Path Planning of UGV Based on Bézier Curves

Yanming Hu†‡¶, Decai Li†‡, Yuqing He†‡∗ and Jianda Han†‡∥

†State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
‡Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110016, China
¶University of Chinese Academy of Sciences, Beijing 100049, China
∥College of Artificial Intelligence, Nankai University, Tianjing 300071, China
E-mails: huyanming@sia.cn, lidecai@sia.cn, jdhan@sia.cn

(Accepted December 5, 2018. First published online: January 21, 2019)

SUMMARY
An effective path planner is critical for autonomous traversal of unmanned ground vehicles (UGVs) in harsh environments. This paper describes a novel path planning method considering Bézier curves and a two-layer planning framework. In the two-layer framework, a road centerline (RCL) estimator located on the upper layer works as a global planner to obtain the local target for the bottom local planner. The RCL is estimated from a series of candidate Bézier curves based on a safety criterion. In the bottom layer, an optimal trajectory planner and a speed planner make up the local planner to obtain the desired steering turning angle and linear speed. The criteria for optimal trajectory selection are designed for comfortable driving. Road safety is considered in the speed planner for robust driving. Three sets of simulations are used to evaluate and quantify the relative performance of variations of our path planning algorithm. The proposed path planning method is implemented on a modified Polaris RZR 800 UGV, too. Two experiments based on this UGV are set up in the country road environment to demonstrate the viability of the proposed method.

KEYWORDS: Path planning; Bézier curves; Unmanned ground vehicle; Comfortable driving; Speed planning.

1. Introduction
Unmanned ground vehicles (UGVs) are used to autonomously drive and perform certain tasks under different road and environmental conditions. UGVs have broad application prospects in society, particularly in the military field, such as in high-risk operations, special operations, and material transportation tasks. A complete UGV system usually consists of perception, path planning, and control modules. Here in this paper, we focus mainly on the path planning problem in country road environment with flat soil surface and sensible road shape. At the same time, we assume that the environment map and obstacles are available through some existing method, such as that in ref. [2], while the precise vehicle states can be acquired by many existing approaches.3–5

A path planning task for obtaining an optimal solution in real time in a long range and uncertain environment is complex. The layering concept, which has been widely used in planning tasks,6,7 aims to reduce this complexity by dividing the complete path planning task into several simple or small subtasks. Some sparse waypoints8 or even a direct reference global path9 is a good alternative

* Corresponding author. E-mail: heyuqing@sia.cn
Bézier Curves based Path Planning

Fig. 1. Two-layer path planning scheme of a mobile robot.

to a precise global map. Figure 1 shows the two-layer planning framework, global planner, and local planner employed in this study. The global planner uses sparse waypoints or a local map to obtain the local target for local planner. Then the desired controlling commands of UGVs can be generated in the local planner based on the local target and local map obtained from sensors such as a camera and radar.

The two-layer path planning scheme has been successfully applied in Stanley, which won the 2005 DARPA Grand Challenge. Its algorithm combines three parts. First, a road centerline (RCL) is estimated from environmental map and route definition data format. Second, a base trajectory and a set of candidate trajectories are obtained by RCL. In the end, some optimization criteria are used to select the optimal trajectory among multiple candidate trajectories. The two-layer path planning scheme is also widely used in the 2007 DARPA Urban Challenge, such as the top three places, Boss, Junior, and Odin. These related works demonstrate that the layering path planning scheme is suitable for UGV system.

In most two-layer planning methods, the road estimator and local planner are two important problems. The road estimator is often used to obtain RCL. For example, Stanley defines RCL as the center of two boundaries of the road, which is estimated by using the discrete obstacle map and the given accurate waypoints. But in most on-road cases, such as the country road environment, the road boundary may be short grass or small stones, which is travelable by UGV. If they are to be treated as obstacles, then the traversable road with short grass and small stones will be judged impassible. There is a dilemma between road shape estimation (RSE) and travelable area extraction for discrete obstacle map-based methods. Boss estimates RCL between two connected accurate waypoints by using a sample importance resample filter, where Taylor expansion of a clothoid is used to represent the road. This method requires that the curbs and obstacles are estimated before road estimation. These additional computational burdens and the fact of its dependence on the original sensor data reduce its real-time performance. RCL of Junior and Odin is directly computed from accurate waypoints through spline interpolation techniques. These methods are feasible for an urban environment with accurate waypoints and structured road, instead of unstructured and tough terrain. In addition, accurate waypoints could be collected from an accurate global map or recorded while driving. Both of them need to explore the environment by robot or human in advance. A more common technique is to use the Google Earth to obtain waypoints while losing some accuracy. It is important for UGVs to improve the ability of road estimation based on only local real-time information (local map) while using inaccurate global information.

After RSE, local planning is another challenging problem. Odin’s local planner uses pre-computed movement primitives (actions) and the A* graph search method to obtain the optimal sequence of actions. This method plans velocity and steering rate simultaneously. However, it is usually difficult to balance the two commands to match the predesigned path shape. Thus, Stanley, Boss, and Junior all decouple velocity and steering commands in their local planning algorithms. This means that the local planner is split into trajectory planner and speed planner. Trajectory planner aims to obtain the optimal executable trajectory (OET), which is used to calculate the desired steering command (curvature). Speed planner plans the desired velocity command for UGVs.

Trajectory planners of Stanley and Boss are called discrete optimization. In this scheme, a finite set of trajectories is firstly generated as candidate trajectories. Then, OET is obtained among
these candidate trajectories based on some predesigned criteria. Junior also uses this trajectory planning scheme, which is presented in ref. [15] without detailed explanation. Trajectory generators of Stanley and Boss are based on the model predictive control (MPC) approach pioneered by Kelly and Stentz. But MPC simulations are necessary in each iteration process of optimization, which is sensitive to initial parameters, that is, the optimization may not converge to targets without proper initial parameters. To solve this problem, a coarse mapping of vehicle states and corresponding action parameters is stored in a parameter lookup table. Besides MPC-based method, sampling the state space is conducted in lots of works. As a sampling-based method, a quartic Bézier curve-based trajectory generation was proposed in our previous work. The curvature continuous and kinematical constraints of UGV are assured due to the Bézier curves themselves. Besides, a Bézier curves-based trajectory generator is insensitive to initial parameters.

After candidate generation, the optimal criteria, such as safety, smoothness, and consistency, also need to be designed to select OET. Safety favors trajectory far away from obstacles. Smoothness favors smoother turning of running, which will lead to a higher cost for the trajectory with bigger curvature. Consistency favors smaller changes of heading. The criteria of Stanley, Boss, and Junior punish running over obstacles (safety) and away from RCL. Under these criteria, UGV may take a sharp turn to avoid obstacles. Thus Boss considers smoothness of trajectory, which encourages UGV to follow a trajectory with lower curvature. Besides, the inconsistent curvature between two connected planning cycles may occur in some scenarios. Here is an example scenario to explain the necessity of the criterion of consistency. Assuming that UGV aim to avoid a front obstacle. Current local perception information may encourage UGV to run from the left side of obstacle. Without considering consistency, path planner may encourage UGV to bypass the obstacle from its right side when new environment information is obtained. This scenario may cause drastic changes of steering. In fact, if the differences between two sides are not too big, our human drivers will keep consistency driving. Hence, it is important for the UGV to select OET considering safety, smoothness, and consistency simultaneously. This problem, however, is scarcely researched.

As far as the speed planner is concerned, its importance is often ignored since, in most cases, the UGV moves at some constant speed, except the starting and stopping phase. However, if it is necessary to change moving speed or to wander in some tough terrain, the speed planner is important to ensure safety of the UGV. The speed planner of Boss is directly computed using the interpolation method based on several predesigned velocity limitations. The speed planner of Stanley considers the road factors such as terrain slope and roughness. Real experiments show that it is very useful to reduce resonance in bumpy terrain and thus present great priorities. However, some other factors such as road width and obstacles are also useful and important for the speed planner and thus should be considered.

In this paper, on the basis of traditional two-layer scheme, we develop a new path planning algorithm. In the upper layer, an RCL estimation method is proposed to gain local waypoints for the bottom local planner. Then, the trajectory planner and speed planner are designed to obtain the desired steering angle and velocity, respectively. Inspired by our previous Bézier curves-based trajectory generator, the second-order and third-order Bézier curves are used to generate candidate RCLs and trajectories, and we use the discrete optimization approach to estimate both RCL and OET. It should be noted that our new OET criteria simultaneously considers safety (obstacle avoiding), smoothness (average curvature of trajectory), and trajectory), and consistency (change between two connected OETs). The speed planner combining curvature and road safety improves robustness metrics. The computational complexity of both RCL estimator and trajectory planner is $O(n)$, where $n$ is the number of candidate RCLs or trajectories. It is enough for real applications.

Compared with related work, we focus on on-road environment and only sparse inaccurate waypoints are available. This requires the planner to be stronger and robust with respect to uncertainties. And computational efficiency is particularly important if the platform lacks the computational resources necessary for more complex algorithms. The following three differences show the novelty of our method:

First, RCL is optimized using the concept of traversability defined by the occupancy probabilities (OP) map (a statistical approach). This enables the algorithm to preserve great robustness under perception uncertainties.

Second, the trajectory planner considers the criteria of safety, smoothness, and consistency simultaneously to select OET. This enables the UGV to drive along a smooth and stable trajectory with
great robustness behavior. Furthermore, since the curvature continuous and kinematical constraints of UGV are assured due to Bézier curves themselves, the average time cost of trajectory planning is in milliseconds level. This result will be shown in experimental.

Third, our speed planner considers both body safety and environment safety simultaneously through considering curvature and road traversability. The flexible regulation of speed in a risky road enables the UGV to move at lower speed and thus keep high safety.

The remainder of this paper is organized as follows. Hardware and software frameworks of the UGV setup are briefly introduced in Section 2. The method of generating local target is covered in detail in Section 3. The design of local path planner comprising candidate trajectory generator, optimal trajectory selector, and speed planner is explained in Section 4. The results of simulations, time complexity analysis, and experiments are presented in Section 5. Finally, conclusions of this study are presented in Section 5.

2. Hardware and Software Framework
Before presenting the methodology in detail, the experimental platform of the target application is introduced. First, we present the hardware and software systems of the UGV. We then focus on path planning, which is the primary focus of this paper.

2.1. Hardware layout and system framework
The UGV platform used in this study was a modified version of Polaris RZR 800, equipped with a four-stroke twin-cylinder engine. The engine provides electric power to the computing system of UGV. The minimum turning radius of RZR 800 is 4 m. The friction factor between tire and on-road surface is approximately 0.3.

Two LIDARs and an IMU were installed on board. Figure 2 shows their layout. A Velodyne HDL-32E is mounted on the front roof of the vehicle for obstacle and road detection. To improve the performance of detection of obstacle and road, a Velodyne HDL-32E is mounted towards the front at a forward angle, and a SICK LIDAR was mounted behind the vehicle to provide sensor coverage around the entire vehicle. The maximum sensor range of Velodyne HDL-32E is 70 m and of SICK LIDAR is 50 m. Because of the mounting attitude of sensors, the effective sensor range is lower than 30 m. In addition, the point cloud density is too lower to use for obstacle extraction, when the distance is more than 10 m. So, our UGV only can discover an obstacle within 10 m in front of it. Such short sight limits the maximum speed of UGV (2.5 m/s for on-road environment with obstacles on road surface, and 3.5 m/s for on-road environment without obstacles on road surface in this paper). In addition, the maximum velocity of the robot is also associated with other factors such as the controller, environment, and hardware. In this paper, velocity constrains are obtained by testing. The test steps are as follows:

**Step 1:** Initialization. Given pose $P_I = (x_I, y_I, \varphi_I)$ and an obstacle in front of it; $v_{\text{max}} = 1.0 \text{ m/s}$.

**Step 2:** Driving UGV to the pose $P_I$.

**Step 3:** UGV running towards the obstacle by $v_{\text{max}}$. Once the obstacle is found, the UGV will decrease its velocity as soon as possible. If UGV stopped without collision, then $v_{\text{max}} \leftarrow v_{\text{max}} + v_\delta$, where $v_\delta$ is a small positive number, and go to Step 2; otherwise, return $v_{\text{max}} \leftarrow v_{\text{max}} - v_\delta$ and stop test procedure.

For inertial navigation, OxTS Inertial+ and external GNSS receivers are mounted at the center of the roof. OxTS Inertial+ uses accelerometers and angular rate sensors to smooth the jumps occurring in GNSS and fill in missing data. Other important measurements such as heading, pitch, and roll can be measured.

The dataset of LIDARs and IMU were acquired with a robot operation system (ROS)-equipped computer consisting of four-core Intel CPU (2.6 GHz) and 4 Gb of memory. The perception and planning modules were installed on this computer to obtain the desired commands for another computer equipped with QNX. The output signs were used to control four motors corresponding to the brake, throttle, gear, and steering wheel.

The software system of the UGV setup comprises four parts, as shown in Fig. 3, namely sensor data acquisition, perception, path planning, and motor control. The main tasks of sensor data
acquisition include receiving and preprocessing the data. The perception module can be divided into mapping and localization. In the mapping process, laser data are used to build the local occupancy-elevation grid map of the environment. Localization is performed to make the robot aware of its position in world coordinates. The objective of path planning is to obtain control commands based on environment information. The motor controller module uses the path planning output as the desired input for the various motors and thereby controlling them (i.e., steer and throttle).

2.2. Path planning framework
The objective of path planning of UGV is to use sparse waypoints and periodically update the local map to obtain the desired control commands (i.e., steer turn angle and linear speed). Figure 4 shows the two-layer path planning framework. The local map and waypoints used as environmental information are inputted to the first layer in the global planning stage. The global planning stage is divided into two parts: a road shape estimator and a local target generator. The road shape estimator is used to obtain the most probable RCL using the current local map. Subsequently, in the local target generator module, the most probable RCL is corrected using the current waypoint. Finally, the local target is selected on the corrected RCL.
3. Global Planning

The global planning module (Fig. 5) is used to obtain the local target for the local path planner. First, a set of candidate RCLs is generated using Eq. (B2), as given in the Appendix. An optimal RCL is then selected based on the safety of every curve. Next, the current waypoint is used to correct the result. The final local target is chosen on the corrected RCL. Before presenting the methods of global planning, the safety of curves, which is a common criterion for RCL estimation, is discussed in this section. OET estimation is given in the next section.

3.1. 2D grid local map

A 2D grid local map is directly obtained by the robot-centric elevation mapping method. (The inertial frame and local frame are two basic coordinate systems in our UGV system, which are explained...
in detail in Appendix A. The coordinate of a message can be conveniently transformed into different coordinate systems by TF, a ROS package. Without loss of generality, all messages used in this paper were transformed into the local frame. This method uses 3D point clouds to construct a 2D grid local map stored by an occupancy map message (http://docs.ros.org/api/nav_msgs/html/msg/OmnioccupancyGrid.html) type in ROS. Each grid cell contains three information denoted by \( M_i = (x_i, y_i, p_i) \). \( x_i \) and \( y_i \) are the coordinates of \( i \)th cell in local frame. \( p_i \) is the associated OP, which is in the range \([0, 100]\). A higher value means a higher risk. If \( p_i = 100 \), then occupancy probability of this cell is 1.0, which means there is an obstacle in this cell. Both discrete obstacle map and occupancy map will judge an obstacle by the object’s height or other features such as color and texture for vision-based method. In this paper, if the height of an object is greater than 0.5 m, it will be an obstacle for our UGV described in Section 2. The difference between the two types of map is that an object with a height less than 0.5 m is a free area for discrete obstacle map but an occupancy probability for occupancy map. An occupancy map retains more information of environment, which is useful for path planning. For example, although both flat and stone road are drivable, an occupancy map will use a path planning algorithm to choose the flat road.

Figure 6(a) shows an example of mapping result. Comparing the mapping result in Fig. 6(b) with the real scene in Fig. 6(a), the road can be recognized in 2D grid local map. For visualization purposes, the occupancy grid map was transformed to gray image in this paper. The relation between probability \( p_i \) and gray scale value \( g_i \) is shown in Eq. (1).

\[
g_i = 2.55 \times (100 - p_i) \tag{1}
\]
3.2. Cost of safety of curves

An OET represents the cost of safety by $S_c$ when the UGV moves on the road or along the trajectory. The OP of each grid cell in the local map represents the risk of this cell. The car bumps and sensor noise may result in inaccuracy or even errors in the local map. In these cases, an average OP is more robust than the maximum OP, because maximum OP can represent a noisy measurement of occupancy probability.

Figure 7 shows the calculation of the likelihood of RCL for curves. In this study, we use the average OP value in the active areas of the curve (similar to the area surrounded by white curves in Fig. 7) to approximately determine $S_c$ of the curve. The curve with a lower average OP is more likely to be an RCL. The size of the active area of the curve is controlled by the parameter $d$, as shown in Fig. 7. Assuming that uniform sampling is done in the active area and $N$ sample points are determined, we can obtain the safety of the curve as follows:

$$S_c = \frac{1}{N} \sum_{i=1}^{N} p_i$$

Here, $p_i$ is OP of the $i$th sampling point in the active area of candidate RCL.

The active area of the curve is determined using the parameter $d$, which represents the sensitivity ranges of the curve. We directly choose the half-road width $d = 4$ m in estimating RCL and $d = 0.8$ m in estimating OET. The road width equals $\sim 8$ m, and the width of UGV equals 1.6 m.

3.3. Road shape estimator

RCL expresses road information in this study. The local change in true road value is small. Thus, a second Bézier curve can be used to accurately estimate the local RCL. Three control points should be selected for every candidate RCL using Eq. (3). The origin of coordinate of local frame, which is also the position of UGV’s body center, is used as the first control point ($P_{0i}$), where $i$ is the index of $i$th candidate RCL. We used an assigned point in front of UGV, which is on the X-axis of local frame, as the second control point ($P_{1i}$). The distance between $P_{1i}$ and $P_{0i}$ is $\gamma_g = 1/6L_{map}$ in this paper, where $L_{map}$ is the height or width of the local map. The last control point ($P_{2i}$) is obtained by sampling the front edge of the local map.

$$P_{0i} = \begin{cases} x_{pi,0} = 0 \\ y_{pi,0} = 0 \end{cases}$$

$$P_{1i} = \begin{cases} x_{pi,1} = \gamma_g \\ y_{pi,1} = 0 \end{cases}$$

$$P_{2i} = \begin{cases} x_{pi,2} = 3\gamma_g \\ y_{pi,2} = \left(i - \frac{N_c}{2}\right)d_g \end{cases}$$
Fig. 8. Road shape estimation using a half local map wherein only the front part of the local map is shown. (a) The blue curve represents one of the candidate RCLs, and the yellow shadow area represents the active area of this curve. (b) The candidate RCLs and their active areas are determined. (c) and (d) The red curves indicate the final optimal RCLs.

where $N_g$ is the number of candidate RCLs and $d_g$ is the sample interval. After $N_g$ sets of control points are collected, the candidate RCLs can be generated using Eq. (B2) in the Appendix B.

Figure 8 shows the optimization process of RCL. The candidate solutions are directly sampled on the local map. Then $S_c$ of every candidate curve is calculated using Eq. (2). The candidate RCL, which has the highest value of $S_c$, is selected as the current RCL. RCL obtained using only the local information of the environment might misguide the UGV to choose a wrong branch of road at a particular junction. The waypoint near the junction is suitable to guide the UGV. In the next section, the method of using the waypoint to correct the optimal RCL is explained.

3.4. Corrected RCL

In Fig. 9, there are two branch roads, one of which goes straight ahead and the other one is towards the left, as shown in the local map. The optimal RCL, indicated in yellow, obtained using the safety of curves guides the UGV to an incorrect road without considering the global information. Thus, the most probable RCL needs to be fixed using the current waypoint ($P_{cw} = (x_{cw}, y_{cw})$). The last control point determines the direction of RCL, so it is corrected using Eq. (4). Then the new three control points $R_o = \{P_{o0}, P_{o1}, P_{o2}\}$ can be used to generate the corrected optimal RCL (the red curve shown in Fig. 9).

$$P_{new2} = \begin{cases} x_o^2 + y_o^2 \cos (\phi) \\ y_o^2 + y_o^2 \sin (\phi) \end{cases}$$

where $\phi = \tan^{-1} \left( \frac{y_{o2}-y_{o1}}{x_{o2}-x_{o1}} \right)$.

Finally, the objective of global planning is to obtain the local target for local planning. Figure 9 shows that the local target is directly chosen on the corrected RCL using Eq. (5). Notably, the orientation of local target is obtained as the derivative of the point where the target is located on the corrected RCL. The distance between the local target and the UGV is $2y_g$, which ensures that the local target lies on the local map and maintains a reaction distance with respect to the UGV.

$$P_g = \arg \min_{P=(x,y,\phi) \in ORCL} \left\{ D_{P2R} | D_{P2R} > 2y_g \right\}$$

where $P = (x, y, \phi)$ is the sampling pose on the corrected optimal RCL, $D_{P2R}$ is the distance from this pose to UGV. To conclude, the complete global planning algorithm is given in Table I. The inputs include current local map (M), current waypoint ($P_{cw}$), number of candidate RCLs ($N_g$), sampling interval ($d_g$), curve interpolation interval ($\Delta t$), half-road width ($d$), one-sixth of the local map length ($y_g$). The output of global planning algorithm is the local target ($P_g$).
4. Local Planning

The objective of the local planner is to generate the desired commands for motor controller, as shown in Fig. 10. The candidate trajectory generator, optimal trajectory selector, and speed planner are the three main modules of the local path planner in this study. Collision detection and accessibility detection help in deleting infeasible curves from the candidate samples. The optimal trajectory selector considers safety, smoothness, and consistency with respect to the previous optimal trajectories and
selects the optimal trajectory among the candidate trajectories. The speed planner considers the inherent constraints of a robot. Moreover, the speed planner uses a curvature-related function and a decay factor, which reflect the safety of the road.

4.1. Candidate trajectory generator
4.1.1. Third Bézier candidate trajectories. The third Bézier curve, calculated by Eq. (B4) in the Appendix, can be used to estimate the trajectory. First, we need to obtain the four points of every candidate trajectory. We use \((x, y, \phi)\), which represents the pose on the local frame. \(x\) and \(y\) are the coordinates on the local map, and \(\phi\) is the heading angle of the pose. The poses of the UGV and the current local target are assumed as \(P_s = (0, 0, 0)\) and \(P_g = (x_g, y_g, \phi_g)\), respectively.

a) \(d\) is the sample interval of candidate trajectories. Here, \(d\) is directly equal to the resolution of the local map represented by the grid cells.\(^{32}\)

b) \(D_{s,g}\) is the distance to the UGV from the current local target.

c) \(N\) is the number of candidate trajectories. \(N = r_w/d\), where \(r_w\) is road width.

d) \(\gamma\) is the parameter used to adjust smoothness of the curves. We chose 1/3 in this study for smoothness trajectories.

e) \(P_{i-1} = (x_{pi,i-1}, y_{pi,i-1})\) is \(i\)th control point of the \(i\)th candidate trajectory.

f) The four control points of the \(i\)th candidate trajectory can be calculated using Eq. (6).
980

Bézier Curves based Path Planning

Fig. 11. Examples of the generated candidate trajectories, for which the value of the local target heading is 0. The four group points of the candidate trajectories are indicated by p0, p1, p2, and p3.

\[
P_i = \begin{cases} 
  x_{pi,0} = 0 \\
  y_{pi,0} = 0
\end{cases}
\]

\[
P_i = \begin{cases} 
  x_{pi,1} = y_{D_{2\pi}} \cos(\varphi_g) \\
  y_{pi,1} = y_{D_{2\pi}} \sin(\varphi_g)
\end{cases}
\]

\[
P_i = \begin{cases} 
  x_{pi,2} = x_{pi,3} - y_{D_{2\pi}} \cos(\varphi_g) \\
  y_{pi,2} = y_{pi,3} - y_{D_{2\pi}} \sin(\varphi_g)
\end{cases}
\]

\[
P_i = \begin{cases} 
  x_{pi,3} = x_g + d \left( i - \frac{N-1}{2} \right) \sin(\varphi_g) \\
  y_{pi,3} = y_g + d \left( i - \frac{N-1}{2} \right) \cos(\varphi_g)
\end{cases}
\]

Figure 11 shows a clearer example of generating candidate trajectories. To put it simply, the local target, selected in Fig. 10, is directed straight ahead of the UGV pose, and \( \varphi_g = 0 \). The current UGV position \( P_i \) is used as the first control point \( P_0 \) of trajectory, which is the start of those candidate trajectories. The local target \( P_g \) is used as the reference point. The last control points of candidate trajectories are uniformly sampled from line \( L_1 \) in Fig. 11(b), which is through \( P_g \) and perpendicular to \( L_2 \). The line \( L_2 \) is through the local target and with angle \( \varphi_g \). The local target is in the middle of these last control points. The second control point of candidate trajectories is on line \( L_3 \) in Fig. 11(c). \( L_3 \) passes through UGV position and has heading angle \( \varphi_s \). The distance from the second control point to UGV pose is one-third length of the distance from UGV pose to local target \( D_{2\pi} \). The third control points are sampled on line \( L_4 \), which is perpendicular to \( L_2 \). The distance between \( L_1 \) and \( L_4 \) is \( \frac{1}{3} D_{2\pi} \). We use Eq. (6) to calculate the four control points of all candidate trajectories. It should be noted that all the candidate trajectories share the same \( P_0 \) and \( P_1 \). The difference lies in \( P_2 \) and \( P_3 \).

4.1.2. Collision detection. If the UGV follows a trajectory and faces a collision, we define the trajectory as collision trajectory. Collision detection is only associated with obstacles \( (p_i < 100) \) on 2D map. The other objects \( (p_i < 100) \) may cause confusion. So, in Fig. 12, the map retaining only the obstacles is used to explain the collision detection process without loss of generality.

In Fig. 12, obstacle is represented by a black rectangle. Figure 12(a) shows collision-free trajectories in the case where the robot is treated as a particle. Each point on the collision-free trajectory cannot appear in the obstacle area. However, the UGV body constraint was neglected in collision detection. Thus, the UGV may be at a risk of colliding with obstacles when the UGV follows a collision-free trajectory generated using the sample detection method.
Bézier Curves based Path Planning

Fig. 12. Examples of collision detection. (a) Collision-free trajectory without considering the body size of the vehicle. (b) Collision detection by considering the body size of the vehicle. (c) The results of collision detection.

The UGV body constraints should be considered in collision detection. We use specific length and width of the rectangle to represent the body of UGV. The length and width of the rectangle is associated with those of the UGV. Figure 12(b) shows the collision detection process. Any point in the rectangle associated with the collision-free trajectory cannot exist in the black area, which denotes the area of the obstacle. Figure 12(c) shows collision-free trajectories.

4.1.3. Accessibility detection. The UGV is a non-holonomic constraint robot. If a trajectory does not meet the non-holonomic constraint, the trajectory is considered inaccessible. The accessibility of every collision-free trajectory should be detected appropriately because of the constraint on the minimum wheel radius of the UGV. This ensures that the optimal trajectory is correct and feasible. The relationship between curvature and wheel radius is as follows.

\[ k_{\text{max}} = \frac{1}{r_{\text{min}}} \]  

(7)

Here, \( r_{\text{min}} \) is the minimum wheel radius, and \( k_{\text{max}} \) is the relevant maximum curvature.

We can use the constraint of the maximum curvature of each collision-free trajectory, represented by \( k_{i, \text{max}} \), to further test collision-free trajectories. If \( |k_{i, \text{max}}| < k_{\text{max}} \), the relevant trajectories are deleted from the original space of candidate trajectories.

Now, all the candidate trajectories are feasible for the UGV; however, the number of trajectories is still high. Next, we discuss the working of the optimal trajectory selector and obtain the final optimal trajectory.

4.2. Optimal trajectory selector

The optimization criteria are obtained to estimate the trajectory using the optimal trajectory selector. First, the optimal trajectory selector is designed to select an optimal trajectory from a number of candidate trajectories. Safety \( \vartheta_r \), smoothness \( \vartheta_s \), and consistency \( \vartheta_c \) of the trajectory are used as optimality criteria. The weighted sum of the three properties (8) is used to evaluate the usefulness of each feasible candidate trajectory. The optimization problem is divided into three sub-optimization problems.

\[ \text{cost} = \omega_r \vartheta_r + \omega_s \vartheta_s + \omega_c \vartheta_c \]  

(8)

4.2.1. Safety. We need to choose a better trajectory when the number of feasible candidate trajectories is more than 1. Notably, the trajectory farther away from the obstacle must be chosen because this one might be safer than the one nearer to the obstacle. This factor is called safety.

We determine the safety of every feasible candidate trajectory using Eq. (2). The parameter \( d \) is chosen as 0.8 in this study, which is the half-width of UGV. Figure 13 shows the safety of each feasible candidate trajectory. An obstacle is marked using a black cell on the map. The gray cell is an obstacle, but has a lower safety value than the white cell. Figure 5 shows the detailed mapping from the gray cell for safety. The left-most candidate trajectory with a maximum \( s_c \) is chosen as the best trajectory. We normalized \( s_c \) of all the feasible candidate trajectories using maximum \( s_c \) and minimum \( s_c \) into [0, 1]. In the following section, we use Norm(.) to mark the normalized process.
4.2.2. Smoothness. Safety is not the only consideration in practical driving. We need to also consider driving quality, which can be represented by the smoothness of the trajectory followed by the UGV. Smoothness is represented by the mean curvature of all the points on the trajectory. The cost associated with smoothness is represented by $\vartheta_s$, and the cost associated with smoothness of the $i$th candidate trajectory is as follows.

$$\vartheta_i^s = \text{Norm} \left( k_i^0 \right)$$  \hspace{1cm} (10)

Here, $k_i^0$ is the mean curvature of all the points on the trajectory.

Figure 14(a)-(b) shows that the optimal trajectory considering only the criteria of smoothness is the shortest one among the feasible trajectories. However, the trajectory passes very close to the obstacles, implying that the UGV has a higher probability of collision when following this trajectory. A more rational solution requires combining the safety and smoothness criteria. Figure 14(c)-(d) shows the results considering both the safety and smoothness criteria. The optimal trajectory, indicated using the red curve in Fig. 14(c), is obtained as a compromise between the safety and smoothness criteria.

4.2.3. Consistency. The properties of safety and smoothness consider only the environment information at the current step time. Thus, the current optimal trajectory is vastly different from the previous one. If this condition is frequently observed, the steering wheel could violently shake. In particular, if the response of the actuators in controlling the UGV is slow, a sudden change in trajectory may cause driving instability or a collision with obstacles.

Hence, the property of consistency is introduced, which is used to measure the difference between the current candidate trajectory and the previous optimal trajectory. The only information of the optimal trajectory is curvature of the initial point on it. We use the curvature to generate the desired steering wheel angle for motor controller. This indicates that using the difference between the curvature of the initial point on the previous optimal trajectory and the curvature of the initial point on the current feasible candidate trajectories is beneficial in representing consistency. The curvature of the initial point on the third curve can be calculated using Eq. (B6). $\vartheta_c$ represents the cost of consistency of the $i_{th}$ candidate trajectory.

$$\vartheta_i^c = \text{Norm} \left( |k_i^0 - k_p^0| \right)$$  \hspace{1cm} (11)

Here, $k_i^0$ is the curvature of the initial point on the $i$th candidate trajectory. $k_p^0$ is the curvature of the initial point on the previous OET.
4.2.4. Desired steering angle. The desired steering angle is the final output of the optimal trajectory selector. The curvature of the initial point on the optimal trajectory is an important information required to generate the desired steering angle. The kinematic model of the UGV uses the front wheel steering model of mobile robots. The relationship between front wheel steering and curvature is as follows.

$$k = \frac{\tan(\alpha)}{L}$$  \hspace{1cm} (12)

Here, $\alpha$ is steering angle, and $L$ is vehicle wheelbase.

As a general rule, the steering angle of the UGV should meet the requirement $-\frac{\pi}{2} < \alpha < \frac{\pi}{2}$. The desired wheel steering angle can be unique, calculated as follows

$$\alpha = \arctan(k \cdot r)$$  \hspace{1cm} (13)

Here, $r$ is the radius of the wheel.

4.3. Speed planner
So far, we only obtained one of the two commands, that is, desire steering angle. The UGV would receive the linear speed command for running. The speed planner considers both curvature and road safety. Speed is obtained using Eq. (14).
Here, \( r_s \) is part of road safety, and \( v_k \) is part of curvature.

First, we need to determine the maximum speed \( v_{\text{max}} \), the minimum speed \( v_{\text{min}} \), and the maximum curvature \( k_{\text{max}} \) of the UGV, which determines the performance in the given speed range. The centripetal acceleration formula can be used to obtain the distance between curvature and line speed\(^{22}\) as follows.

\[
v_k = \sqrt{\frac{a}{k}}
\]  

Here, \( a \) is the centripetal acceleration, and \( k \) is the curvature of the initial point on the current optimal trajectory. The risk of sliding is associated with centripetal acceleration. In this study, \( a = 0.3 \).

The road width and obstacles on the road were neglected when determining velocity using curvature, which are important factors affecting the velocity of UGV. The number of feasible candidate trajectories was used to represent the current safety of the road, because the candidate trajectories are arranged evenly on the local map. We can calculate the risk of the current road using the ratio of the feasible candidate trajectories to that of the total number of candidate trajectories. In this method, all the obstacles are considered to have similar importance for road safety. However, an obstacle at the center of the road is more dangerous than that on the side of the road. Because of that most of the time the UGV is near the centerline of the road. The more important obstacles should be given maximum weightage. We used a Gaussian function to generate these weights using Eq. (16).

\[
t_i = e^{-\frac{|v_i - n_o|}{2\sigma^2}}
\]
Fig. 16. Results of risk factor on road with different conditions: (a) a wide road, (b) obstacles near the edge of a wide road, (c) obstacles near the edge of a narrow road, and (d) obstacles ahead of the UGV on a narrow road.

Fig. 17. Curvature and road safety-related speed function. $r_s$ denotes the safety of the road. The higher the value of $r_s$, the safer the road will be, although the speed is high. $k_0$ is equivalent to the safety of the trajectory. The higher the value of $k_0$, the greater the probability of slide slipping by the UGV. Thus, the speed should be slower.

Here, $\rho (P_i, P_l)$ is the distance from the last control point on the candidate trajectory to the local target. $\sigma$ is the parameter to control degree distinction. We used Eq. (17) to calculate road safety $r_s$.

$$r_s = \frac{\sum_{i \in \text{feasible}} t_i}{\sum_{i=1}^{n} t_i}$$

Here, $i \in \text{feasible}$ implies that the candidate trajectory with index $i$ is feasible. There are four paradigms for estimating road safety, as shown in Fig. 16. $r_s$ is sufficiently small to obtain a safe speed when the obstacle is at the center of the road.

Now, for any given curvature and safety of the road, we can obtain the value of the corresponding speed, based on the function shown in Fig. 17.

5. Results
In this section, simulations and experiments are conducted to verify the effectiveness of the new proposed algorithms.
Fig. 18. Simulation environment diagram. $X_W O_W Y_W$ represents the world space and $X_R O_R Y_R$ is the robot (UGV) space. Both height and width of the world space equal 120 m. Both height and width of the robot space equal 30 m. The length of UGV equals 2.642 m and the width equals 1.537 m. The bottom-right white box shows scale relation between simulation environment and physical world. The red dot is the waypoint whose position is $(x, y) = (104m, 94m)$.

5.1. Simulations

Three sets of simulations are conducted to verify the effectiveness of our path planning algorithm. In all the simulations, a virtual on-road environment is firstly constructed based on the data collected from a segment of country road in physical world. The length of the road segment is about 100 m and the width of the road is 10 m or so. The environment map is obtained by using an open-source software package called Berkeley localization and mapping (https://github.com/erik-nelson/blam#berkeley-localization-and-mapping).

Figure 18 shows the simulation environment, a region of $120 \times 120$ m. The LiDAR data of the segment country road are shown in the sub-graph on the top right. The global environment map is located in the world space $X_W O_W Y_W$, and $X_R O_R Y_R$ denotes the body framework. The gray value of the figure denotes the traversability of the environment. The higher the gray value, the lower the road’s travelability. Obstacles can be easily added by directly changing certain pixels into black. The small white box represents the UGV and the red arrow shows its forward direction.

We use average OP (AOP), average curvature (AC), average consistency of curvature (ACC), and length (L [in meters]) of the final running path of the UGV to evaluate the performance of our methods. The definitions of AOP, AC, and ACC are given in Eqs. (18)–(20).

$$AOP = \frac{1}{N_1} \sum_{i=1}^{N_1} p_i$$ (18)

where $N_1$ is the number of sample points in the active area of the running path. The width of the area is 0.8 m. See Fig. 7 for more details. $p_i$ is occupancy probability at $i$th point.

$$AC = \frac{1}{N_2} \sum_{i=1}^{N_2} k_i$$ (19)

$$ACC = \frac{1}{N_2 - 1} \sum_{i=1}^{N_2-1} (k_{i+1} - k_i)$$ (20)

where $N_2$ is planning steps of the running path. $k_i$ is the desired curvature at $i$th planning step.
5.1.1. Robustness of road shape estimator. In this simulation, an inaccurate waypoint, which is shown in both Figs. 18 and 19(a), is given as the global information of the environment. The minimum distance between the waypoint and true RCL is more than 15 m. In Fig. 19, the green curves are the global RCLs estimated by our method. It should be noted that our RCL is an estimation of local road and updated at each decision period. We connect those local targets generated by local RCLs at each period to represent the global RCL estimated by our algorithm. The reference RCL is shown as brown curves on road center in Fig. 19. The reference RCL is set manually while considering the global environment map. The offset between estimated RCL by the proposed method and baseline RCL is used to measure the performance of our RSE algorithm. The offset $E_{\text{offset}}$ can be calculated by using Eq. (21).

$$E_{\text{offset}} = \frac{1}{N_{\text{step}}} \sum_{i=1}^{N_{\text{step}}} \min_{P \in \text{reference RCL}} D (P_{gi}, P)$$

where $D (\cdot, \cdot)$ is the distance operator for two points, $P_{gi}$ is the local target obtained by our RSE at $i$th running step, $P$ is the point on baseline RCL, and $N_{\text{step}}$ is the total running steps.

In this subsection, the weights of safety, smoothness, and consistency in Eq. (8) are 0.3, 0.3, and 0.4, respectively. Two scenarios are set up to explore the performance of our proposed RSE. Figure 19(a) shows the results of our RSE without any obstacles on the road surface. RCL estimated by our proposed RSE and baseline RCL almost overlap each other. $E_{\text{offset}}$ equals 0.470978 m. This result shows that the proposed RSE algorithm can handle inaccurate or wrong waypoint and estimate a precise RCL.

Figure 19(b) shows the results of our proposed RSE algorithm on road environment with four groups of obstacles. Those obstacles are zoomed in for clear observing in Fig. 19(b). The estimated RCL and baseline RCL almost overlap each other. $E_{\text{offset}}$ equals 0.530885 m, which is only 0.06 m larger than the previous scene. This result shows that the proposed RCL algorithm is insensitive to on road obstacles.

In summary, the above results show the effectivity and robustness of our proposed RSE algorithm for on-road environment with obvious road border, such as the environments shown in Figs. 6 and 18. We would stress that the purpose of our proposed RSE algorithm was to estimate the RCL and obtain the proper local target for local planning. Traversability evaluation and optimal executable trajectory selection will be considered in local planning.

5.1.2. Safety, smoothness, and consistency of the trajectory planner. In this subsection, several simulations are conducted to show how the optimizing criteria (safety, smoothness, and consistency)
Table II. Performance of different trajectory planning algorithms.

|       | AOP     | AC       | ACC     | L (m)  |
|-------|---------|----------|---------|--------|
| R     | 53.02669| 0.0667613| 0.0232196| 120.435|
| S     | 57.1263 | 0.0522559| 0.0186074| 117.216|
| C     | 56.8775 | 0.0518174| 0.0197839| 116.294|
| RSC   | 54.6705 | 0.0411102| 0.0163578| 115.656|

Fig. 20. Results of UGV running on road with obstacles. (a) UGV only considers the criterion of safety, abbreviated to R. (b) UGV only considers the criterion of smoothness, abbreviated to S. (c) UGV only considers the criterion of consistency, abbreviated to C. (d) UGV considers the criteria of safety, smoothness, and consistency, abbreviated to RSC. The black pixels on road denote the newly added obstacles, and a narrow channel is in the red ellipse. Those obstacles in the red circle are zoomed in for clear observing.

affect the performance of trajectory planner. A number of obstacles are added in environmental map by replacing a number of gray pixels with black ones as shown in Fig. 20.

Figure 20 shows the results of four different simulations. In Fig. 20(a), the trajectory planner only considers the criterion of safety marked “R,” that is, \( \omega_s = \omega_c = 0 \). Figure 20(b) and (c) is associated with the smoothness criterion (marked “S”) and consistency criterion (marked “C”), respectively. The last simulation shows the results of combining the three criteria (marked “RSC”).

In order to compare quantitatively the performance of different algorithms, the AOP, AC, ACC, and L for all simulations are computed and listed in Table II.

From Table II, we have the following results:

1) The safety criterion encourages the UGV to drive away from obstacles. But its AC and ACC are higher than others, that is, it ignores the smoothness and consistency performance as we analyzed in the Introduction.

2) AC, ACC, and L of S and C are lower than R, but their AOP is higher than R. It means that the criteria of smoothness and consistency encourage a smooth and stable driving behavior by scarifying certain kind of safety.

3) AC, ACC, and L of RSC are the smallest among the four simulations. This means that combining the criteria of safety, smoothness, and consistency encourages a smoother and more stable driving behavior than other cases. AOP of RSC is lower than S and C, but higher than R. It means that the UGV acquires more smoothness and stable driving with reduced safety.

Figure 20 applies a more intuitive understanding of how these criteria influence the planned trajectories. The running path of R passes through the center of the narrow channel in the red ellipse,
5.1.3. Safety-based speed planner. In this subsection, several simulations are conducted to show the safety of our new proposed speed planner. In these simulations, our algorithm is compared with the speed planner without considering road safety (basic). These simulations have the same environments and the same trajectory planner as that in the previous subsection. AOP, AC, ACC, and L for both the basic speed planner and our speed planner in this simulation are computed and listed in Table III.

From Table III, AOP, AC, ACC, and L of our method are all less than those of the speed planner without road safety. AOP, AC, and ACC are improved by 1.8%, 7.1%, and 10.9%, respectively. It means that considering road safety in speed planning improves the robustness metrics of UGV.

Figure 21 shows the running path of both speed planning algorithms. Figure 22 shows the velocity of UGV at each position. The green line is the velocity considering road safety and the blue line is the velocity without considering road safety factor. The four red circles highlight four regions where two algorithms present clear differences. These four regions are the same as the four red circles in Fig. 21(b). In all these regions, the value of green line is reduced by road safety factor. This means a slower running speed, which gives the UGV more time to handle the scenario.

5.1.4. Time complexity analysis. The time complexity of our path planning algorithm is analyzed in this part. Figure 23 shows the time cost of our road estimation algorithm (blue line), trajectory planning algorithm (green line), and the complete path planning algorithm (pink line). The time complexity of the algorithm is O(n), where n is the number of candidate samples. The number of candidate samples is 85 for road estimator and 75 for trajectory planning in this paper.

From Fig. 23, (1) the average time cost of road estimation is 4.378 ms at each running step; (2) the average time cost of the trajectory is 2.476 ms; (3) the average time cost of the complete path planning is 7.215 ms. Our decision cycle is equal to 100 ms, which is enough for a UGV with the maximum velocity of 10 m/s. Our total time cost is far below the decision cycle.

5.2. Experiments
To verify the performance of the proposed path planning algorithm, two experiments are conducted in country road environment. A description of our UGV platform is given in Section 2. The average usage of computational resources for sensors, perceptions, and path planning are logged and listed in Table IV.
Table IV. Usage of computational resources for each module.

|                         | 2.6 GHz CPU (4 cores) | Memory 4 Gb |
|-------------------------|------------------------|-------------|
| Sensors                 | 2%                     | 1%          |
| Perceptions             | 35%                    | 3.9%        |
| Path planning           | 8%                     | 1.4%        |

Fig. 22. The velocity of two simulations. The green line shows the velocity of speed planner with road safety, and the blue line shows the velocity of speed planner without road safety. The four red circles correspond to the four bad road segments shown in Fig. 21.

Fig. 23. The time cost of our path planning algorithm at each planning step. The horizontal axis is the running step of UGV. The vertical axis is the time cost value (in milliseconds). The pink line is the total time cost of our path planning algorithm. The blue line is the time cost of road estimation (RCL), and the green line is the time cost of trajectory planning (OET).

Figure 24 shows the actual scene and the top view of country road environment. The whole length of this road is more than 2 km and the average width of this road is about 10 m.

5.2.1. Experiment 1: country road without obstacle. The first experiment aims to test the normal road (without obstacles) driving ability of our path planning algorithm. Figure 25 shows the results of the first experiment. In Fig. 25(a), the start position (green dot), waypoints (cyan dots), goal position (red dot), branch road (yellow lines), fork points (yellow dots), and running path (cyan lines) are shown in the top view map acquired from the Google Earth. The waypoints, fork points, and goal position are directly extracted from the Google Earth map. The error between them and actual world positions are about 10 m. The whole length of the running path is approximately 2.2 km. The average velocity of the UGV is 2 m/s and the total time spent is about 20 min.

Figure 25(b) shows the actual scene wherein a UGV runs on the center of the road. Figure 25(c) shows the actual velocity of the UGV throughout the experiment. The maximum velocity approached 3.5 m/s in this experiment. This means that the corresponding curvature keeps a low value and the road safety factor is close to 1.
In summary, our path planning algorithm encourages the UGV to run in the middle of the road without obstacles. The average executive rate of path planning is near 10 Hz, which is enough for the UGV motor controller.

5.2.2. Experiment 2: country road with obstacles. The second experiment verifies the ability of our path planning algorithm in country road environment with obstacles. Figure 26 shows the actual scene of the second experiment. The effective testing distance of the road range is 100 m, and the road width is 10 m. Figure 27(a) shows the top view of this test scene. Obstacles indicated by ①, ②, and ③ are barricades. ① and ③ are two narrow channels. ⑥ is an obstacle on the center of the road. The red dot represents the goal position in this experiment. The green line shows the running path of the UGV.

Figure 27(b)–(j) shows nine photographs taken during this experiment. This nine snapshots are corresponding to the nine markers (b)–(j) in Fig. 27(a). Figure 27(b) shows the initial state of the UGV. It is ready to move from a static state at the start position. Figure 27(c) and (d) shows the UGV avoiding the first and second obstacles. The physical body of UGV is collision-free and the size of turning bend is similar as human driver. Figure 27(e) and (g) shows the running scenes in which the distance between ahead obstacles and the UGV is more than 10 m. The behavior of UGV is similar
Bézier Curves based Path Planning

Fig. 26. An actual scene of the second experiment on a road with obstacles.

Fig. 27. (a) A top view of the experiment scene with six groups of obstacles on road. (b)–(j) Actual photographs taken during the second experiment. The red dot represents the goal position in this experiment. The nine snapshots (b)–(j) correspond to the nine markers (b)–(j) in (a).

to normal road driving in the first experiment. Figure 27(f) and (i) shows the UGV passing through narrow channels. The corresponding markers (f) and (i) in Fig. 27(a) show the running path of the UGV in the middle of the channels. These results are consistent with simulation results. In Fig. 27(h), the UGV barely turns its steering angle before the obstacle. The main reason is that the obstacle marked ⃝ in Fig. 27(a) is on the left front of the UGV where the safety criterion will punish left turning. The criteria of smoothness and consistency encourage the UGV going forward. Combining safety, smoothness, and consistency, our path planning algorithm allows the UGV to go forward with enough safety. The last obstacle marked ⃣ in Fig. 27(a) is straight ahead of the UGV. The UGV may turn left or turn right repeatedly without the consistency criterion because a symmetrical candidate trajectory is observed in this scene. The inconsistencies of the connected planning cycles, which may cause collision between the UGV and ahead obstacles, didn’t appear in this experiment because we considered the consistency criterion in our path planning algorithm. The experiment shows the same
results as that of the simulations and demonstrates the viability of the whole path planning algorithm proposed in this paper in a physical UGV.

Figure 28 shows the actual velocity of the UGV in this experiment. The maximum speed for this experiment is 2.5 m/s. Nine markers (b)–(j) correspond to the nine snapshots in Fig. 27(b)–(j). The actual velocity of the UGV depends on the curvature obtained by trajectory planner and road safety. Figure 27(b) shows the UGV starts moving from a static state. Then the velocity increases to a value near the maximum speed. This is because the nearest ahead obstacle is not in the local map space. The velocity reduces with forward running of the UGV because obstacles are detected. A big steering angle is required to avoid the obstacle, which means a big curvature. So the velocity reduces to a very low value at (c) area. The same happens at (d) area. In (c) and (d) areas that are close to obstacles, both big curvature and low road safety cause the velocity to reduce. The reduction of velocity also appears in (e) area, where the UGV owns a big turning (curvature), but without obstacles (road safety factor approximately equals 1). So, velocity reduction here is smaller than that in (b) and (c) areas. The curvatures of markers (g) and (h) in Fig. 27(a) are similar to each other, but the velocity areas (g) and (h) in Fig. 28 are of different types. This is because the road safety factor of (g) is higher than that of (h), where the obstacles are on the left head of UGV. Comparing (f) and (i) with (j) shows the obstacles ahead of UGV causing a more obvious speed reduction than the case of left or right obstacles. In summary, with the proposed algorithm, the UGV presents the following characteristics: (1) the curvature and road safety factor together affect the result of speed planner; (2) a bigger curvature and a lower road safety factor means a lower desired velocity; (3) the road safety factor is more sensible to the obstacles straight ahead than those on the left or right side.

6. Conclusion
This paper presented a novel path planning method based on the layering concept and Bézier curves for a UGV driving naturally on a harsh road. Our path planning algorithm consists of three parts: road estimator, trajectory planner, and speed planner. The road estimator is based on the local map and inaccurate waypoints. The effectiveness and performance of road estimator proposed in this paper is verified with the first set of simulations. The results show that RCL estimated by our RSE algorithm enables a safer, smoother, and shorter running path for the UGV in this environment than the naïve method. Trajectory planner is based on three well-designed criteria based on OET estimation. After the second set of simulations, the results show that the performance of combining three criteria, including safety, smoothness, and consistency, is better than either of them alone. Both our road estimator and trajectory planner are based on the Bézier curves and discrete optimization scheme. In Section 5 we analyzed the time complexity (or time cost) of road estimation algorithm, trajectory planning algorithm, and the complete path planning algorithm. The results show that the time cost of our path planning algorithm is a quite low (7.215 ms). The performance of road safety-based speed planner is verified with the third set of simulations. The velocity of the UGV will reduce by a road safety factor for bad road segments. The lower velocity gives the UGV more response time to handle emergency situations. The results show that the driving behavior of the UGV improved with road safety-based speed planner. The results of our experiments demonstrate that our path planning algorithm can be used in physical experiments to ensure the safe behavior of UGV.
The results of second simulation show that combining all the three criteria can expose the robot to higher risk. In fact, all the three criteria are important for driving behavior. It is essential to find a suitable tradeoff between safety, smoothness, and consistency. In this paper, the weights of safety, smoothness, and consistency were tuned by trials, which consumed much time and required experience. The tuned weights are based on the exact models of both the UGV's behavior and its environment. Consequently, manually tuned weight has its limitations when the UGV has to adapt to new situations. In recent years, some machine learning techniques, such as inverse reinforcement learning (IRL) and inverse optimal control (IOC), have been used to help the robot learn the parameters of cost function from demonstrations. The future work is towards learning the weights of three criteria using IRL or IOC to improve the robustness of UGV.

Acknowledgment
We thank two anonymous reviewers for their helpful comments on an earlier draft of this paper. We thank Nature Sciences Foundation of China (Grant Nos.U1608253, 91748208), Chinese Academy of Sciences (Grant No. 6141A01061601), and State Key Laboratory of Robotics (No. 2017-Z07) for support.

Funding
This research is supported by the Nature Sciences Foundation of China (Grant Nos.U1608253, 91748208), Chinese Academy of Sciences (Grant No.6141A01061601), and State Key Laboratory of Robotics (No. 2017-Z07).

References
1. L. Moreno, “Navigation of mobile robots: Open questions,” *Robotica* 18, 227–234 (2000).
2. P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter and R. Siegwart, “Robot-Centric Elevation Mapping with Uncertainty Estimates,” in CLAWAR, 1–8 (2014).
3. B. W. Parkinsson, P. Enge, P. Axelrad and J. J. S. Jr, *Global Positioning System: Theory and Applications*, vol. II (American Institute of Aeronautics and Astronautics, 1996) pp. 121–175.
4. H. Qi and J. B. Moore, “Direct Kalman filtering approach for GPS/INS integration,” *IEEE Trans. Aerosp. Electron. Syst.* 38, 687–693 (2002).
5. A. Noureldin, A. El-Shafie and M. Bayoumi, “GPS/INS integration utilizing dynamic neural networks for vehicular navigation,” *Inf. Fusion* 12, 48–57 (2011).
6. P. Chand and D. A. Carnegie, “A two-tiered global path planning strategy for limited memory mobile robots,” *Rob. Auton. Syst.* 60, 309–321 (2012).
7. F. Yu, H. Tien-Ruey and C. Sheng-Luen, “Multi-waypoint visual homing in piecewise linear trajectory,” *Robotica* 31, 479–491 (2013).
8. Y. Ma, G. Zheng, W. Perruquetti and Z. Qiu, “Local path planning for mobile robots based on intermediate objectives,” *Robotica* 33, 1017–1031 (2015).
9. M. Hank and M. Hadid, “A hybrid approach for autonomous navigation of mobile robots in partially-known environments,” *Rob. Auton. Syst.* 86, 113–127 (2016).
10. M. Bajracharya, A. Howard, L. H. Matthis, B. Tang and M. Turmon, “Autonomous off-road navigation with end-to-end learning for the LAGR program,” *J. Field Rob.* 26, 3–25 (2009).
11. M. Buehler, K. Iagnemma and S. Singh, *The DARPA Grand Challenge: The Great Robot Race* (Springer Publishing Company, Incorporated, 2007) pp. 1–45.
12. M. Buehler, K. Iagnemma and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic* vol. 56, (Springer Publishing Company, Incorporated, 2009) pp. 1–1.
13. S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian and P. Mahoney, “Stanley: the robot that won the DARPA grand challenge,” *J. Field Rob.* 23, 661–692 (2006).
14. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittelman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y. W. Seo, S. Singh, J. Snider, A. Stentz, W. “Red” Whittaker, Z. Woikovicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms and D. Ferguson, “Autonomous driving in urban environments: Boss and the urban challenge,” *J. Field Rob.* 25, 425–466 (2008).
15. M. Montemerlo, S. Bhat, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marci, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pfueger, G. Stanek, D. Stavens, A. Vogt and S. Thrun, “Junior: The stanford entry in the urban challenge,” *J. Field Rob.* 25, 569–597 (2009).
Appendix A. Vehicle model and coordinate system

Figure A1 shows the vehicle model and its coordinate systems used in this paper. The UGV is represented by a rectangle located on a 2D surface for which an inertial frame (world coordinate system $\{X_W, O_W, Y_W\}$) is defined. The UGV has three degrees of freedom on the surface. The posture ($P_i$) of the UGV at any given instant $i$ constitutes its 2D position $(x_i, y_i)$ and heading angle $(\psi_i)$. Here $x_i$ and $y_i$ are the coordinates of the UGV’s body center in the world coordinate system. The local frame (robot coordinate system $\{X_R, O_R, Y_R\}$) is also constructed. The heading direction of the UGV is X-axis of local frame. The heading angle $(\psi_R)$ of the UGV in inertial frame is the angle between $X_W$ and $X_R$. It should be noted that the posture of UGV always equals $(0, 0, 0)$ in robot coordinate system. A coordinate of a message can be conveniently transformed into different coordinate systems with TF, a ROS package.
Bézier Curves based Path Planning

Fig. A1. Vehicle mode and its coordinate systems.

The states of the UGV can be refined to 2D position \((x, y)\), heading \(\psi\), and curvature \(\kappa\). These states can be related by vehicle kinematic model and Ackerman steering mechanism. Let \(\kappa, v, t, s\) denote the curvature, linear velocity, time, and distance traveled, respectively. Then, the curvature, turning radius, wheelbase, and steering angle can be related by (see ref. [24]):

\[
\kappa \triangleq \frac{1}{r} = \frac{\tan(\alpha)}{L}
\]

(A1)

As a typical nonholonomic system, the kinematics of the UGV can be expressed by (see ref. [28]):

\[
\begin{align*}
\dot{\psi}(t) &= v(t) \kappa(t) \\
\dot{x}(t) &= v(t) \cos(\psi(t)) \\
\dot{y}(t) &= v(t) \sin(\psi(t))
\end{align*}
\]

(A2)

Appendix B. Bézier curve

The Bézier curve, a smooth path with continuous curvature, is defined by a sequence of points \(P_0, \ldots, P_n\), where \(n\) is called its degree. The first and last points are on the curve. However, the intermediate points (if any) generally do not lie on the curve, but they control the shape of the resulting curve.

A general definition of the Bézier curve with \(n\) degree is:

\[
\vec{B}(t) = \sum_{i=0}^{n} \binom{n}{i} \vec{P}_i (1-t)^{n-i}t^i, \quad t \in [0,1]
\]

(B1)

where \(\vec{B}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}\), \(\vec{P}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}\), \(x_i, y_i\) are the corresponding values of coordinates \(X\) and \(Y\).

A second Bézier curve is:

\[
\vec{B}(t) = \vec{P}_0(1-t)^2 + 2\vec{P}_1t(1-t) + \vec{P}_2t^2, \quad t \in [0,1]
\]

(B2)

Differentiating Eq. (B2) by \(t\), we have:

\[
\frac{\partial \vec{B}(t)}{\partial t} = 2(t-1)\vec{P}_0 + 2(1-2t)\vec{P}_1 + 2t\vec{P}_2, \quad t \in [0,1]
\]

(B3)

An explicit form of the third Bézier curve is:

\[
\vec{B}(t) = \vec{P}_0(1-t)^3 + 3\vec{P}_1t(1-t)^2 + 3\vec{P}_2t^2(1-t) + \vec{P}_3t^3, \quad t \in [0,1]
\]

(B4)
Bézier Curves based Path Planning

The points on a cubic Bézier curve are given by:

\[ k(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'^2(t) + y'^2(t))^2} \]  

(B5)

The curvature of the initial point on a cubic Bézier curve is:

\[ k(0) = \frac{2}{3} \cdot \frac{(\vec{P}_1 - \vec{P}_0) \times (\vec{P}_2 - \vec{P}_1)}{|\vec{P}_1 - \vec{P}_0|^3} \]  

(B6)