedes, M.; Correia, L.; Barata, J. Stereo-based all-terrain obstacle detection using visual saliency. *J. Field Robot.* **2011**, *28*, 241–263. [CrossRef]
23. Karumanchi, S.; Allen, T.; Bailey, T.; Scheding, S. Non-parametric learning to aid path planning over slopes. *Int. J. Robot. Res.* **2010**, *29*, 997–1018. [CrossRef]
24. Peynot, T.; Lui, S.T.; McAllister, R.; Fitch, R.; Sukkarieh, S. Learned stochastic mobility prediction for planning with control uncertainty on unstructured terrain. *J. Field Robot.* **2014**, *31*, 969–995. [CrossRef]
25. Krebs, A.; Pradalier, C.; Siegwart, R. Adaptive rover behavior based on online empirical evaluation: Rover–terrain interaction and near-to-far learning. *J. Field Robot.* **2010**, *27*, 158–180. [CrossRef]
26. Krüsi, P.; Furgale, P.; Bosse, M.; Siegwart, R. Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments. *J. Field Robot.* **2017**, *34*, 940–984. [CrossRef]
27. Santamaria-Navarro, À.; Teniente, E.H.; Morta, M.; Andrade-Cetto, J. Terrain Classification in Complex Three-dimensional Outdoor Environments. *J. Field Robot.* **2015**, *32*, 42–60. [CrossRef]
28. Mohanan, M.G.; Salgoankar, A. A survey of robotic motion planning in dynamic environments. *Robot. Auton. Syst.* **2018**, *100*, 171–185. [CrossRef]

29. Rasekhipour, Y.; Khajepour, A.; Chen, S.; Litkouhi, B. A Potential Field-Based Model Predictive Path-Planning Controller for Autonomous Road Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1255–1267. [CrossRef]
30. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
31. Sato, H.; Iwai, T. A new, globally convergent Riemannian conjugate gradient method. *Optimization* **2015**, *64*, 1011–1031. [CrossRef]
32. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
33. Pivtoraiko, M.; Knepper, R.A.; Kelly, A. Differentially constrained mobile robot motion planning in state lattices. *J. Field Robot.* **2009**, *26*, 308–333. [CrossRef]
34. LaValle, S.M.; Kuffner, J.J. Randomized Kinodynamic Planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
35. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
36. Amar, F.B.; Bidaud, P.; Ouezdou, F.B. On modeling and motion planning of planetary vehicles. In Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, 26–30 July 1993; pp. 1381–1386. [CrossRef]
37. Howard, T.M.; Kelly, A. Optimal rough terrain trajectory generation for wheeled mobile robots. *Int. J. Robot. Res.* **2007**, *26*, 141–166. [CrossRef]
38. Furgale, P.; Barfoot, T.D. Visual teach and repeat for long-range rover autonomy. *J. Field Robot.* **2010**, *27*, 534–560. [CrossRef]
39. McManus, C.; Furgale, P.; Stenning, B.; Barfoot, T.D. Lighting-invariant visual teach and repeat using appearance-based lidar. *J. Field Robot.* **2013**, *30*, 254–287. [CrossRef]
40. Rekleitis, I.; Bedwani, J.L.; Dupuis, E.; Lamarche, T.; Allard, P. Autonomous over-the-horizon navigation using LIDAR data. *Auton. Robots* **2013**, *34*, 1–18. [CrossRef]
41. Woods, M.; Shaw, A.; Tidey, E.; Van Pham, B.; Simon, L.; Mukherji, R.; Maddison, B.; Cross, G.; Kisdi, A.; Tubby, W. Seeker—Autonomous Long-range Rover Navigation for Remote Exploration. *J. Field Robot.* **2014**, *31*, 940–968. [CrossRef]
42. Silver, D.; Bagnell, J.A.; Stentz, A. Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain. *Int. J. Robot. Res.* **2010**, *29*, 1565–1592. [CrossRef]
43. Gandhi, P.; Adarsh, S.; Ramachandran, K. Performance Analysis of Half Car Suspension Model with 4 DOF using PID, LQR, FUZZY and ANFIS Controllers. *Procedia Comput. Sci.* **2017**, *115*, 2–13. [CrossRef]
44. Ikenaga, S.; Lewis, F.L.; Campos, J.; Davis, L. Active suspension control of ground vehicle based on a full-vehicle model. In Proceedings of the 2000 American Control Conference, Chicago, IL, USA, 28–30 June 2000; pp. 4019–4024. [CrossRef]
45. Ho, K.; Peynot, T.; Sukkarieh, S. Traversability estimation for a planetary rover via experimental kernel learning in a Gaussian process framework. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 3475–3482. [CrossRef]
46. Chu, K.; Kim, J.; Jo, K.; Sunwoo, M. Real-time path planning of autonomous vehicles for unstructured road navigation. *Int. J. Automot. Technol.* **2015**, *16*, 653–668. [CrossRef]