electronics

MDPI

Article

Spatial Path Smoothing for Car-like Robots Using Corridor-Based Quadratic Optimization

Yongkang Lu, Yuanqing Wu, Wenjian Zhong, Yanzhou Li and Meng Chen *

School of Automation, Guangdong University of Technology, Guangzhou 510006, China
* Correspondence: mengc3070@163.com

Abstract: The global planning module of car-like robots usually plans a coarse spatial path, which may be non-smooth and kinematically infeasible for car-like robots. This study proposes an efficient spatial path smoothing approach, which is capable of optimizing a rough spatial path to be a high quality one. Two novel designs contribute to the proposed approach. One is a direct corridor construction method that provides an optimization region for path optimization. Based on a redefined path representation in the generated corridor, the second one is a core spatial path optimization method, where the optimization problem is formulated as a multi-objective quadratic programming (QP) with corridor and maximum-curvature constraints. Meanwhile, integrating a two-steps strategy into an optimization process yields a good trade-off between efficiency and quality. Experiment results validate that the proposed approach has the ability to online generate a high quality path.

Keywords: car-like robots; path smoothing; multi-objective optimization; quadratic programming

Citation: Lu, Y.; Wu, Y.; Zhong, W.; Li, Y.; Chen, M. Spatial Path Smoothing for Car-like Robots Using Corridor-Based Quadratic Optimization. *Electronics* **2023**, *12*, 819. https://doi.org/10.3390/electronics12040819

Academic Editors: He Cai and Zhiyun Lin

Received: 4 November 2022
Revised: 13 January 2023
Accepted: 13 January 2023
Published: 6 February 2023

1. Introduction

In the last decades, as the technology of autonomous navigation achieving huge progress, mobile vehicles with autonomous navigation system have been applied in various scenarios, such as floor cleaning, industrial inspection and cargo delivery. To handle the various tasks, mobile vehicles are expected to have the ability to avoid collision with surrounding objects [1]. As an essential module of autonomous navigation system, the planning module plays an important role in guaranteeing the safe and comfort vehicle motion by generating a smooth and collision-free path from the starting state to the terminal state [2,3]. In practical application, the planned path is also required to satisfy kinematic constraints of the vehicle. All these considerations make path planning for self-driving system be more challenging work.

A typical path planning framework mainly involves two parts: path finding and trajectory generation. Most of the path finding algorithms such as search-based methods [4,5], sampling-based methods [6,7] and others [8], can work well in collision-free path finding. However, the planned results are rough paths, which consist of straight lines and sharp turns, and therefore may violate certain constraints such as continuity and vehicle kinematics. A common method for continuous path generation uses parametric curves, such as polynomial [9], spline [10] and Bézier [11] curves. However, the kinematic feasibility of the generated path cannot be guaranteed by working with these curves directly [12]. The other pipeline for trajectory generation is to formulate an nonlinear optimization problem considering continuity, safety and vehicle kinematic. Due to the limited capability of trajectory optimization, a rough path may increase the optimization difficulty and results in a low quality trajectory. In addition, many methods formulate trajectory planning as nonlinear and nonconvex optimization problems, which cannot guarantee that the optimization converges to the optimal solution.

Our goal is to design a convex optimization approach considering the kinematic constraints of car-like robots for spatial path optimization. In our design, the path optimization

problem is decomposed into spatial corridor generation and corridor-based path optimization. Firstly, a direct and efficient corridor generation method is proposed to construct a feasible optimization corridor along the reference path. This corridor is used to be the convex optimization domain in which the optimal solution can be found Secondly, on the basis of the generated corridor, multi-objective optimization cost function with a quadratic form is designed by introducing a new path point representation. Meanwhile, in order to guarantee a kinematically feasible path, our method incorporates corridor constraints and maximum-curvature constraints. The maximum-curvature constraints are formulated as approximate linear constraints. Finally, a two-steps optimization strategy is designed to solve the problem effectively. The contributions of this article are summarized as follows:

1. A simple and efficient spatial corridor generation method for spatial corridor extraction is proposed. The problem of path optimization based on the generated spatial corridor are developed as quadratic forms.
2. A convex multi-objective path optimization problem with corridor and maximum-curvature constraints is designed to generate a smooth and kinematically feasible path. The overall problem of path optimization is developed as a quadratic programming problem, which can be quickly solved. Meanwhile, a two-steps optimization strategy is designed to balance the optimization efficiency and quality.
3. Our proposed method can be utilized to optimize the path in various scenarios. In addition, the experimental results demonstrate that our approach has a good performance in operational efficiency.

2. Related Work

The traditional path finding methods, such as search-based methods [4,5] and sampling-based methods [6,7], mainly focus on the efficiency or the optimality of finding a collision-free path from the starting position to the destination, but fail to concern the path feasibility for car-like robots. The planning results of these methods are usually rough and can not be used directly to navigate robots. To fill this gap, some methods post-process searching primitives via model predictive control methods to guarantee the kinematical feasibility of planning the results [13,14]. Inheriting the idea of the A* algorithm and introducing a model-based node expansion strategy can plan a vehicle-drivable trajectory in [15,16]. The variant of the rapidly exploring random tree is reported in [17], which considers kinematics constraints of robots in the searching process to enhance the feasibility of the planned path. However, several aspects of the planning objectives can not be fully considered in these methods, including length, smoothness and kinematical feasibility.

To pursue a high quality path, lots of results with regard to path optimization have been reported in recent years. Some of the works formulate their path optimization problem based on the pre-built corridor around the initial path. The reference [18] segments the safe region into the straight regions and corner region along the initial path. Convex polygons, then, are constructed within those segment regions and form a corridor jointly. A novel corridor structure is proposed in [19], namely, the spatial-temporal semantic corridor. This structure provides a representation for various kinds of semantic elements in the Frenét frame, extracting a free-space corridor for safe trajectory generation. Both the works in [18,19] can find collision-free and continuous paths by constraining the control points of each segment Bézier curve inside the corridor using the convex hull property of the Bézier curve. However, the feasibility of the generating paths in these methods cannot be guaranteed due to the lack of taking vehicle kinematic constraints into account. To deal with this problem, based on the corridor-based optimization method, ref. [20] introduces an approximate method to enforce kinematic constraints, which can find a kinematically-feasible path while ensuring the convexity of the optimization problem. The final optimization problem is formulated as a quadratic programming problem with quadratic constraints, and solved in several milliseconds, on average.

Path optimization is usually a multi-objective nonlinear optimization problem with various constraints [21–27]. Many path optimization problems are usually formulated as

non-convex problems [28] and therefore have the following shortcomings: (i) optimization results may highly depend on the initial guess, (ii) optimization problems may easily stuck in local optimal solutions, and (iii) optimization problems may not be solved in a computationally efficient way [29]. To avoid these shortcomings, some works develop path optimization problems as convex problems, which can be solved by efficient techniques [29–31]. Thanks to the convexity of the problem, the global optimal solution can be found within finite iterations. Moreover, the problem with quadratic functions and linear constraints is a special case of convex problems, which is called the QP problem. In [25], based on the left and right side bound of the road map, a quadratic path length cost function is designed to minimize the required lap time. To find a path with the best trade-off between shortest length and minimum-curvature, approaches in [18,26], first give quadratic objective functions to find the shortest path and minimum-curvature path independently, and then find the best trade-off between the shortest path and minimum-curvature path. However, these methods focus on achieving the lowest lap time in the racing game, which may be not suitable for other scenarios. The method in [27] solves a trajectory optimization problem using sequential QP. In [20,32], an approximate method to enforce kinematic constraints is introduced to generate a kinematically feasible path using sequential QP. However, introducing kinematic constraints at the very start of the optimization process is not a good optimization strategy to quickly find a solution. This is because, in some cases, the optimization results are already kinematically feasible without introducing kinematic constraints. In addition, introducing kinematic constraints will increase the complexity of the optimization problem.

3. Spatial Corridor Generation

In this section, a simple and efficient method focusing on computing the corridor boundary points along the reference path is designed to generate a spatial corridor. The computed boundary points are used to represent the corridor, which are suited for the establishment of the path optimization problem.

Corridor Boundary Generation

Given a initial path $\mathbf{P}^{\text{init}} = \{P_1^{\text{init}}, P_2^{\text{init}}, ..., P_m^{\text{init}}\}$, this subsection computes the reference path $\mathbf{P}^{\text{ref}} = \{P_1^{\text{ref}}, P_2^{\text{ref}}, ..., P_n^{\text{ref}}\}$ and corridor boundaries represented by $\mathbf{P}^{\text{L}} = \{P_1^{\text{L}}, P_2^{\text{L}}, \ldots, P_n^{\text{L}}\}$ and $\mathbf{P}^{\text{R}} = \{P_1^{\text{R}}, P_2^{\text{R}}, \ldots, P_n^{\text{R}}\}$, as shown in Figure 1. A point has two dimension information $P_i^{\text{j}} = [x_i^j, y_i^j]^T$. We directly compute the slope k_i of the line segment between P_i^{ref} and P_{i+1}^{ref}. The corridor region is stretched along the reference path laterally by computing the perpendicular line of the corresponding line segment. Given the corridor lateral stretching distance parameters $\mathbf{d}^{\text{L}} = \{d_1^{\text{L}}, d_2^{\text{L}}, ..., d_n^{\text{L}}\}$ and $\mathbf{d}^{\text{R}} = \{d_1^{\text{R}}, d_2^{\text{R}}, ..., d_n^{\text{R}}\}$, the corridor boundary points can be computed by:

$$\begin{aligned} x_i^{\text{L}} &= x_i^{\text{ref}} + h\frac{d_i^{\text{L}}}{\sqrt{{k_i}^2+1}}, \\ x_i^{\text{R}} &= x_i^{\text{ref}} - h\frac{d_i^{\text{R}}}{\sqrt{{k_i}^2+1}}, \\ y_i^j &= k_i(x_i^j - x_i^{\text{ref}}) + y_i^{\text{ref}}, \quad j \in \{\text{L}, \text{R}\}, \end{aligned} \tag{1}$$

where k_i represents the slope of the perpendicular of i-th line segment and is computed by $k_i = (y_{i+1}^{\text{ref}} - y_i^{\text{ref}})/(x_{i+1}^{\text{ref}} - x_i^{\text{ref}})$. h is used to distinguish P_i^{L} and P_i^{R}, defined as follows:

$$h = \begin{cases} -1, & \vec{v}_{1,i} \times \vec{v}_{2,i} > 0 \\ 1, & \vec{v}_{1,i} \times \vec{v}_{2,i} < 0, \end{cases} \tag{2}$$

with

$$\begin{aligned}\vec{v}_{1,i} &= [x_i^{\mathrm{ref}} - x_i^{\mathrm{L}}, y_i^{\mathrm{ref}} - y_i^{\mathrm{L}}],\\ \vec{v}_{2,i} &= [x_{i+1}^{\mathrm{ref}} - x_i^{\mathrm{ref}}, y_{i+1}^{\mathrm{ref}} - y_i^{\mathrm{ref}}].\end{aligned} \tag{3}$$

Noted that, in practice, a safe optimization region is guaranteed by pushing the corridor into the safe region of the road map. The size of corridor region is determined by $\mathbf{P}^{\mathrm{L}}$ and $\mathbf{P}^{\mathrm{R}}$. We only need to modify $\mathbf{P}^{\mathrm{L}}$ and $\mathbf{P}^{\mathrm{R}}$ to adjust the corridor.

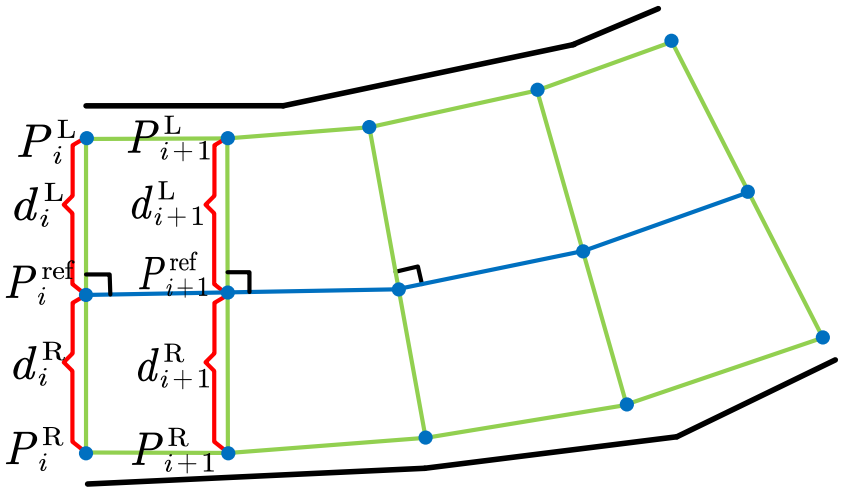
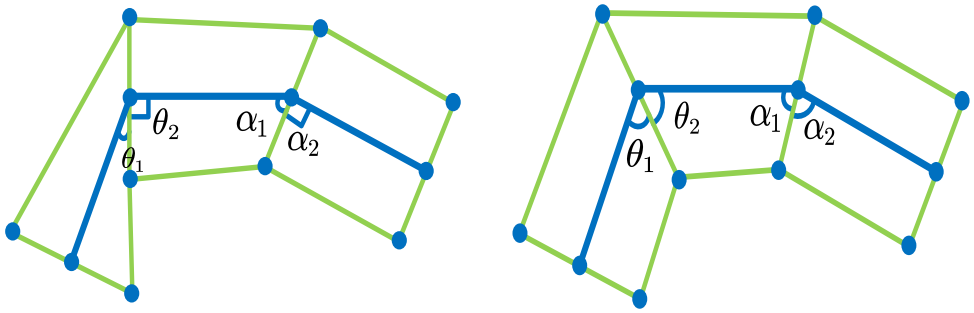


Figure 1. An illustration of spatial corridor. The black lines represent the road boundaries. The green grid represents the generated corridor. The blue line represents the reference path.

Algorithm 1 shows the process of the proposed spatial corridor generation. The algorithm will begin with a linear interpolation for generating a path with uniformly and densely distributed waypoints, if there is a path with the sparse path point distribution. Then, we compute the segment line slope (Line 4 in Algorithm 1) and its vertical line slope (Line 5 in Algorithm 1) for extracting spatial corridor. There are n vertical line slope parameters computed in Algorithm 1 from line 3 to line 10. It is worth noting that the function (Line 11 in Algorithm 1) is introduced to reconstruct the local segment of the corridor so that a reasonable corridor along path can be obtained. An example for local corridor reconstruction is shown in Figure 2. Based on the final slope set $\mathbf{k}^{\mathrm{smooth}}$, boundary points are computed (Line 12–17 in Algorithm 1).



(**a**) (**b**)

Figure 2. (**a**) The corridor is obtained in the case of the turn, with $\theta_2 = 90°$ and $\alpha_2 = 90°$. The generated corridor in this case extracts a non-ideal region. In (**b**), an ideal corridor is constructed by readjusting the slopes of horizontal lines, in which $\theta_1 = \theta_2$ and $\alpha_1 = \alpha_2$.

Algorithm 1 Spatial Corridor Generation.

Input: $\mathbf{P}^{\text{init}}$, point set of the initial reference path; $\mathbf{d}^{\text{L}}$, $\mathbf{d}^{\text{R}}$, the set of corridor width parameters

Output: $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$, boundary points of spatial corridor

$\mathbf{k} \leftarrow \emptyset, \mathbf{k}^{\text{smooth}} \leftarrow \emptyset, \mathbf{P}^{\text{L}} \leftarrow \emptyset, \mathbf{P}^{\text{R}} \leftarrow \emptyset$
$\mathbf{P}^{\text{ref}} \leftarrow$PathInterpolation($\mathbf{P}^{\text{init}}$)
for $i = 1$ **to** $n - 1$ **do**
    $k_i \leftarrow$ ComputeLineSlope(P_i^{ref}, P_{i+1}^{ref})
    $k_i^{'} \leftarrow 1/k_i$
    $\mathbf{k} \leftarrow \mathbf{k} \cup k_i^{'}$
end for
$k_n \leftarrow$ ComputeLineSlope(P_{n-1}^{ref}, P_n^{ref})
$k_n^{'} \leftarrow 1/k_n$
$\mathbf{k} \leftarrow \mathbf{k} \cup k_n^{'}$
$\mathbf{k}^{\text{smooth}} \leftarrow$ ComputeSmoothSlopes($\mathbf{k}$)
for $i = 1$ **to** n **do**
    $P_i^{\text{L}} \leftarrow$ ComputeLeftBoundaryPoint(k_i^{smooth}, P_i^{ref}, d_i^{L})
    $P_i^{\text{R}} \leftarrow$ ComputeRightBoundaryPoint(k_i^{smooth}, P_i^{ref}, d_i^{R})
    $\mathbf{P}^{\text{L}} \leftarrow \mathbf{P}^{\text{L}} \cup P_i^{\text{L}}$
    $\mathbf{P}^{\text{R}} \leftarrow \mathbf{P}^{\text{R}} \cup P_i^{\text{R}}$
end for
return $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$

4. Path Optimization

The goal of path optimization is to optimize a discrete path to be a high quality path. To formulate the path optimization problem, firstly, we describe the path point representation based on the generated corridor. Secondly, we discuss quadratic objective functions and linear constraints of path optimization to jointly construct a QP problem. Finally, a two-steps optimization strategy is designed to quickly find a feasible solution.

4.1. Path Point Representation

Given the reference path, instead of directly taking the path points as the optimization variables, similar to [15], we use scale factors to identify the path points and take these factors as the optimization variables. Optimized points can be obtained by these factors, once the optimization procedure is completed. The representation of a path point within the generated corridor is described as follows:

$$P_i = P_i^{\text{L}} + \rho_i (P_i^{\text{R}} - P_i^{\text{L}}), \quad i = \{1, ..., n\} \tag{4}$$

where $\rho_i \in [0, 1]$ is the factor to identify point P_i within the line segment between P_i^{L} and P_i^{R}. With this path point representation, we only need to use ρ_i to represent a path point in the optimization process. Moreover, for a path with n waypoints, an optimization problem directly optimizing the x-direction and y-direction depends on $2n$ variables. Based on the above discussion, only n factors are optimized.

4.2. Multi-Objective Path Optimization

4.2.1. Path Length Optimization

A coarse reference path often includes redundant displacement due to unreasonable path shapes. An evaluation term for the computing squared euclidean distance between

adjacent path points is used to minimize the path length. The objective function for path length minimization is formulated as:

$$\begin{aligned} J_l &= \sum_{i=1}^{n-1} \|P_{i+1} - P_i\|^2 \\ &= \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2. \end{aligned} \tag{5}$$

Let $\Delta P_{x,i} = x_{i+1} - x_i$, $\Delta P_{y,i} = y_{i+1} - y_i$, $\Delta x_i = x_i^{\mathrm{R}} - x_i^{\mathrm{L}}$ and $\Delta y_i = y_i^{\mathrm{R}} - y_i^{\mathrm{L}}$. According to the path point representation in Equation (4), here, we consider the case of x-direction. Then, $\Delta P_{x,i}$ can be rewritten as:

$$\begin{aligned} \Delta P_{x,i} = &\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_i & 0 \\ 0 & \Delta x_{i+1} \end{bmatrix} \begin{bmatrix} \rho_i \\ \rho_{i+1} \end{bmatrix} \\ &+ \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} x_i^{\mathrm{L}} \\ x_{i+1}^{\mathrm{L}} \end{bmatrix}, \end{aligned} \tag{6}$$

where x_i^R and x_i^L are boundary points information given by the generated spatial corridor. Moreover, the path length minimization problem can be viewed as a quadratic form:

$$(\Delta P_{x,i})^2 = \bar{\rho}_i^T H_{l,x,i} \bar{\rho}_i + B_{l,x,i} \bar{\rho}_i + \text{constant}, \tag{7}$$

where $\bar{\rho}_i = [\rho_i \ \ \rho_{i+1}]^T$, while $H_{l,x,i}$ and $B_{l,x,i}$ are constant matrices. For y-direction, we have the same form:

$$(\Delta P_{y,i})^2 = \bar{\rho}_i^T H_{l,y,i} \bar{\rho}_i + B_{l,y,i} \bar{\rho}_i + \text{constant}, \tag{8}$$

Finally, the total quadratic form is a combination of Equations (7) and (8):

$$J_l = \sum_{i=1}^{n-1} \bar{\rho}_i^T (H_{l,x,i} + H_{l,y,i}) \bar{\rho}_i + (B_{l,x,i} + B_{l,y,i}) \bar{\rho}_i + \text{constant}. \tag{9}$$

4.2.2. Path Smoothness Optimization

It is desirable to generate a smooth path for the vehicle to track so that the vehicle can run comfortably and steadily. In order to smooth the path, an elastic band cost function [33] is used to penalize the path smoothness:

$$\begin{aligned} J_s &= \sum_{i=1}^{n-2} \|(P_{i+2} - P_{i+1}) - (P_{i+1} - P_i)\|^2 \\ &= \sum_{i=1}^{n-2} \|P_{i+2} - 2P_{i+1} + P_i\|^2 \\ &= \sum_{i=1}^{n-2} \|A_i\|^2, \end{aligned} \tag{10}$$

where

$$\begin{aligned} A_i = &\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_i & 0 & 0 \\ 0 & \Delta x_{i+1} & 0 \\ 0 & 0 & \Delta x_{i+2} \end{bmatrix} \begin{bmatrix} \rho_i \\ \rho_{i+1} \\ \rho_{i+2} \end{bmatrix} \\ &+ \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_i^{\mathrm{L}} \\ x_{i+1}^{\mathrm{L}} \\ x_{i+2}^{\mathrm{L}} \end{bmatrix}. \end{aligned} \tag{11}$$

The designed cost function is intended to shorten the distance among three adjacent points along the reference path. The shorter distance, the more likely these three adjacent points are in a straight line. The smoothness of the path with the straight line shape is

guaranteed naturally. Therefore, curvature of the path can also be minimized by this cost function. Generally, the path tracking module of a self-driving car is prefered to a path with lower curvature due to vehicle kinematic restrictions.

For further consideration, an elastic band cost function in [34] is adopted to penalize the jerk of the B-spline for minimizing the curve smoothness. This cost function takes account of four adjacent points. In this paper, we used the similar term to smooth the path further:

$$\begin{aligned} J_j &= \sum_{i=1}^{n-3} \|(P_{i+3} - 2P_{i+2} + P_{i+1}) - (P_{i+2} - 2P_{i+1} + P_i)\|^2 \\ &= \sum_{i=1}^{n-3} \|P_{i+3} - 3P_{i+2} + 3P_{i+1} - P_i\|^2 \\ &= \sum_{i=1}^{n-3} \|S_i\|^2, \end{aligned} \tag{12}$$

where

$$\begin{aligned} S_i = &\begin{bmatrix} -1 \\ 3 \\ -3 \\ 1 \end{bmatrix}^T \begin{bmatrix} \Delta x_i & 0 & 0 & 0 \\ 0 & \Delta x_{i+1} & 0 & 0 \\ 0 & 0 & \Delta x_{i+2} & 0 \\ 0 & 0 & 0 & \Delta x_{i+3} \end{bmatrix} \begin{bmatrix} \rho_i \\ \rho_{i+1} \\ \rho_{i+2} \\ \rho_{i+3} \end{bmatrix} \\ &+ \begin{bmatrix} -1 \\ 3 \\ -3 \\ 1 \end{bmatrix}^T \begin{bmatrix} x_i^{\mathrm{L}} \\ x_{i+1}^{\mathrm{L}} \\ x_{i+2}^{\mathrm{L}} \\ x_{i+3}^{\mathrm{L}} \end{bmatrix}. \end{aligned} \tag{13}$$

Define $\hat{\rho}_i = [\rho_i, \rho_{i+1}, \rho_{i+2}]^T$ and $\tilde{\rho}_i = [\rho_i, \rho_{i+1}, \rho_{i+2}, \rho_{i+3}]^T$. Both cost functions J_s and J_j are represented as quadratic forms:

$$J_s = \sum_{i=1}^{n-1} \hat{\rho}_i^T (H_{s,x,i} + H_{s,y,i}) \hat{\rho}_i + (B_{s,x,i} + B_{s,y,i}) \hat{\rho}_i + \text{constant}, \tag{14}$$

$$J_j = \sum_{i=1}^{n-1} \tilde{\rho}_i^T (H_{j,x,i} + H_{j,y,i}) \tilde{\rho}_i + (B_{j,x,i}^T + B_{j,y,i}^T) \tilde{\rho}_i + \text{constant}, \tag{15}$$

where $H_{s,x,i}$, $H_{s,y,i}$, $B_{j,x,i}$ and $B_{j,y,i}$ are the known parameters.

4.2.3. Path Deviation Optimization

Generally, the given reference path is safe and reliable. It is desirable that the optimized path keeps the shape of the reference path. For this, the sum of deviation between optimized points and reference points along the path is considered to minimize the deviation. As previously noted, the reference path points are taken as the middle points of the corridor to generate corridor boundaries. A reference path point in the corridor is represented as $g(i) = (P_i^L + 0.5(P_i^R - P_i^L))$. The cost function of reference path deviation is measured as:

$$\begin{aligned} J_d &= \sum_{i=1}^{n} \|P_i - g(i)\|^2 \\ &= \sum_{i=1}^{n} \|D_i\|^2, \end{aligned} \tag{16}$$

where

$$D_i = \left[-\Delta x_i\right] \left[\rho_i\right] + 0.5 \Delta x_i. \tag{17}$$

Then, J_d can be rewritten as:

$$J_d = \sum_{i=1}^{n} \rho_i^T (H_{d,x,i} + H_{d,y,i})\rho_i + (B_{d,x,i} + B_{d,y,i})\rho_i + \text{constant}. \tag{18}$$

where $H_{d,x,i}$, $H_{d,y,i}$, $B_{d,x,i}$ and $B_{d,y,i}$ are constants.

The overall objective function is a linear combination of the cost functions of path length, smoothness and deviation, given as:

$$J_{total} = \omega_l J_l + \omega_s J_s + \omega_j J_j + \omega_d J_d, \tag{19}$$

where ω_l, ω_s, ω_j and ω_d are the corresponding weighting parameters.

4.3. Optimization Constraints

The constraints for path optimization in this paper are separated into two classes, including corridor and kinematic constraints. The corridor constraints are to constrain the optimization domain of the problem into the corridor region, i.e., paths generated in the iteration optimization process must be inside the corridor. To satisfy vehicle kinematic constraints, we consider the maximum turning radius of the vehicle and introduce maximum curvature constraints to limit the curvature of the path during optimization.

4.3.1. Corridor Boundary Constraints

Since the vehicle always passes the starting and ending points, these two boundary points are directly fixed without modification. That is, the boundary constraints are viewed as affine equality constraints. We introduce boundary constraints by fixing scalar factors of two boundary points:

$$\begin{aligned} \rho_1 &= \rho_{\text{start}}, \\ \rho_n &= \rho_{\text{goal}}, \end{aligned} \tag{20}$$

where ρ_{start} and ρ_{goal} represent the boundary factors of starting and ending points, respectively.

4.3.2. Corridor Internal Constraints

The feasibility of the path is guaranteed by enforcing each path segment inside the corresponding feasible domain. That is, the corridor internal constraints give the certain optimization scope to find the optimal solution within the corridor. These constraints are classified as affine inequality constraints:

$$\begin{aligned} 0 + \varepsilon_2 \leqslant &\rho_2 \leqslant 1 - \varepsilon_2, \\ 0 + \varepsilon_3 \leqslant &\rho_3 \leqslant 1 - \varepsilon_3, \\ &\vdots \\ 0 + \varepsilon_{n-1} \leqslant &\rho_{n-1} \leqslant 1 - \varepsilon_{n-1}, \end{aligned} \tag{21}$$

where $\varepsilon_i \in [0, 0.5]$ and $i = 2, 3, \ldots, n-1$. ε_i is the constant parameter to adjust the optimization domain.

4.3.3. Curvature Constraints

Curvature constraints play an important role in guaranteeing the feasibility of the path kinematically. $R_{\min}$ is defined as the minimum turning radius, which is given by the vehicle's physical restrictions. The maximal curvature can be represented as $\kappa_{\max} = 1/R_{\min}$. The following descriptions are based on an assumption that each path point P_i is uniformly and densely distributed over the path. Thus, with the illustration in Figure 3, several approximate formulas are obtained as follows:

$$\Delta s \approx d, \tag{22}$$

$$\begin{aligned} \sin(\alpha) &= \frac{\Delta s}{R} \approx \frac{d}{R}, \\ \sin(\alpha) &\approx \alpha \end{aligned} \tag{23}$$

where $\Delta s \approx d = \sum_{i=1}^{n-1} \|P_{i+1}^{\text{ref}} - P_i^{\text{ref}}\|/(n-1)$. To evaluate the path curvature, we first compute the following formulas according to the illustration in Figure 3:

$$\begin{aligned} \overrightarrow{\mu} &= \overrightarrow{P_{i+1}P_i} + \overrightarrow{P_{i+1}P_{i+2}}, \\ \|\overrightarrow{\mu}\| &= \|\overrightarrow{P_{i+1}P_i} + \overrightarrow{P_{i+1}P_{i+2}}\|, \\ &= \|P_i + P_{i+2} - 2P_{i+1}\|, \end{aligned} \tag{24}$$

$$\|\overrightarrow{\mu}\| = 2d\sin(\frac{\alpha}{2}) \approx \Delta s^2 \kappa, \tag{25}$$

where κ is the curvature and $\kappa \leq \kappa_{\max}$. Equation (25) is used to approximately compute $\|\overrightarrow{\mu}\|$. Thus, we have:

$$\|\overrightarrow{\mu}\| \leq \Delta s^2 \kappa_{\max}, \tag{26}$$

i.e.,

$$\|P_i + P_{i+2} - 2P_{i+1}\| \leq \Delta s^2 \kappa_{\max}. \tag{27}$$

The square of Equation (27) is used to limit the curvature:

$$\|P_i + P_{i+2} - 2P_{i+1}\|^2 \leq (\Delta s^2 \kappa_{\max})^2. \tag{28}$$

The form of curvature constraint in Equation (28) is a nonlinear inequality. Directly introducing this constraint will make the optimization problem difficult to be solved. To full this gap, the Taylor approximation method retaining first-order item is used to linearize the curvature constraint in this paper. Denote $F(\xi_i) = \|P_i + P_{i+2} - 2P_{i+1}\|^2$, where $\xi_i = [P_i, P_{i+1}, P_{i+2}]^T$. One has:

$$F(\xi_i) \approx F(\xi_i^{\text{ref}}) + F'(\xi_i^{\text{ref}})(\xi_i - \xi_i^{\text{ref}}), \tag{29}$$

where ξ_i^{ref} can be obtained from the reference path. Denote E_i as the constant item of Equation (29) and rewrite $F(\xi_i)$ as:

$$F(\xi_i) = F'(\xi_i^{\text{ref}})\xi_i + E_i. \tag{30}$$

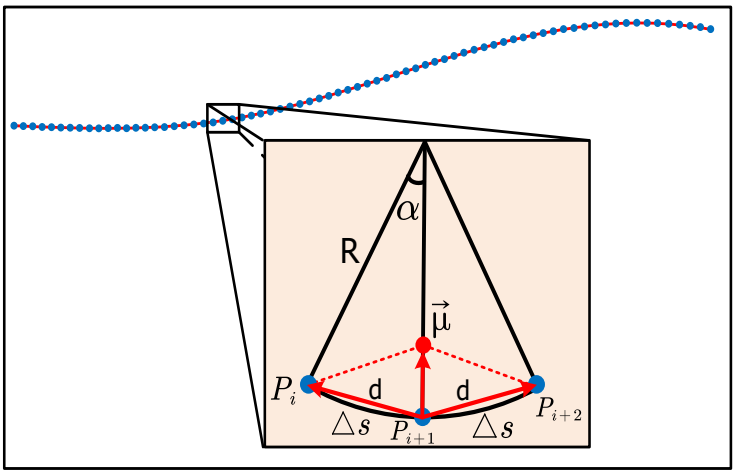


Figure 3. Illustration of linearized procedure of curvature constraint.

Equation (28) can be rewritten as:

$$0 \leqslant F(\xi_i) \leqslant (\Delta s^2 \kappa_{\max})^2, \tag{31}$$

Combining Equations (30) and (31), we have:

$$-E_i \leqslant F^{'}(\xi_i^{\mathrm{ref}})\zeta_i \leqslant (\Delta s^2 \kappa_{\max})^2 - E_i, \tag{32}$$

It is worth mentioning that optimization variables of the convex problem in this paper are the scalar factors . According to the path point representation in Equation (4), one has:

$$-E_i \leqslant B_i\tilde{\rho}_i + G_i \leqslant (\Delta s^2 \kappa_{\max})^2 - E_i, \tag{33}$$

where B_i and G_i are constants. Accordingly, the curvature constraints can be formulated as:

$$\begin{aligned} -E_1 - G_1 &\leqslant B_1\tilde{\rho}_1 \leqslant (\Delta s^2 \kappa_{\max})^2 - E_1 - G_1, \\ -E_2 - G_2 &\leqslant B_2\tilde{\rho}_2 \leqslant (\Delta s^2 \kappa_{\max})^2 - E_2 - G_2, \\ &\vdots \\ -E_{n-2} - G_{n-2} &\leqslant B_{n-2}\tilde{\rho}_{n-2} \leqslant (\Delta s^2 \kappa_{\max})^2 - E_{n-2} - G_{n-2}. \end{aligned} \tag{34}$$

To sum up, the optimization objectives presented in Equations (9), (14), (15) and (18) are quadratic functions ($\rho^T H\rho + q\rho$). Boundary constraints in Equation (20) are formulated as affine equalities ($A_{\mathrm{eq}}\rho = b_{\mathrm{eq}}$). Both corridor internal constraints in Equation (21) and corridor boundary constraints in Equation (34) have the form of affine inequality ($A_{\mathrm{ie}}\rho \leqslant b_{\mathrm{ie}}$). Therefore, the overall path optimization problem can be described as a general QP problem:

$$\begin{aligned} \text{minimize} \quad & \rho^T H\rho + q\rho, \\ \text{subject to} \quad & A_{\mathrm{eq}}\rho = b_{\mathrm{eq}}, \\ & A_{\mathrm{ie}}\rho \leqslant b_{\mathrm{ie}}. \end{aligned} \tag{35}$$

The implementation process of the path optimization is presented in Algorithm 2. Here, $\rho = [\rho_1, \rho_2, ..., \rho_n]^T$ is defined as the set of scalar factors. The constraints in the path optimization stage are separated into two classes, including boundary constraints and curvature constraints. In particular, the optimization problem without curvature constraints can be solved efficiently. The smoothing cost function can smooth the path and push the path curvature into a reasonable level. Due to most cases that the curvature of initial reference path is reasonable, the generated path after path optimization without curvature constraints is already feasible in curvature. Therefore, a two-steps path optimization strategy in Algorithm 2 is proposed to speed up the path optimization in this paper. The first step optimization does not consider the curvature constraints (Line 4 in Algorithm 2). To speed up the optimization, we provide a initial guess for optimization (Line 3 in Algorithm 2). Here, the initial guess is considered to stay within the corridor and keep close to the reference path. This step may obtain a path which violates the maximal curvature. To check the feasibility of the path, we compute the maximal curvature of the generated path (Line 6 in Algorithm 2). If the generated path passes the check of validity of the maximum curvature (Line 7-9 in Algorithm 2), a feasible solution is returned. If the generated path violates the maximum curvature constraint, the procedure will enter the second step optimization, which is the so-called path optimization with curvature constraints. In this step, the solution of the first step is retained to initialize the second step. Significantly, the parameter Δs is computed approximately before the optimization, which cannot guarantee that the optimized result is strictly feasible. For this, the designed second step optimization will iteratively check and optimize the path until the optimized path passes the check of validity. We first use the last optimization results to update the parameter Δs (Line 12 in Algorithm 2). Then, the path optimization with maximal curvature constraints is activated once to find a solution (Line 13 in Algorithm 2). Finally, the procedure will compute the maximal curvature and further check the validity of the maximum curvature of the solution (Line 15–18 in Algorithm 2). The iteration optimization procedure will terminate until the optimized result passes the feasible check (Line 15–18

in Algorithm 2) or the number N reaches the upper iteration number $N_{\max}$ (Line 11 in Algorithm 2).

Algorithm 2 Spatial path optimization.

Input: $\mathbf{P}^{\text{ref}}$, point set of reference path; $\mathbf{P}^{\text{L}}$ and $\mathbf{P}^{\text{R}}$, boundary points of spatial corridor; $\kappa_{\max}$, the maximal curvature; $N_{\max}$, the upper number of iterations.

Output: $\mathbf{P}^{\text{opt}}$, waypoints of the optimized path

1: $N \leftarrow 0$
2: $Flag \leftarrow False$
3: $\rho^{\text{init}} \leftarrow$ ComputeInitialGuess($\mathbf{P}^{\text{ref}}$, $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$)
4: $\rho^{\text{opt}} \leftarrow$ OptimizationWithoutCurvatureConstraints(ρ^{init}, $\mathbf{P}^{\text{ref}}$, $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$)
5: $\mathbf{P}^{\text{opt}} \leftarrow$ GetOptimizedPoints(ρ^{opt}, $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$)
6: $\kappa \leftarrow$ GetPathMaxCurvature($\mathbf{P}^{\text{opt}}$)
7: **if** $\kappa < \kappa_{\max}$ **then**
8: $\quad Flag \leftarrow True$
9: **end if**
10: **if** *Flag is False* **then**
11: $\quad$ **while** $N < N_{\max}$ **do**
12: $\quad\quad \Delta s \leftarrow$ UpdateSegmentArcLength($\mathbf{P}^{\text{opt}}$)
13: $\quad\quad \rho^{\text{opt}} \leftarrow$ OptimizationWithCurvatureConstraints(ρ^{opt}, $\mathbf{P}^{\text{ref}}$, $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$, Δs)
14: $\quad\quad \mathbf{P}^{\text{opt}} \leftarrow$ GetOptimizedPoints(ρ^{opt}, $\mathbf{P}^{\text{L}}$, $\mathbf{P}^{\text{R}}$)
15: $\quad\quad \kappa \leftarrow$ GetPathMaxCurvature($\mathbf{P}^{\text{opt}}$)
16: $\quad\quad$ **if** $\kappa < \kappa_{\max}$ **then**
17: $\quad\quad\quad$ **break**
18: $\quad\quad$ **end if**
19: $\quad\quad N \leftarrow N + 1$
20: $\quad$ **end while**
21: **end if**
22: **return** $\mathbf{P}^{\text{opt}}$

5. Experiment Results

5.1. Implementation Details

Since the optimization problem can be formulated as a QP one, the path optimization method proposed in this article is implemented in C++11 and with QP solver OSQP [31]. All modules of the complete algorithm are tested on a computer with an Intel i5-8265U processor at 1.60 GHz and 8 GB RAM.

To verify the performance of the proposed algorithm, we first conduct experiments in a vector road map, which is built by an open source mapping tool Lanelets [35]. The vector roadmap is shown in Figure 4. In our test, a global planning module is used to find the global planning path according to the given starting position and the terminal position. Due to manually labeling the map elements, the quality of paths from the global planning module is not guaranteed and these paths are needed to be further optimized. Our proposed algorithm is integrated into the planning module for global reference path optimization. In order to demonstrate that our approach is capable of handling path optimization for different scenarios, we have applied our approach to achieve standard and racing path optimization. Secondly, we also apply our approach to the on-road environment with obstacles for the safe path generation. Finally, in order to demonstrate complexity and feasibility of the designed method, experiments in two different environments are also conducted. A visualization of the path optimization process is shown in Figure 5.

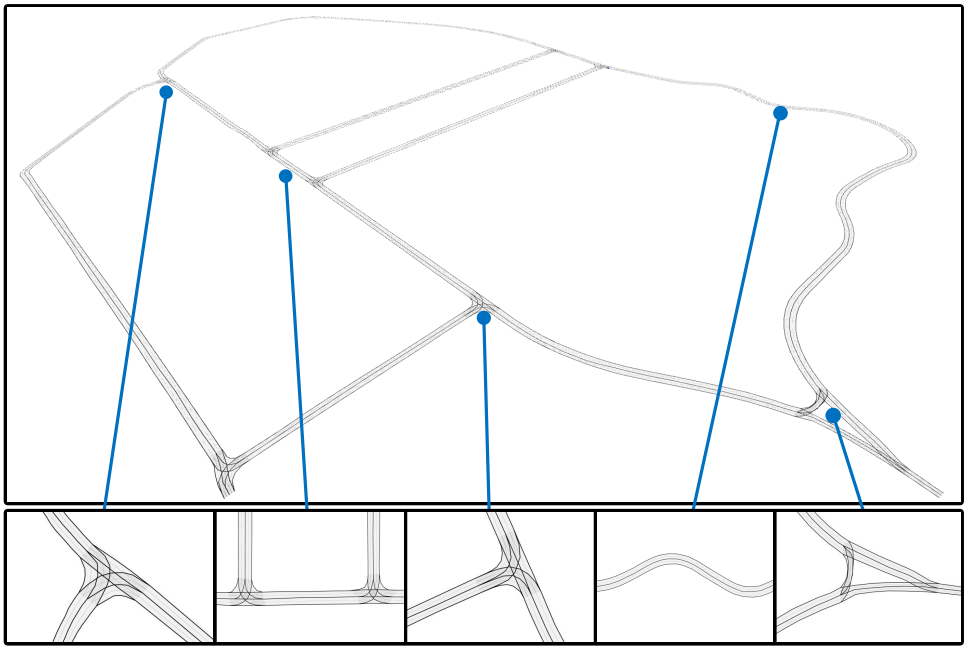


Figure 4. The road map built in campus environment.

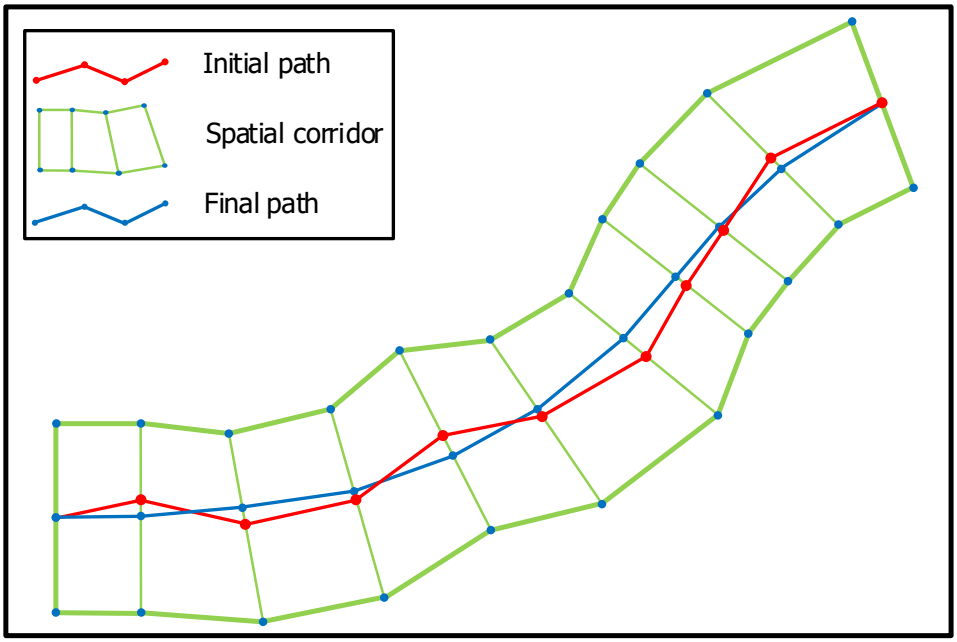


Figure 5. An illustration of the results of path optimization. Spatial corridor and high quality path are generated after optimization. The spatial corridor is directly constructed by the initial path. After optimization, a path is obtained within the region of the spatial corridor. The quality of resulting path (the blue line) is higher than the one before (the red line).

5.2. Standard Path Optimization

In the autonomous driving scene, a high quality reference path is desirable to be provided for self-driving, which primarily fulfills requirements of safety, smoothness and following the initial path as much as possible. The requirement of the shortest path is considered as the secondary goal. Different road conditions including the straight line and turning curve are tested, as shown in Figure 6. Reference paths in Figure 6a,c are given by the global planner module. The points along the reference path are distributed unevenly. These paths are hard to be executed directly, due to having impracticable curvature. After

optimization, resulting paths in Figure 6b,d are high quality paths adjusted in several aspects, including smoothness, curvature, deviation and length of the path.

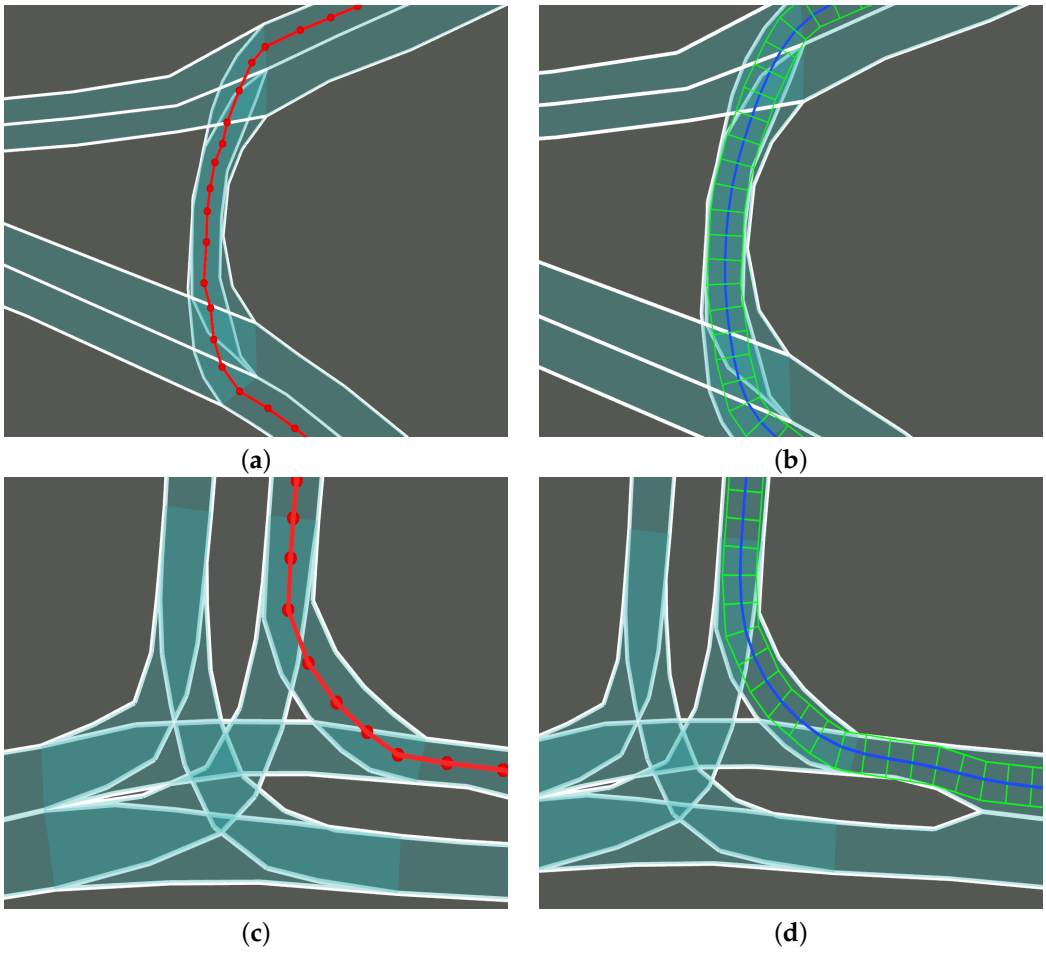


Figure 6. Results of standard path optimization in the static turning environment. (**a**) Initial path; (**b**) Optimized path; (**c**) Initial path; (**d**) Optimized path.

As shown in Table 1, the cost of smoothness of optimized paths is reduced, reaching a lower level. That is, the quality of smoothness of the path is improved. The maximum curvature is calculated to show that the curvature of the generated path is optimized and subject to the maximal curvature constraint.

Table 1. Comparison of evaluation items of paths.

$w_l = 0.500,\ w_s = 0.250,\ w_j = 0.800,\ w_d = 0.300$				
Path	**Length Cost**	**Smoothness Cost**	**Deviation Cost**	**Maximum Curvature Cost**
Figure 7a	105.170	43.371	0.000	0.381
Figure 7b	104.790	1.968	2.266	0.114
Figure 7c	177.627	48.650	0.000	0.427
Figure 7d	176.764	3.093	3.182	0.130

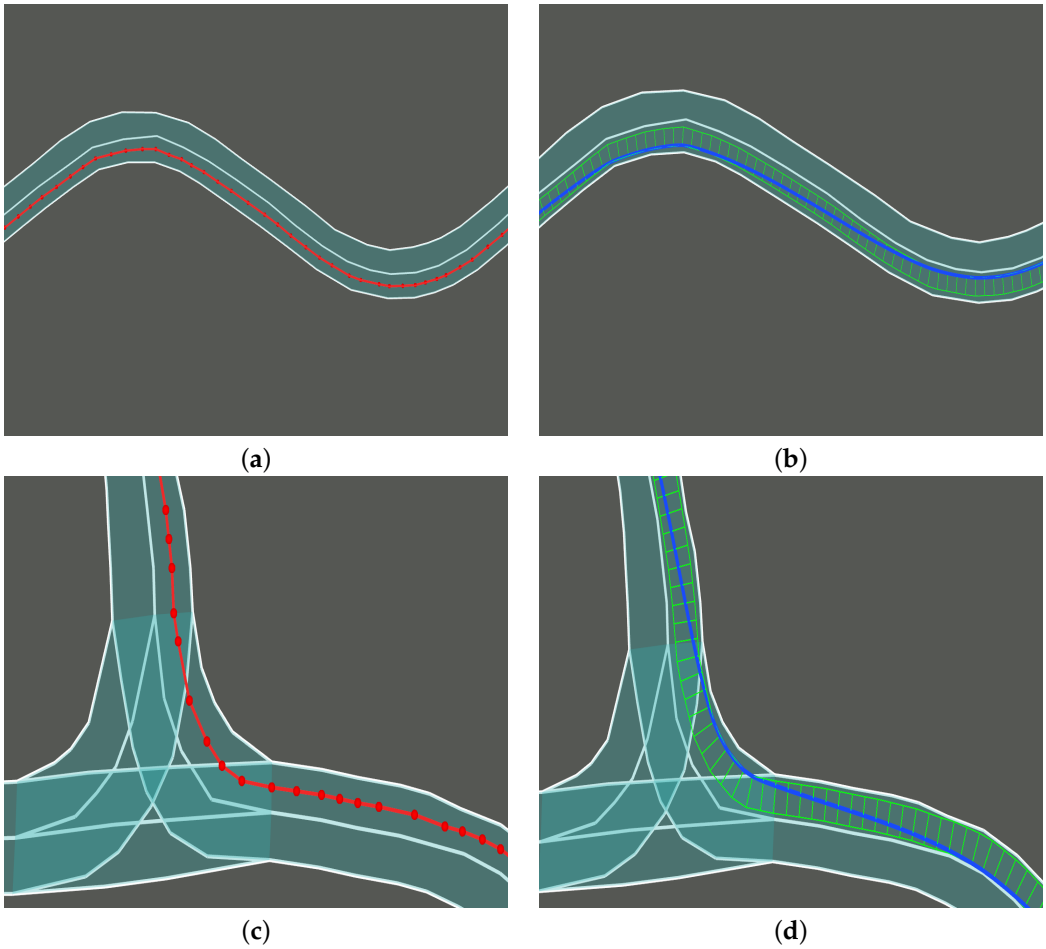


Figure 7. Results of racing line optimization. The red lines represent the initial paths. The blue lines represent the optimized paths. (**a**) Initial path; (**b**) Optimized path; (**c**) Initial path; (**d**) Optimized path.

5.3. Racing Path Optimization

In racing games, a high quality racing path takes the smoothness and length of path into consideration, so that lap-times of driving can be reduced as much as possible. In this test, we apply our approach to find a path mainly minimizing both length and smoothness. Weights of w_l, w_a, w_j and w_d are adjusted to suit for this case, without modification of the algorithm frame. The optimization results in Figure 7b,d show that the generated paths deviate from the reference path and have a better performance in smoothness and length than the initial paths in Figure 7a,c. The associated evaluation results are shown in Table 2.

Table 2. Comparisons of evaluation items of paths.

$w_l = 2.050,\ w_s = 1.200,\ w_j = 0.001,\ w_d = 1.880$				
Path	**Length Cost**	**Smoothness Cost**	**Deviation Cost**	**Maximum Curvature Cost**
Figure 8a	125.215	41.550	0.000	0.107
Figure 8b	122.441	4.304	111.262	0.039
Figure 8c	183.819	32.078	0.000	0.063
Figure 8d	180.907	3.400	137.722	0.028

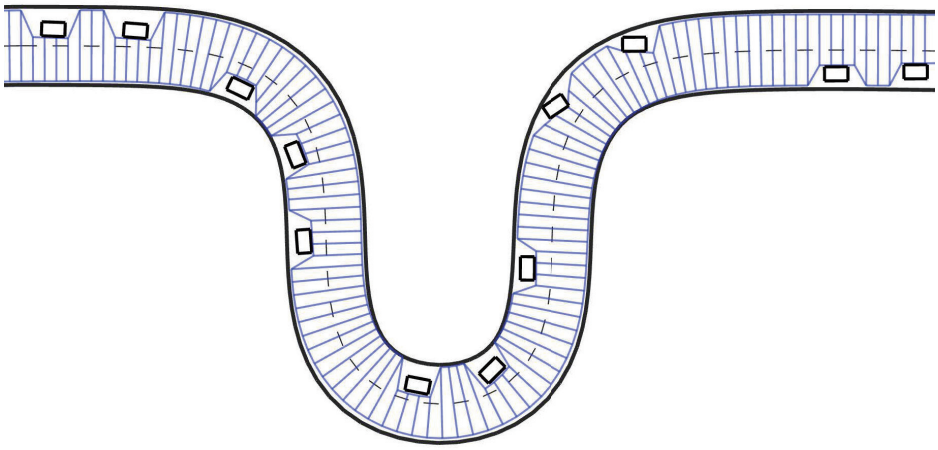


Figure 8. Corridor generation in the on-road environment. The black reatangles represent the static obstacles. The blue grid represents the generating corridor.

5.4. On Road Path Optimization

The above mentioned experiments are conducted in the obstacle-free environments. The corridor generation part is implemented using the consistent parameters, constructing a comparatively regular optimization corridor. The path optimization is achieved on the base of the regular corridor. In this subsection, we show that the proposed approach can be used to optimize the road path in the on-road environment with static obstacles shown in Figure 8. To ensure that the optimized path is passable in the obstacle environment, the corridor generation part first constructs an obstacle-free corridor. As shown in Figure 8, the generated blue corridor along the road is an obstacle-free corridor with irregular shapes. This irregular corridor is still suitable for the path optimization formulation. This corridor is also viewed as a convex domain for path optimization. We take the path connecting intermediate points of the generated corridor as the initial guess for the path optimization, as the red line shown in Figure 9. The initial guess is an non-smooth path with a large curvature. To consider the safe distance from the surrounding obstacles, we compute the parameters $\varepsilon_i = D_{safe}/\|P_i^R - P_i^L\|, i \in \{1, 2, \cdots, n-1\}$ to adjust the corridor internal constraints, where D_{safe} is the safe distance from obstacles. In this test, we set $D_{safe} = 1.5$ m. After path optimization, the returned result is a both smooth and kinematically feasible path, as the green line shown in Figure 9. Figure 10 shows the curvature distributions of the output path along the travel distance.

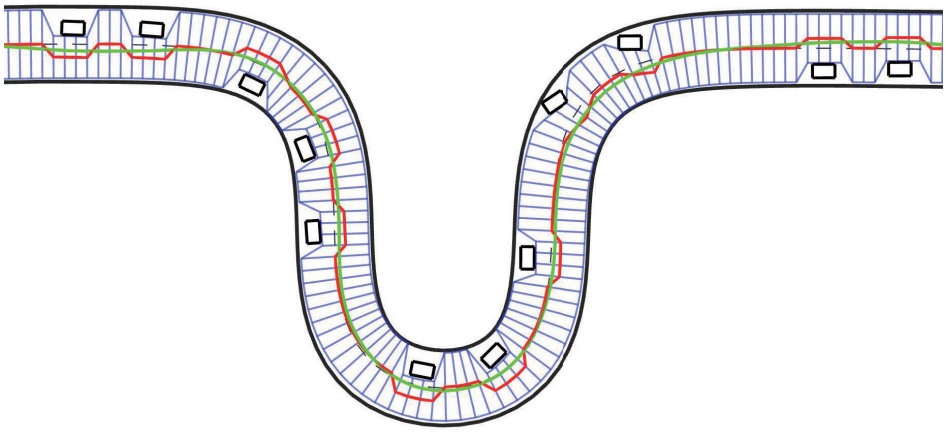


Figure 9. Path optimization in the on-road environment. The red line represents the initial path, while the green line represents the optimized path.

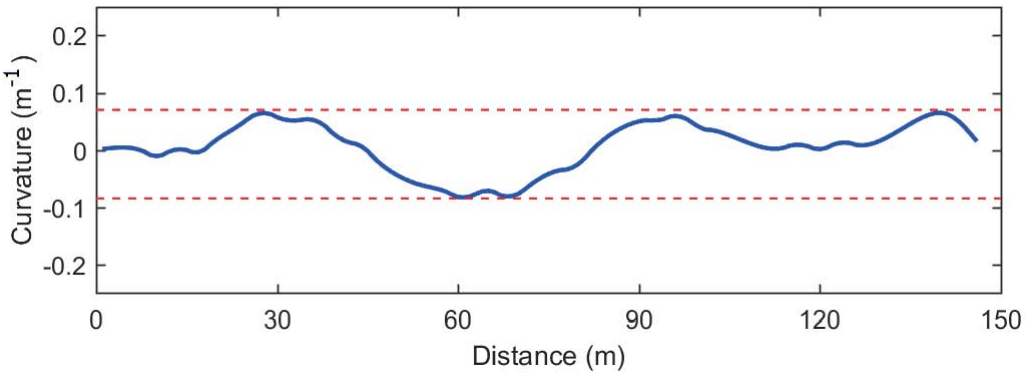


Figure 10. Curvature distributions of the optimized path.

5.5. Complexity and Feasibility

Figure 11 shows the average time consumption of each part of the proposed algorithm. Paths with different lengths are adopted to test the efficiency of our proposed method. A path with more than 1900 waypoints can be optimized completely in less than 40 ms, which shows that the proposed algorithm is able to run in real-time.

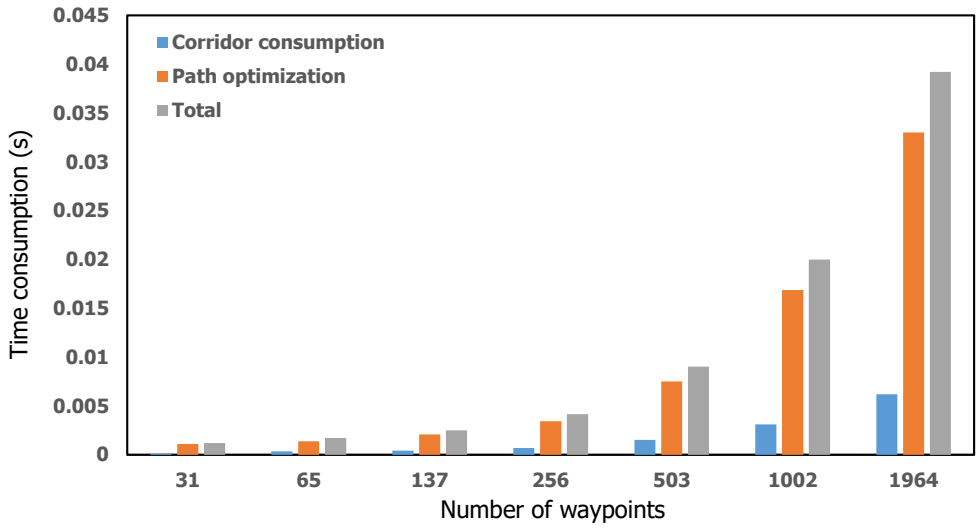
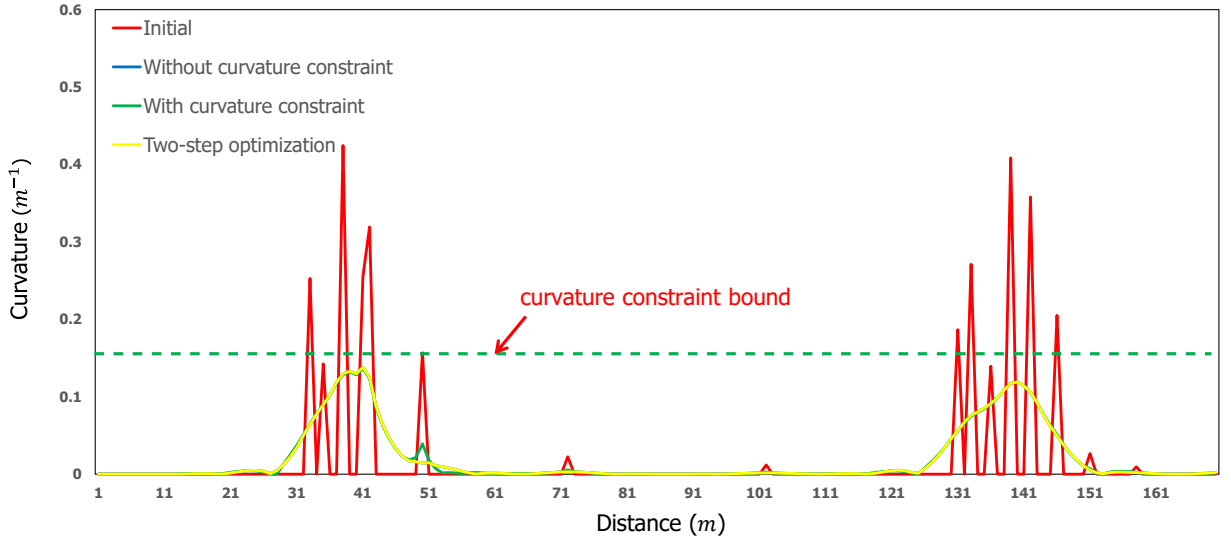
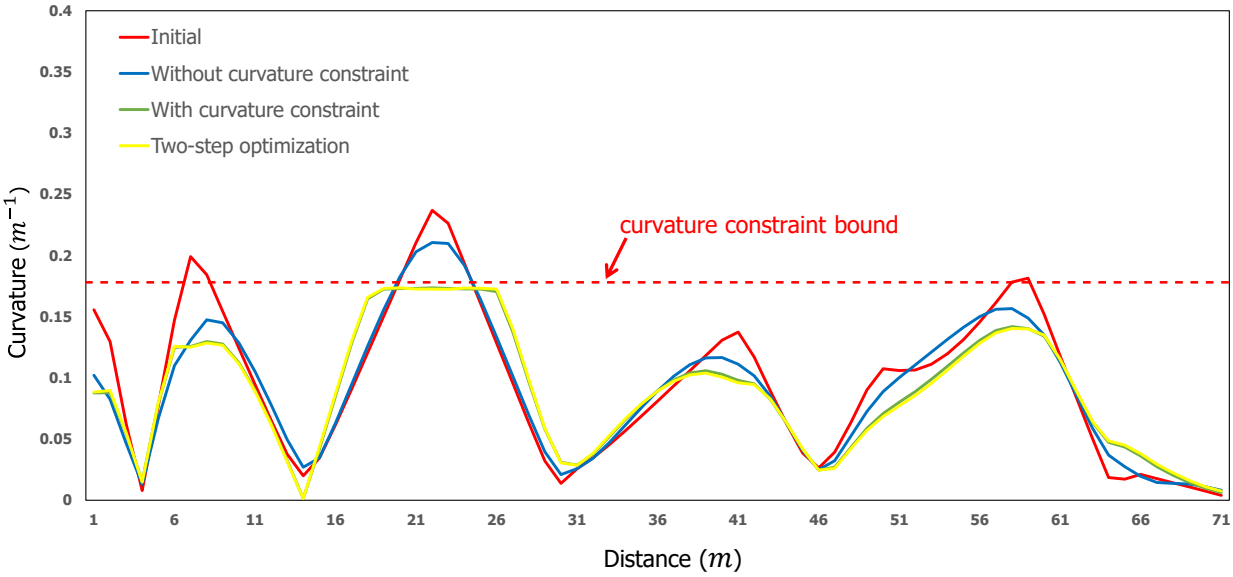


Figure 11. Time consumption of each part of the proposed method, including corridor generation, path optimization and complete process.

In order to prove capabilities of the proposed algorithm to generate a both smooth and kinematically feasible path, we first make a comparison among following three cases: (i) path optimization only with boundary constraint. (ii) Path optimization with boundary constraint and curvature constraint. (iii) Two-steps path optimization with boundary constraint and alternative curvature constraint (showed in Algorithm 2). In Figure 12, the curvature distribution of the initial path surpasses the permissible maximum curvature, as shown in the red curve. In Figure 12b, the optimization result of case (i) violates the maximum curvature constraint, while results of case (ii) and case (iii) satisfy the maximum curvature constraint. In Figure 12a, all the cases can generate a feasible path satisfying the miximal curvature constraint.



(**a**)



(**b**)

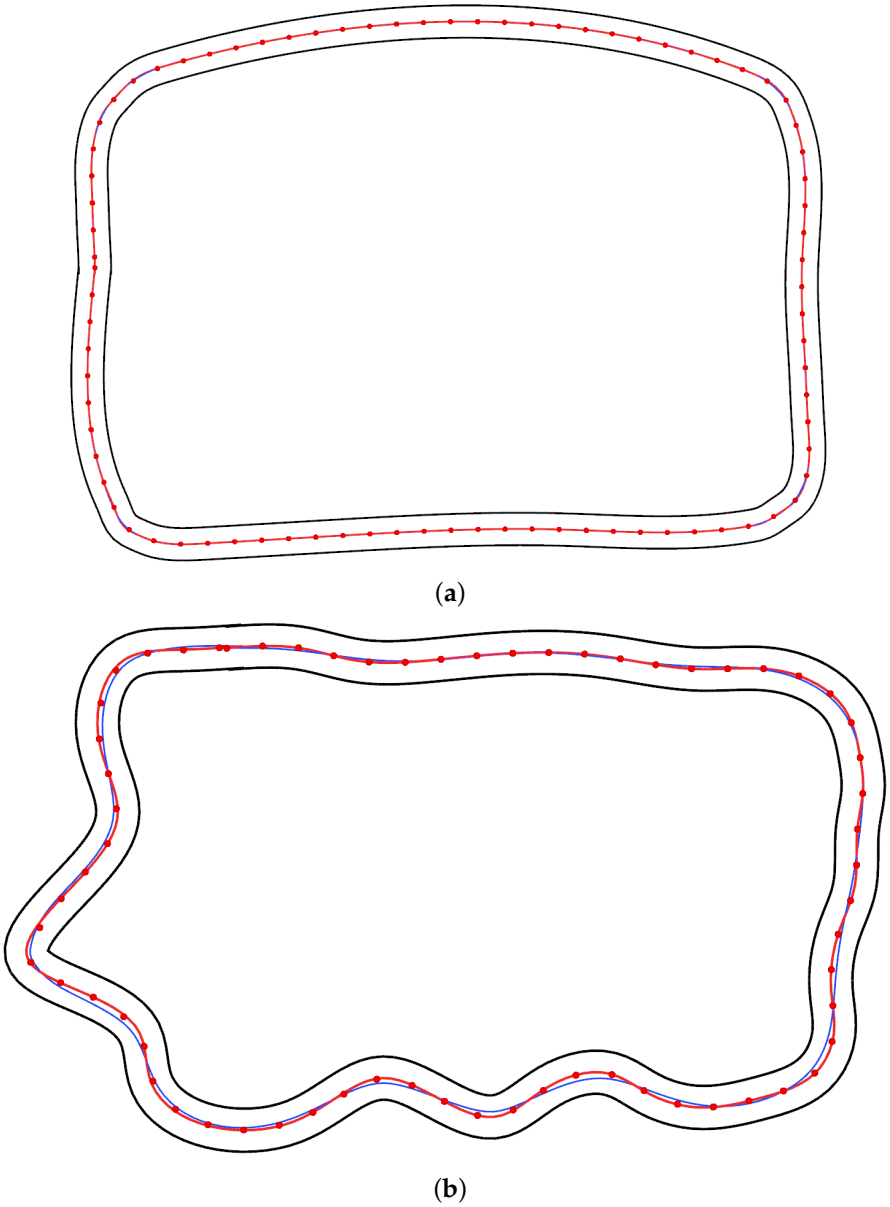
Figure 12. Comparison of the curvature distribution of reference path and paths optimized by different methods. The red curve represents the curvature distributions of the initial path. Blue, green and yellow curves are generated by the method only with boundary constraint, method with boundary constraint and curvature constraint and two-step optimization method, respectively. (**a**) Optimization results in simple environment; (**b**) Optimization results in complex environment.

Table 3 indicates that the case (iii) has the same level of computation time as the case (ii) in the complex road environment shown in Figure 13b. In the simple road environment shown in Figure 13a, the computation time of the case (iii) is close to the case (i), possessing a high computing speed.

According to the above description of Figure 12 and Table 3, it can be concluded that the case (i) is capable of finding a path in the corridor with a lower computation complexity, but fails to ensure the curvature feasibility of the path. The case (ii) is able to generate a feasible path but runs in a higher computational complexity than others. The case (iii), which is called the two-steps path optimization in Algorithm 2, achieves a good trade-off between the curvature feasibility and computation complexity.

Table 3. Performance of path optimization.

Env.	Time Consumption (s)		
	Without Curvature Constraints	**With Curvature Constraints**	**Two-Steps Optimization**
Figure 9a	0.00611	0.00997	0.00653
Figure 9b	0.00537	0.01606	0.01678



(**a**)

(**b**)

Figure 13. Different environments are used to test the designed method. (**a**) is a simple environment with normal lanes, while (**b**) is a unconventional environment with complex lanes. (**a**) Path optimization in simple environment; (**b**) Path optimization in complex environment.

6. Conclusions and Future Work

This paper presented a corridor-based method for spatial path optimization, which is mainly containing two modules: spatial corridor generation and two-steps path optimization. The spatial corridor construction module is designed to extract an optimization domain for path optimization. The two-steps path optimization approach is designed to generate a smooth path in real-time while ensuring the feasibility of path curvature. The

overall path optimization is formulated as a general QP problem and solved efficiently. To evaluate our method, we used our proposed method to achieve path optimization in different environments. Simulation experiments demonstrate the validity of the proposed method. Further work will be presented to investigate the path speed optimization and integrate it into our system. Moreover, it is promising to extend its applications to other scenarios.

Author Contributions: Conceptualization, Y.L. (Yongkang Lu) and Y.W.; methodology,Y.L. (Yongkang Lu); software, Y.L. (Yongkang Lu); validation, Y.L. (Yongkang Lu), Y.W. and W.Z.; formal analysis, Y.L. (Yanzhou Li); investigation, M.C.; resources, W.Z.; data curation, M.C.; writing—original draft preparation, W.Z.; writing—review and editing, Y.L. (Yongkang Lu); visualization, W.Z.; supervision, Y.L. (Yanzhou Li); project administration, Y.W.; funding acquisition, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 62003100, 62276074.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Rasekhipour, Y.; Fadakar, I.; Khajepour, A. Autonomous driving motion planning with obstacles prioritization using lexicographic optimization. *Control. Eng. Pract.* **2018**, *77*, 235–246. [CrossRef]
2. Claussmann, L.; Revilloud, M.; Gruyer, D.; Glaser, S. A review of motion planning for highway autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1826–1848. [CrossRef]
3. Katrakazas, C.; Quddus, M.; Chen, W.H.; Deka, L. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transp. Res. Part Emerg. Technol.* **2015**, *60*, 416–442. [CrossRef]
4. 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]
5. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
6. LaValle, S.M.; Kuffner, J.J. Rapidly-Exploring Random Trees: Progress and Prospects. *Algorithmic Comput. Robot.* **2001** , *5*, 303–307.
7. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
8. Liu, M. Robotic online path planning on point cloud. *IEEE Trans. Cybern.* **2015**, *46*, 1217–1228. [CrossRef] [PubMed]
9. Nelson, W. Continuous-curvature paths for autonomous vehicles. In Proceedings of the International Conference on Robotics and Automation, Scottsdale, AZ, USA, 14–19 May 1989; pp. 1260–1264.
10. Berglund, T.; Jonsson, H.; Soderkvist, I. An obstacle-avoiding minimum variation b-spline problem. In Proceedings of the International Conference on Geometric Modeling and Graphics, London, UK, 16–18 July 2003; pp. 156–161.
11. Hwang, J.H.; Arkin, R.C.; Kwon, D.S. Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453), Las Vegas, NV, USA, 27–31 October 2003; Volume 2, pp. 1444–1449.
12. Connors, J.; Elkaim, G. Analysis of a spline based, obstacle avoiding path planning algorithm. In Proceedings of the Vehicular Technology Conference-VTC2007-Spring, Dublin, Ireland, 22–25 April 2007; pp. 2565–2569.
13. Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; How, J.P. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control. Syst. Technol.* **2009**, *17*, 1105–1118. [CrossRef]
14. Ma, L.; Xue, J.; Kawabata, K.; Zhu, J.; Ma, C.; Zheng, N. Efficient sampling-based motion planning for on-road autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 1961–1976. [CrossRef]
15. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Practical search techniques in path planning for autonomous driving. *Ann. Arbor.* **2008**, *1001*, 18–80.
16. Dolgov, D.; Thrun, S. Autonomous driving in semi-structured environments: Mapping and planning. In Proceedings of the IEEE international Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3407–3414.
17. Webb, D.J.; Van Den Berg, J. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 5054–5061.
18. Cardamone, L.; Loiacono, D.; Lanzi, P.L.; Bardelli, A.P. Searching for the optimal racing line using genetic algorithms. In Proceedings of the IEEE Conference on Computational Intelligence and Games, Copenhagen, Denmark, 18–21 August 2010; pp. 388–394.

19. Ding, W.; Zhang, L.; Chen, J.; Shen, S. Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2997–3004. [CrossRef]
20. Zhu, Z.; Schmerling, E.; Pavone, M. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In Proceedings of the IEEE conference on decision and control (CDC), Osaka, Japan, 15–18 December 2015; pp. 835–842.
21. Schouwenaars, T.; De Moor, B.; Feron, E.; How, J. Mixed integer programming for multi-vehicle path planning. In Proceedings of the European control conference (ECC), Porto, Portugal, 4–7 September 2001; pp. 2603–2608.
22. Ziegler, J.; Bender, P.; Dang, T.; Stiller, C. Trajectory planning for Bertha—A local, continuous method. In Proceedings of the IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, 8–11 June 2014; pp. 450–457.
23. Wu, X.; Li, Z.; Kan, Z.; Gao, H. Reference trajectory reshaping optimization and control of robotic exoskeletons for human–robot co-manipulation. *IEEE Trans. Cybern.* **2019**, *50*, 3740–3751. [CrossRef] [PubMed]
24. Chai, R.; Savvaris, A.; Tsourdos, A.; Xia, Y.; Chai, S. Solving multiobjective constrained trajectory optimization problem by an extended evolutionary algorithm. *IEEE Trans. Cybern.* **2018**, *50*, 1630–1643. [CrossRef] [PubMed]
25. Caporale, D.; Fagiolini, A.; Pallottino, L.; Settimi, A.; Biondo, A.; Amerotti, F.; Massa, F.; Caro, S.D.; Corti, A.; Venturini, L. A planning and control system for self-driving racing vehicles. In Proceedings of the IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI), Palermo, Italy, 10–13 September 2018; pp. 1–6.
26. Braghin, F.; Cheli, F.; Melzi, S.; Sabbioni, E. Race driver model. *Comput. Struct.* **2008**, *86*, 1503–1516. [CrossRef]
27. Ziegler, J.; Bender, P.; Schreiber, M.; Lategahn, H.; Strauss, T.; Stiller, C.; Dang, T.; Franke, U.; Appenrodt, N.; Keller, C.G.; et al. Making bertha drive—An autonomous journey on a historic route. *IEEE Intell. Transp. Syst. Mag.* **2014**, *6*, 8–20. [CrossRef]
28. Gu, T.; Snider, J.; Dolan, J.M.; Lee, J. Focused trajectory planning for autonomous on-road driving. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Gold Coast, Australia, 23–26 June 2013; pp. 547–552.
29. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
30. Domahidi, A.; Chu, E.; Boyd, S. ECOS: An SOCP solver for embedded systems. In Proceedings of the European Control Conference (ECC), Zurich, Switzerland, 17–19 July 2013; pp. 3071–3076.
31. Stellato, B.; Banjac, G.; Goulart, P.; Bemporad, A.; Boyd, S. OSQP: An operator splitting solver for quadratic programs. *Math. Program. Comput.* **2020**, *12*, 637–672. [CrossRef]
32. Zhou, J.; He, R.; Wang, Y.; Jiang, S.; Zhu, Z.; Hu, J.; Miao, J.; Luo, Q. Dl-iaps and pjso: A path/speed decoupled trajectory optimization and its application in autonomous driving. *arXiv* **2020**, arXiv:2009.11135.
33. Quinlan, S.; Khatib, O. Elastic bands: Connecting path planning and control. In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, USA, 2–6 May 1993; pp. 802–807.
34. Gao, F.; Wang, L.; Zhou, B.; Zhou, X.; Pan, J.; Shen, S. Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments. *IEEE Trans. Robot.* **2020**, *36*, 1526–1545. [CrossRef]
35. Bender, P.; Ziegler, J.; Stiller, C. Lanelets: Efficient map representation for autonomous driving. In Proceedings of the IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, 8–11 June 2014; pp. 420–425.