Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization

YU MENG, YANGMING WU, QING GU, AND LI LIU
School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, 100083 China

Corresponding author: Yu Meng (e-mail: myu@ustb.edu.cn).

This work was supported by the National Key Research and Development Program of China under Grant No.2018YFC0604403, No.2018YFE9102900, the National High Technology Research and Development Program of China (863 program) under Grant No.2011AA060408 and the Fundamental Research Funds for the Central Universities under Grant No.FRF-TP-17-010A2.

ABSTRACT This paper presents a decoupled trajectory planning framework based on the integration of lattice searching and convex optimization for autonomous driving in structured environments. For a 3D trajectory planning problem with timestamps information, due to the presence of multiple kinds of constraints, the feasible domain is non-convex, so it is easy to fall into local optimum for trajectory planning. And the solution space of this problem is so enormous that it is difficult to identify an optimal solution in polynomial time. To address this non-convex problem, and to improve the convergence speed of an optimization process, the approach based on lattice searching is adopted in consideration of the ability to discretize driving environments and reduce the solution space. And the resulting path generated by lattice searching typically lies in the neighborhood of the global optimum. But this solution is neither spatiotemporally smooth nor globally optimal, so it is generally called the rough solution. For this reason, a subsequent nonlinear optimization process is introduced to refine the rough trajectory (combined by path and speed). The proposed framework is implemented and evaluated through simulations in various challenging scenarios in this paper. The simulation results verify that the trajectory planner can generate high-quality trajectories, and the execution time is also acceptable.

INDEX TERMS Trajectory planning, Path planning, Speed profile planning, Lattice searching, Dijkstra's algorithm, Nonlinear optimization.

I. INTRODUCTION

RESEARCH on autonomous vehicles has made considerable progress during the past few decades. As a core module of the autonomous driving system, the planning module takes into account all information sensed by sensors, including driving environments information and vehicles state, and then generate a safe, spatiotemporally smooth, and feasible trajectory to feed into the control module. Although many excellent trajectory planning algorithms are proposed, it is still difficult to generate a good trajectory in real time in dynamic uncertain environments. And these methods are not generic in all scenarios.

A trajectory planning problem is a typical 3D constrained planning problem by considering timestamps information and spatial information. In structured environments, the 3D feasible set is non-convex due to the constraints of traffic rules, time consumption, nonholonomic and collision avoidance. Hence, the following difficulties need to be dealt with simultaneously: (1) how to reduce the computational complexity of the planning algorithm to execute fast enough to meet real-time requirement for planning and replanning, which means we need to quickly generate a feasible solution in an enormous solution space, (2) how to address spatiotemporal obstacles in 3D state space, which means the planning problem is a highly non-convex problem, so it is extremely difficult to ensure that the solution converges to a single, global optimum, and (3) how to generate a spatiotemporally smooth trajectory, which means the curvature profile, speed

profile, acceleration profile, and jerk profile of a trajectory are all continuous and smooth.

A. RELATED WORK

According to whether the configuration space is continuous or not, trajectory planning techniques for autonomous driving can be divided into two categories: sampling-based methods and optimization-based methods. According to whether the state space is decoupled or not, there are also two different categories in trajectory planning techniques: direct planning methods and decoupled planning methods. This paper will review typical trajectory planning techniques in terms of whether the state space is decoupled or not.

Direct methods attempt to find one optimal trajectory in the Cartesian coordinate system or the frenet coordinate system [1]. These planning methods perform searching or optimization in spatiotemporal state space, which means they can directly deal with dynamic obstacles. Ziegler et al. [2] carefully design an objective function and constraints to transform a nonconvex problem into a convex one. And they introduce sequential quadratic programming (SQP) algorithm to solve the nonlinear optimization problem. SQP often seems as state-of-the-art technique to solve the nonlinear programming problem [3], [4]. But the convergence speed in SQP without a good initial value is not fast enough for the real-time requirement for a medium scale optimization problem.

A fast planning algorithm called the convex feasible set (CFS) algorithm is proposed to solve optimization-based motion planning problems with convex objective functions and nonconvex constraints [5], [6]. The main idea of the CFS algorithm is to transform the original problem into a sequence of convex subproblems and iteratively solve subproblems until convergence, which makes the computation faster than SQP and interior point method (IP). In [7], [8], the constrained iterative LQR (CILQR) is proposed to efficiently solve the optimal control problem with nonlinear system dynamics and general form of constraints. And the computation efficiency of CILQR is shown to be much higher than the standard SQP solver. But, it is easy to fall into a stationary point for these technologies without being given a good initial solution. A trajectory is a one-to-one mapping between the time domain and the space domain, so these optimization-based techniques above usually sample at equal time intervals. And positions of a vehicle in the space domain are free variables. According to constraint functions and performance index, an appropriate algorithm is picked to optimize the decision variables. However, in a clustered environment, the feasible region is usually highly non-convex both in spatial and temporal. And the convexification of the feasible region is not easy for them. In addition, the generalization ability of these algorithms is also a problem. For example, when the decision-making layer issues an instruction, it is not easy to adjust the speed profile or path flexibly for these algorithms.

Howard et al. [9], [10] develop an efficient and general model predictive trajectory generation technology via the shooting method and Newton's method. Usually, a set of terminal states are sampled in the state space, then Howard's technique is used to connect the initial state with the sampling states while respecting nonholonomic constraints. At last, the carefully designed cost function is used to select the best trajectory with the lowest cost. Similarly, another widely used method is the lattice searching approach. In [11], [12], a conformal spatiotemporal lattice with time and speed dimensions is proposed to generate a feasible trajectory in a dynamic scenario. And dynamic programming is adopted to search an optimal trajectory in a lattice. The highlight of lattice searching is that the sampling resolution of a lattice must be balanced between the time consumption and completeness. The execution time increases as the resolution increases. And if the resolution is reduced, a feasible solution can't even be found.

Different from direct methods, decoupled methods reduce the dimensions of the state space and perform planning in two 2D state space separately. In Werling et al's method [1], they generate lateral and longitudinal trajectories using quantic polynomials versus time, which ensures continuous speed profile and acceleration profile. However, this method may lead to frequent swerving of a vehicle and cannot guarantee the optimality of the generated solution.

Impressively, another popular decoupled method is the path-speed decoupling method which plans path and speed profile, respectively. Path planning typically takes static obstacles into consideration, and then a speed profile is generated based on the generated path. Two representative methods are A* search [13]and random sampling-based method that mainly refers to the rapidly-exploring random tree (RRT) [14], [15]. These two class of methods usually search in a grid map or sample in the configuration space to generate a continuous path. For graph search-based methods, the resulting paths are not smooth, which means the subsequent smoothing process should be employed to make the path meet kinodynamic constraints. For random sampling-based methods, the resulting paths tend to be jerky and redundant, which means the subsequent smoothing processing is also introduced.

Still path-speed decoupling, in [16], [17], polynomial spirals are used to solve the two-point boundary value problem in order to generate a path which satisfies nonholonomic constraints, and then the trapezoidal velocity profile is employed for the longitudinal velocity profile generation. This reactive trajectory planning method can meet the real-time requirement, but it cannot guarantee the trajectory is globally optimal. At the same time, the speed profile is not flexible enough to adapt to complicated scenarios. This method sampling in state space only specifies a finite set of motion primitives, which reduces the motion potential of vehicles. To overcome these shortcomings, xu et al. [18] propose a two-step planning framework. They first sample separately in posture space (x, y, θ, κ) [24] and speed space, then search for a rough path and a rough speed profile, finally enter the optimization step to optimize the trajectory. However, motion primitives in their work are spirals which are not

TABLE 1. Comparison of trajectory planning methods

Methods	Implemented in	Optimality	Mobility	Completeness	Spatial smoothness	Temporal smoothness	Anytime	Driving environment	Flexibility	Curvature constraint	State space
direct methods	Ziegler et al.[2]	Locally optimal	High	✓	✓	✓	✓	FE and SE	×	✓	Continuous
	CFS[5][6]	Locally optimal	High	✓	✓	✓	✓	FE and SE	×	✓	Continuous
	CILQR[7][8]	Locally optimal	High	✓	✓	✓	✓	FE and SE	×	✓	Continuous
	Howard et al. [9] [10]	Suboptimal	Low	resolution complete	✓	×	✓	FE and SE	✓	✓	Discrete
	Spatiotemporal lattice[11][12]	Suboptimal	Low	resolution complete	✓	×	×	FE and SE	✓	✓	Discrete
decoupled methods	Werling et al. [1]	Suboptimal	Low	resolution complete	✓	✓	✓	SE	✓	✓	Discrete
	Li et al. [16] [17]	Suboptimal	Low	resolution complete	✓	×	✓	SE	✓	✓	Discrete
	Lim et al.[4]	✓	High	✓	✓	✓	✓	SE	✓	×	Discrete and continuous
	Fan et al.[19]	✓	High	✓	✓	✓	✓	SE	✓	×	Discrete and continuous
	Xu et al.[18]	✓	High	✓	✓	✓	✓	SE	✓	×	Discrete and continuous
	Liu et al.[31]	✓	High	✓	✓	✓	✓	SE	×	✓	Continuous
	Zhan et al.[35]	✓	Medium	✓	✓	✓	✓	SE	✓	×	Discrete and continuous

good choices compared to polynomials in terms of computational complexity, and the subsequent optimization do not guarantee the trajectories satisfy nonholonomic constraints and sideslip constraints.

In Lim et al's method [4], they sample some vertices in frenet coordinates and the Space-Time coordinates, respectively, and then use dynamic programming to search a rough path, and use the Hybrid A* algorithm [23] to search the rough speed profile. Finally, SQP is introduced to optimize the rough trajectory generated by the decoupled method in [4]. In [19], a combination of dynamic programming and spline-based quadratic programming is proposed to construct a scalable and easy-to-tune framework to handle traffic rules, obstacle decisions and smoothness simultaneously. This planner searches path and speed profile respectively, and then optimizes path and speed profile, respectively. The above methods can make the solution jump out of a stationary point. Simultaneously, the optimization process in these methods makes the solution close to the globally optimal solution, and also makes the trajectory smoother than the untreated trajectory. However, the imperfection in their work is that they ignore the curvature constraints during the optimization process, which makes their algorithms not trustworthy algorithms for the trajectory tracking module.

For the sake of clarity, we compare some of the considerable features of direct methods and decoupled methods, including optimality, mobility, completeness, spatial smoothness, temporal smoothness, real-time performance, driving environment, flexibility, curvature constraints and state space, which can be seen in Table 1. Optimality represents the ability of an algorithm to find an optimal solution in a nonconvex space. Among these methods, the solution generated by only the optimization-based technologies is easy to fall into a stationary point, so the optimality is locally optimal. As for the sampling-based technologies, their solutions are generally located in the neighborhood of the global optimum, so the optimality is suboptimal. Mobility of an algorithm is determined by whether the motion potential of a vehicle is restricted or not. Some algorithms [9-12] specify the values of the kinematics and dynamics parameters of a vehicle (i.e., special wheel angles, speed and acceleration), which may reduce the motion performance of a vehicle, so the mobility is also one of the algorithms' considerations. In [9-12], [16-17], motion primitives are generated according to certain predefined rules, so the mobility is low. Completeness ([37], [45]) indicates the ability of an algorithm to find a feasible solution in the solution space. Spatial smoothness is determined by whether the heading profile and curvature profile of a trajectory are continuous and smooth or not. Temporal smoothness indicates the characteristics of the speed profile, acceleration profile, and jerk profile of a trajectory. Driving environments of an autonomous vehicle are basically divided into free environments (FE) and structured environments (SE). Different algorithms are developed depending on the specified environment. Flexibility indicates the ability of an algorithm to adapt to different driving scenarios (i.e., following, parking and lane changes). Considering the computational complexity, some algorithms (i.e., [4], [18-19], [35]) ignore the curvature constraints to improve the convergence speed of a trajectory. The state space term indicates whether the planning space is discrete or not. In general, some algorithms [9-12] perform planning in discrete state space, and some algorithms (i.e., [2], [5-8]) perform planning in continuous state space. But decoupled algorithms (i.e., [4], [18-19], [35]) usually prefer to perform planning in both discrete and continuous state space according to our observations.

To sum up, both direct methods and decoupled methods often use sampling-based technologies to discretize the solution space, and use optimization-based technologies to generate a continuous and smooth solution. However, direct methods search in spatiotemporal lattice or transform a spatiotemporal planning problem into an optimization problem

to generate a feasible trajectory, so static and dynamic obstacles can be considered at the same time. But, if a good initial solution is not given, or the objective function and constraints are not addressed subtly, it is easy to fall into local optimum, and the convergence speed of the algorithm may not be satisfactory. Moreover, the resolution of the spatiotemporal lattice cannot be too high due to the planning in 3D space, and the trajectory generated by lattice is not spatiotemporally smooth enough. For decoupled methods, they usually transform a medium scale searching or optimization problem into two small scale searching or optimization problems, which greatly reduces the computational complexity of the trajectory planner. Inevitably, static and dynamic obstacles are considered separately in the path planning step and the speed planning step, which may result in a suboptimal trajectory. Simultaneously, there are too many parameters to tune in decoupled methods, which makes the adjustment of parameters become a problem.

And our conclusion is that there is currently no universal algorithm that can adapt to multiple scenarios while meeting all the constraints and requirements in Table 1 in structured environments. Short-sighted, locally optimal, incomplete, and inflexible imperfections are distributed separately in current algorithms as shown in Table 1. So developing a generic, real-time, flexible, and comprehensive algorithm is an important research topic for trajectory planning.

B. CONTRIBUTIONS

Based on related work, and in order to solve the trajectory planning problem in real time for autonomous driving in structured environments, we propose a novel decoupled trajectory planning framework which decouples a 3D planning problem into two 2D planning problems. Firstly, we perform path searching and path post-optimization. Then according to the optimized path, we perform speed searching and subsequent speed optimization. Prominently, a feasible solution in discrete configuration space can be quickly searched by the lattice searching step and be fed to the nonlinear optimization step. With lattice searching, nonlinear optimization can quickly converge to an optimal and continuous solution with just several iterations. The main contributions of this paper are summarized as follows:

- A novel decoupled trajectory planning framework we developed combines the advantages of optimization-based methods and sampling-based methods. For optimization-based methods ([2], [5-8]), they can guarantee sufficient smoothness, while sampling-based methods ([16-17], [25-26]) focus on real-time and flexiblity. The highlight of the framework achieves a good balance between constraints (i.e., curvature constraints, collision avoidance constraints and traffic rule constraints) compliance and requirements (i.e., optimality, smoothness, real-time performance, and flexibility) satisfaction through the combination of methods. Compared with the methods in [2], [5-8], the decoupled method in this paper can jump out of a stationary point, reduce the number of iterations, and enhance the flexibility. And the decoupled method also improve the smoothness and completeness in comparison with the methods in [16-17] and [25-26]. Hence, this trajectory planner can quickly generate a spatiotemporally smooth and kinematically-feasible trajectory.
- We introduce a local, continuous method to refine the rough path generated by lattice searching. [19] transforms a path optimization problem into a quadratic programming (QP) problem. And the Simplex algorithm is used for path optimization in [18]. In [4], SQP is introduced to optimize the rough trajectory. These methods also smooth the path or trajectory while avoiding collisions. However, there is no curvature constraints added to the path or trajectory optimization, which may make the generated path or trajectory not satisfy the nonholonomic constraints. Hence, in this paper, we consider the path optimization problem as a nonlinear programming (NLP) problem and add the curvature constraints for the path optimization step in the frenet coordinate system. The highlight is that different from the work in [2], [6-8] and [31], the rough path is fed to the NLP problem as a hot start in this paper, which ensures a fast convergence and prevents a local optimism. And another novelty is that optimization can be performed over lateral offset in this NLP problem and curvature can be calculated by differences of the frenet waypoints.
- We formulate the speed optimization problem into a standard QP problem which optimizes the longitudinal station for all waypoints along the predefined time domain in urban environments. Unlike the spline QP speed optimizer in [19] and the non-derivative Simplex algorithm speed optimizer in [18], we optimize the decision variable $\mathbf{s} = [s_1, s_2, ..., s_n]^T$ that can be mapped to the fixed timestamps, which can ensure that the speed profile meets the kinematic constraints, dynamic constraints, and temporal smoothness requirements. The work in [31] directly models the speed optimization problem as a NLP problem and does not provide a good initial value for optimization. Compared with the work in [31], our work provides a rough speed profile as the initial value for QP, so the number of iterations of the algorithm is much less than that in [31]. And because of the existence of the speed lattice searching, our algorithm can guarantee global temporal optimality. In addition, compared with the trapezoidal linear velocity profile in [9-10] and the smooth trapezoidal velocity profile in [16-17], the speed planner we developed does not specify the values of speed and acceleration. Impressively, we optimizes the speed according to the objective function and constraint functions, which makes the speed planner can adapt to different driving scenarios, and flexibility and realtime performance is guaranteed.

This paper is organized as follows. The algorithm framework is introduced in Section II. The implementation details

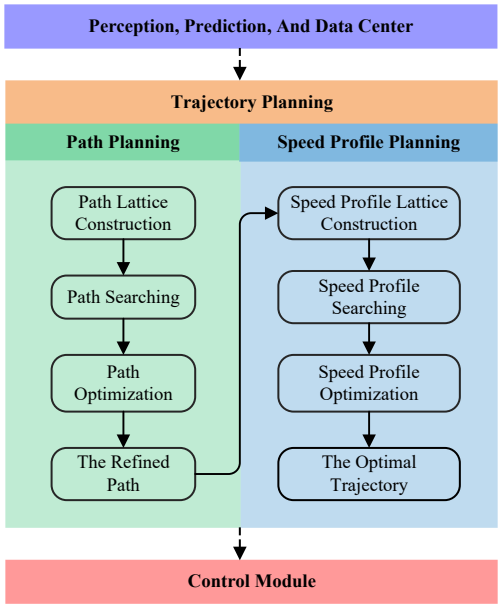


FIGURE 1. The decoupled trajectory planning framework.

of the framework are presented in Section III, which includes the following subsections: rough path searching step, NLP path optimization step, rough speed profile searching step and QP speed optimization step. Section IV explains the settings of simulations, presents four cases of different scenarios and analyzes the performance of the proposed algorithm framework in details. Section V summarizes the contributions and discusses the future work.

II. TRAJECTORY PLANNING FRAMEWORK

The decoupled trajectory planning framework is shown in Fig. 1. The core parts of this framework are path lattice searching, path optimization, speed profile lattice searching and speed profile optimization. In preprocessing, we convert the host vehicle information (i.e., pose, speed, acceleration, and jerk) and the driving environment information (i.e., obstacles information) into frenet coordinates from Cartesian coordinates. And the centerline of a lane approximately parameterized by arc-length is used as a reference line for planning. In the path searching step, we sample in the frenet coordinate system along a lane. And the sampled poses form vertices of a weighted directed acyclic graph (DAG), then the path edges (motion primitives) are generated using the method described in [20]. Kinematic constraints, nonholonomic constraints, obstacle avoidance constraints, smoothness performance, and traffic rules constraints constitute the edge weights of the DAG. Finally Dijkstra's algorithm is picked to find the shortest path to generate a kinematically-feasible, relatively smooth, collision-free path in the non-convex spatial space. In order to optimize the path to meet the spatial smoothness and comfort requirements, we model the path optimization problem into a NLP problem with quadratic objective function and nonlinear constraint functions. The objective function is a linear combination of smoothness and lateral offset from the reference line. And the constraints include boundary value constraints, nonholonomic constraints (curvature constraints) and obstacle avoidance constraints.

During speed profile searching, the trajectory of the ego vehicle and predicted trajectories of obstacles are projected in the Station-Time domain, which forms a non-convex 2D space. After the preprocessing, the speed profile lattice is constructed in the s-t domain. After that, based on the weights which are composed of temporal smoothness and obstacles collision risk, the rough piecewise speed profiles are found using Dijkstra's algorithm. However, the speed profiles are composed of piecewise continuous lines, so it is not temporally smooth. In order to improve the smoothness of the speed profile, we model the speed optimization problem into a standard QP problem, which is much faster than a NLP problem with nonlinear constraints.

III. METHODOLOGY

A. PATH SEARCHING

The frenet coordinate system [1], [4], [20] is a popular technology in the field of autonomous driving because it has an excellent ability to well characterize the curve roads, which brings great convenience to the local planning. In the frenet coordinate system, we use line segments, arcs, polynomial curves, and Euler spirals to connect a series of waypoints. Finally a smooth reference line is built along a lane based on these curves. The above processing refers to the work of [21] and the OpenDRIVE[1] map format. [22] presents a simple and efficient technique to generate approximately arc-length parameterized spline curves that closely match the reference line of a lane. According to our experience, the length of each approximately arc-length parameterized spline curve needs to be selected reasonably. If these segments are too dense, this performance will consume a lot of space to store the coefficients of arc-length parameterized spline curves and the query time will be increased in the subsequent lookup of the coefficients table. And if the cubic spline segments are too sparse, this characteristic will increase the matching error with the reference line of a lane.

Sampling in the frenet coordinate system is a resolution complete [9] path planning technique by efficient discretization of the spatial domain. Hence, we can quickly find a feasible solution using graph-search-based methods. Next, considering the continuity and smoothness of the path, we use a class of curves to connect the two sampling states.

1) Motion Primitives

Polynomial spirals [12], [16]-[18], [24] are popular curves for path planning, which possess as many degrees of freedom as necessary to meet any number of constraints. However, we have to use numerical methods to solve the non-trivial

[1]http://www.opendrive.org/

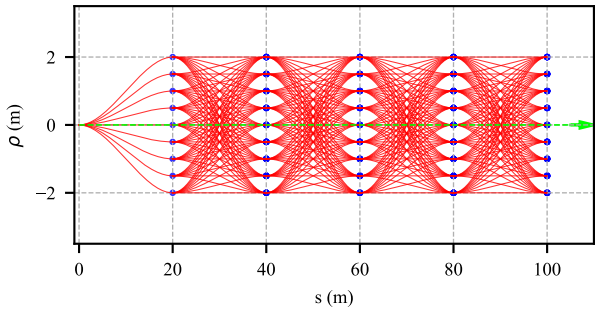


FIGURE 2. Path lattice in the frenet coordinate system. Blue vertices represent the sampled poses. The red motion primitives are cubic splines connecting vertices, and the dotted green line is the reference line of a lane.

constraints and the well-known Fresnel integrals as depicted in [24]. Usually, we can utilize a lookup table which is an efficient means of storing initial guesses for the parameters to make the numerical calculations converge quickly to a relatively accurate solution, but spirals are not as good as polynomials in terms of the computational efficiency. The cubic polynomials take only four parameters to represent a spatially smooth path. Hence, the cubic polynomial curves have a simple form and good performance in solving this two-point boundary value problem. Naturally the motion primitives are generated by connecting sampled endpoints using the cubic polynomials as depicted in [20], [25], [26]. As shown in (1), the function describing the relationship between the arc length s along a lane and the lateral offset ρ is designed to smoothly connect two states in the frenet coordinate system.

$$\rho(s) = a_0 + a_1 s + a_2 s^2 + a_3 s^3, s \in [s_0, s_f] \tag{1}$$

The boundary conditions are described as (2) [26], [27]. We can easily convert these differential constraints into a matrix expression, and the coefficients of the cubic polynomial can be determined as (3).

$$\begin{aligned} \rho(s_0) &= \rho_0 \\ \rho(s_f) &= \rho_f \\ \rho'(s_0) &= \tan\theta_{s_0} \\ \rho'(s_f) &= \tan\theta_{s_f} \end{aligned} \tag{2}$$

$$\begin{bmatrix} 1 & s_0 & s_0^2 & s_0^3 \\ 1 & s_f & s_f^2 & s_f^3 \\ 0 & 1 & 2s_0 & 3s_0^2 \\ 0 & 1 & 2s_f & 3s_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \rho_0 \\ \rho_f \\ \tan\theta_{s_0} \\ \tan\theta_{s_f} \end{bmatrix} \tag{3}$$

In (2) and (3), the start point state is $[s_0, \rho_0, \theta_{s_0}]^T$, and the endpoint state is $[s_f, \rho_f, \theta_{s_f}]^T$. In this paper, θ represents the heading of a vehicle in the Cartesian coordinate system, θ_s represents the heading in the frenet coordinate system, and θ_r represents the heading of the projection point of the rear axle midpoint of a vehicle in the reference line. The geometrical relationship between them can be expressed in (4).

$$\theta = \theta_s + \theta_r \tag{4}$$

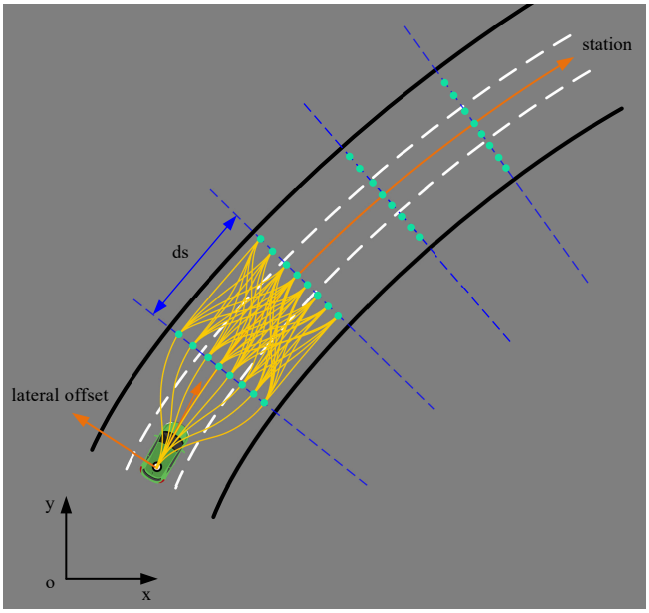


FIGURE 3. Path lattice in Cartesian coordinates. The green vehicle shows the current vehicle pose. Green vertices represent the sampled offset poses of the reference line along a lane, and ds represents the longitudinal interval between the sampled vertices.

The sampling mechanism including the interval distance ds between rows of endpoints and the total station horizon s_{max} along a lane is similar to the method in [28]. The characteristics of the sampling points are usually related to the road structure, the host vehicle's speed and the decisions of behavior planning. In reality, we need to adjust parameters to make them adapt to different scenarios, and this processing can increase the expressiveness and completeness of the solution space of our planner. For example, higher speed might need a longer sampling interval than lower speed, and a straight lane requires a longer sampling interval than a curved lane. As shown in Fig. 2 and Fig. 3, we construct a path lattice by sampling in the frenet coordinate system.

2) Cost Functions

A feasible path need to be found in the path lattice. According to Ziegler et al's work [11], they observe the difficulty of employing heuristic search algorithms, because it's hard to estimate the cost terms of smoothness and collision risk. They therefore propose an exhaustive rather than heuristic search algorithm to find the optimal path. In this paper, we also regard the state lattice as a DAG and employ Dijkstra's algorithm to search the optimal path based on the cumulative costs of motion primitives in the lattice.

When we construct a path lattice using motion primitives, the weights of motion primitives are also calculated. The total weights of each edge are a linear combination of the smoothness term, the lateral offset from the reference line term, and the obstacles collision risk term, as shown in (5).

$$c_{path} = c_{smooth} + c_{ref} + c_{obs} \tag{5}$$

Curvature can characterize the smoothness of a curve, so the smoothness cost term function is designed as the integral of the square of curvature along with the longitudinal station, which is described as (6).

$$c_{smooth} = w_{smooth} \int_{s_0}^{s_f} \kappa(s)^2 ds \tag{6}$$

It is difficult to get an explicit expression of the curvature function about the arc-length parameter, and in order to facilitate the calculation of the computer, we use numerical methods to calculate the integral of curvature's square. For a segment of a curve, we sample N points evenly on it, and an approximate calculation of the integral c_{smooth} can be expressed as the finite sum, which is shown in (7).

$$c_{smooth} = \frac{w_{smooth}}{N} \sum_{n=0}^{N} \kappa_i^2 \tag{7}$$

The curvature function of a curve is defined as:

$$\kappa = \frac{x'y'' - y'x''}{\sqrt[3]{x'^2 + y'^2}} \tag{8}$$

The curvature can be approximated using the finite differences of the sample waypoints as shown in (9).

$$\kappa_i \approx \frac{x_i' y_i'' - y_i' x_i''}{\sqrt[3]{x_i'^2 + y_i'^2}} \tag{9}$$

where,

$$\begin{aligned} x_i' &\approx \frac{x_{i+1} - x_i}{t} \\ x_i'' &\approx \frac{x_{i+2} - 2x_{i+1} + x_i}{t^2} \\ y_i' &\approx \frac{y_{i+1} - y_i}{t} \\ y_i'' &\approx \frac{y_{i+2} - 2y_{i+1} + y_i}{t^2} \end{aligned} \tag{10}$$

Because the coordinates of the cubic polynomials are represented using frenet coordinates $(s - \rho)$, so we need to convert the coordinates into the Cartesian coordinates $(x - y)$ to calculate the curvature using (9). The coordinates conversion function is given by [18] as shown in (11).

$$\begin{aligned} x(s,\rho) &= x_r(s) + \rho\cos(\theta_r(s) + \pi/2) \\ y(s,\rho) &= y_r(s) + \rho\sin(\theta_r(s) + \pi/2) \\ \theta(s,\rho) &= \theta_r(s) \\ \kappa(s,\rho) &= (\kappa_r(s)^{-1} + \rho)^{-1} \end{aligned} \tag{11}$$

where $\theta_r(s)$ represents the heading angle of the reference line, $x_r(s)$ and $y_r(s)$ represent the Cartesian coordinates of the reference line, and $\kappa_r(s)$ represents the curvature of the reference line. In this paper, we define the lateral offset ρ to be positive on the left side of the reference line and negative on the right side.

Usually, the trajectory of the vehicle should follow the reference line of a lane, which means this maneuver can well comply with the traffic rules, and it is also a relatively safe driving strategy. The lateral offset ρ from the reference line is then added to the total edge weights as a penalty. c_{ref} can also be expressed as the finite sum as shown in (12).

$$c_{ref} = \frac{w_{ref}}{N} \sum_{n=0}^{N} (\rho_i - \rho_{ref})^2 \tag{12}$$

where ρ_{ref} represents the lateral offset of the reference line in $s - \rho$. In this paper, the offset ρ_{ref} of the reference line in the rightmost lane is $0.0m$.

One method to calculate the collision risk with obstacles is to calculate the distance to all obstacles [18]. However, this strategy will increase the computational complexity because the distance from all points on the path to all obstacles needs to be calculated one by one. The computation expense is $\mathcal{O}(NM)$, where N is the number of the path sampling points, M is the number of obstacles. Another strategy is to assign the value of collision risk to be $False$ or $True$, which depends on whether there is a collision maneuver for a path [29]. However, for path candidates with no collision, it is obvious that the path closer to obstacles has higher collision risk, which is ignored by this strategy of the binary collision risk index. Therefore, this paper introduces the convolution collision risk indicator, which is successfully used in [25], [26].

The value of a collision checking is given by [26]. They penalize a path that passes solid lane lines by assigning the value of the collision checking to be 0.5, and they assign 0.2 to a path which passes dashed lane lines. If a path does not pass any obstacle and road boundary, the value is defined as 0.0. Instead, if a path hits an obstacle or road boundary the value is defined as 1.0. The function of collision checking can be described as (13).

$$r[i] = \begin{cases} 1.0 & if(passing\ obstacles) \\ 0.5 & if(passing\ solid\ lane\ lines) \\ 0.2 & if(passing\ dashed\ lane\ lines) \\ 0.0 & if(no\ collision) \end{cases} \tag{13}$$

In order to reasonably express the security cost term, the risk index of a path is calculated by discrete Gaussian convolution [25] combined with collision checks as follows.

$$c_{obs}[i] = w_{obs} \sum_{k=1}^{n_\rho} r[k]g[i-k] \tag{14}$$

where,

$$g[i] = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{i^2}{2\sigma^2}) \tag{15}$$

where, i is the index of each candidate path, n_ρ is the number of endpoints sampled laterally, and σ is the standard deviation of the Gaussian kernel $g[i]$. The value of σ indicates the influence scope of collision risk because a collision path poses a collision risk to the nearby paths. Usually, the larger the value is, the higher the collision risk of an adjacent path will be.

The short-term target point cannot be determined explicitly in on-road driving, and because of the existence of the

smoothness cost term, the offset from the reference line cost term, and the collision risk cost term, it is difficult to design the heuristic function for lattice searching. Hence, we use an exhaustive rather than heuristic search, as described in [12]. In this paper, Dijkstra's algorithm is adopted to find the shortest path to generate a kinematically-feasible, relatively smooth, collision-free path in the nonconvex spatial space. Unlike the traditional Dijkstra's algorithm, we do not specify a target pose in advance, but design a $TARGET$ list to store the sampled target poses set. When poses (endpoints) in the list are all expanded, the pose with the least cost is selected as our short-term target pose. The feasible path generated by the lattice searching looks smooth, but we find that the curvature profile has step changes and is jerky after calculating the curvature of the path. Considering the above factors, the smoothness performance is very poor, which brings great difficulties to the trajectory tracking module. In addition, according to the sampling mechanism, the heading of each endpoint of a path segment is always consistent with the reference line. This sampling mechanism limits the motion potential of the vehicle. And when multiple such motion primitives are connected, the heading of the path generated by lattice searching possibly changes frequently. Therefore, the generated path does not meet the smoothness, optimality and mobility requirements and also violates human driving habits.

B. PATH OPTIMIZATION

1) Objective Function

The lattice searching can generate a continuous path, but the curvature profile of the path maybe not continuous, as shown in Fig. 5 (a), (b). Therefore, an optimization process of the rough path is introduced, which makes the final path not only meet internal and external constraints but also smooth enough for an autonomous vehicle to track. The optimal path is defined as the one that minimizes the cost function which can be expressed as the finite sum as follows

$$j(\rho_1, \rho_2, \cdots, \rho_{N_s}) = \sum_{i=1}^{N_s-3} L(\rho_i, \dot{\rho}_i, \ddot{\rho}_i, \dddot{\rho}_i)e \tag{16}$$

with

$$L = w_1(\rho_i')^2 + w_2(\rho_i'')^2 + w_3(\rho_i''')^2 + w_4(\rho_i - \rho_r(s_i))^2$$

where $\rho_r(s_i)$ is the function of the rough path generated by the path lattice searching. ρ_i', ρ_i'' and $\rho_i'''$are related to the frenet heading angle, the first derivative and second derivative of the frenet heading angle in the frenet coordinate system. The objective function has a good balance between spatial smoothness and followability to the rough path.

To numerically calculate the objective function, the path is approximated by N_s waypoints which are sampled at equidistant longitudinal station distance as follows:

$$s_i = s_0 + ie, i = 0, 1, \cdots, N_s \tag{17}$$

where e is the sampling step size. The station derivatives of ρ are approximated by differences of the sampling waypoints as follows:

$$\begin{aligned} \rho_i' &\approx \frac{\rho_{i+1} - \rho_i}{e} \\ \rho_i'' &\approx \frac{\rho_{i+2} - 2\rho_{i+1} + \rho_i}{e^2} \\ \rho_i''' &\approx \frac{\rho_{i+3} - 3\rho_{i+2} + 3\rho_{i+1} - \rho_i}{e^3} \end{aligned} \tag{18}$$

2) Constraint Functions

The optimal path must minimize the objective function (16), but at the same time, the optimization needs to obey a set of internal and external constraints. Internal constraints are introduced by vehicle kinematics and dynamics limitations. In path optimization, internal constraints include curvature constraints which are brought by Ackerman steering geometry. Curvature constraints can be expressed as (19), and their approximate numerical calculations are shown as (9).

$$|\kappa_i| \leq \kappa_{\lim} \tag{19}$$

External constraints are imposed by the driving corridor [3], obstacles, and boundary conditions. Unlike the work in [3], [30], we won't transform obstacles into the convex polygons, which means we don't need to convert non-convex obstacles into the convex constraints. Because the path lattice searching will find a feasible solution close to the optimal solution in a non-convex feasible domain, we only need to use this solution as the initial value to iterate several times to converge to an optimal solution.

In order to perform the collision checking, we have to consider the shape of the ego vehicle, so we introduce the vehicle discs as shown in Fig. 4. We decompose the ego vehicle into three discs with a radius of r_{veh} along the longitudinal axis of the vehicle. The center of one of the discs coincides with the midpoint of the rear axle of the vehicle, and the distance between the centers of the discs is d. The red points represent centers of the discs. Orange curves represent the frenet frames in the lane. The blue points represent the projection points of the red points in the frenet coordinate system.

In this paper, the coordinates $[s_i, \rho_i]^T$ of the center point of the rear axis are given by the localization module, so the center coordinates of the remaining two discs can be obtained using approximate numerical calculations as follows:

$$\begin{aligned} s_{2i} &= s_i + d\cos(\theta(s_i) - \theta_r(s_i)) \\ \rho_{2i} &= \rho_i + d\sin(\theta(s_i) - \theta_r(s_i)) \\ s_{3i} &= s_{2i} + d\cos(\theta(s_i) - \theta_r(s_{2i})) \\ \rho_{3i} &= \rho_{2i} + d\sin(\theta(s_i) - \theta_r(s_{2i})) \end{aligned} \tag{20}$$

Collision avoidance constraints can be written as:

$$\begin{aligned} (s_i - s_j)^2 + (\rho_i - \rho_j)^2 &> (r_{veh} + r_{obs})^2 \\ (s_{2i} - s_j)^2 + (\rho_{2i} - \rho_j)^2 &> (r_{veh} + r_{obs})^2 \\ (s_{3i} - s_j)^2 + (\rho_{3i} - \rho_j)^2 &> (r_{veh} + r_{obs})^2 \end{aligned} \tag{21}$$

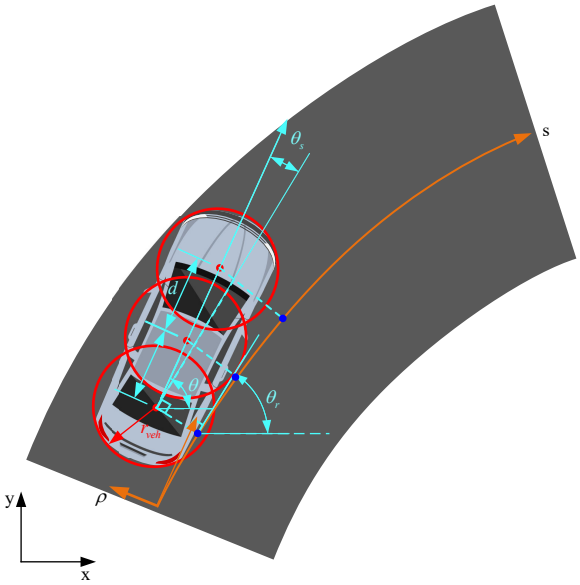


FIGURE 4. The vehicle discs containing the host vehicle body.

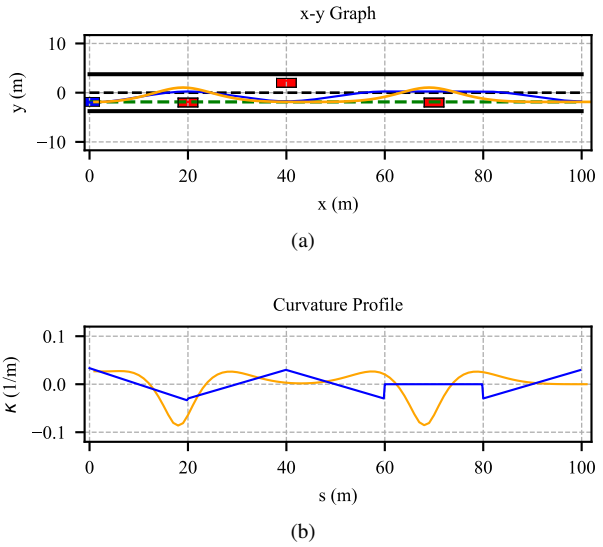


FIGURE 5. Path optimization. (a) Path optimization on structured roads. The blue curve represents the rough path generated by the lattice searching. The orange curve represents the refined path generated by path optimizer. The green dotted line represents the reference line of the right lane. And these three red boxes represent obstacles. (b) Curvature profile of path optimization. The blue curve represents the curvature profile of the rough path, and the orange curve represents the curvature profile of the refined path.

where the frenet coordinate of the jth obstacle is $[s_j, \rho_j]^T$. r_{obs} represents the expansion radius of an obstacle. These constraints create $3n_oN_s$ inequations, where n_o is the number of obstacles. At the same time, the vehicle cannot pass the boundaries of the driving corridor while driving to follow traffic rules, so the following inequality constraints are formed:

$$\rho_{\min} \leq \rho_i \leq \rho_{\max} \tag{22}$$

where $\rho_{\max}$ and $\rho_{\min}$ are the left and right boundaries, respectively, so the number of the driving corridor inequalities here is $2N_s$.

The optimization process also needs to meet the start point state constraints to match the initial heading and curvature. The initial state of the host vehicle given by the perception module is $\mathbf{x}_0 = [x_0, y_0, s_0, \rho_0, \theta_0, \kappa_0]^T$. So the equality constraints formed by the initial state are:

$$\begin{aligned}
&\theta_{s_0} = \theta_0 - \theta_r(s_0) \\
&\tan\theta_{s_0} = \frac{\rho_1 - \rho_0}{s_1 - s_0} \\
&\kappa_0 = \frac{(x_1 - x_0)(y_2 - 2y_1 + y_0) - (y_1 - y_0)(x_2 - 2x_1 + x_0)}{\sqrt[3]{(x_1 - x_0)^2 + (y_1 - y_0)^2}}
\end{aligned} \tag{23}$$

According to (23), the decision variables ρ_1 and ρ_2 can be expressed as:

$$\begin{aligned}
\rho_1 &= e\tan(\theta_0 - \theta_r(s_0)) + \rho_0 \\
\rho_2 &= \frac{A - B + C}{D}
\end{aligned} \tag{24}$$

where,

$$\begin{aligned}
A &= \kappa_0 \sqrt[3]{(x_1 - x_0)^2 + (y_1 - y_0)^2} \\
B &= (x_1 - x_0)(y_r(s_2) - 2y_1 + y_0) \\
C &= (y_1 - y_0)(x_r(s_2) - 2x_1 + x_0) \\
D &= (x_1 - x_0)\sin(\theta_r(s_2) + \frac{\pi}{2}) - (y_1 - y_0)\cos(\theta_r(s_2) + \frac{\pi}{2})
\end{aligned} \tag{25}$$

where x_1, y_1, x_2, y_2 can be calculated according to (11).

Finally, a problem to optimize the rough path is transformed into a nonlinear constrained optimization problem that minimizes the performance index (16) and satisfies the nonlinear constraints and nonlinear boundary conditions. For convenience of notation, we organize all the constraints into the following form:

$$\begin{aligned}
\min \quad & j(\boldsymbol{\rho}) = \sum_{i=1}^{N_s - 3} L(\rho_i, \dot{\rho}_i, \ddot{\rho}_i, \dddot{\rho}_i)e \\
s.t. \quad & \mathbf{Aeq} \cdot \boldsymbol{\rho} = \mathbf{Beq} \\
& \mathbf{C}(\boldsymbol{\rho}) \leq \mathbf{0} \\
& \mathbf{LB} \leq \boldsymbol{\rho} \leq \mathbf{UB}
\end{aligned} \tag{26}$$

where $\mathbf{Aeq} \cdot \boldsymbol{\rho} = \mathbf{Beq}$ is the equality constraints formed by boundary conditions, $\mathbf{C}(\boldsymbol{\rho}) \leq \mathbf{0}$ is inequality constraints consisting of (19) and (21), and $\mathbf{LB} \leq \boldsymbol{\rho} \leq \mathbf{UB}$ is the driving corridor constraints of the decision variables. $\boldsymbol{\rho} = [\rho_0, \cdots, \rho_{N_s}]^T$ is the vector of the decision variables.

In (26), the objective function is a quadratic form which is twice differentiable. And the constraint functions are composed of nonlinear equations and nonlinear inequalities. In

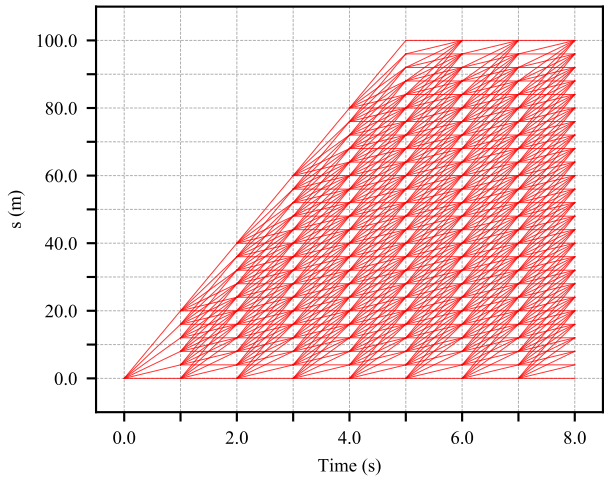


FIGURE 6. Speed profile lattice in the Station-Time domain.

order to solve this problem quickly, a nonlinear constrained optimization solver is picked as shown in Section IV. Fig. 5 (a) shows the rough path (the blue curve) and the optimized path (the orange curve) on a straight road with three obstacles. It can be seen from this figure that both paths seem to be continuous and smooth. Fig. 5 (b) shows the curvature profile (the blue curve) of the rough path contains three step changes, and the curvature profile (the orange curve) of the refined path is spatially smooth.

C. SPEED PROFILE SEARCHING

In recent years, the Station-Time (s-t) graph used by speed profile planning is a popular method [4], [31], [32]. In general, obstacles information is mapped to the s-t graph, which may make the s-t graph become a non-convex domain. In order to generate a temporally smooth speed profile in this non-convex domain, sampling-based methods [33], optimization-based methods [31],[34], and combination-based methods [4], [18], [35] are used widely. And in order to ensure that the generated speed profile is optimal and to prevent the speed profile from falling into a local minimum state, combination-based methods are preferred. It is convenient to explicitly express the mathematical relationship between the longitudinal station and the time in the s-t graph. The planning result also explicitly reflect the relative position between the ego vehicle and moving obstacles in the s-t graph. We can clearly see whether the ego vehicle's behavior is overtaking, following or stopping or not. Speed planning in this paper is divided into two layers like path planning. Firstly, we sample in the s-t graph and search for a rough speed profile, then the QP algorithm is introduced to optimize the rough speed profile.

1) Speed Profile Primitives

We sample some vertices at equal intervals in the s-t graph, then connect these sampled vertices using straight lines to construct a speed profile lattice as shown in Fig. 6. The efficiency of the algorithm is closely related to the resolution of the speed lattice. A lower resolution may result in a larger acceleration or deceleration of a generated speed profile, which affects the temporal smoothness of the trajectory. A higher resolution will produce a smoother speed profile, but it is a challenge in terms of computational efficiency. Fig. 6 shows the time interval $\Delta t = 1.0s$ and the station interval $\Delta s = 4.0m$. Of course, we can adjust the resolution of the lattice to adapt to different scenarios. On the time axis, time series can be expressed as:

$$t_i = t_0 + i\Delta t, i = 0, 1, ..., n \tag{27}$$

According to the sampling interval Δt, the time derivatives of s are approximated by the finite differences as follows:

$$\begin{aligned} s_i' &= v_i \approx \frac{s_i - s_{i-1}}{\Delta t} \\ s_i'' &= a_i \approx \frac{s_i - 2s_{i-1} + s_{i-2}}{\Delta t^2} \\ s_i''' &= j_i \approx \frac{s_i - 3s_{i-1} + 3s_{i-2} - s_{i-3}}{\Delta t^3} \end{aligned} \tag{28}$$

2) Cost Functions

Like the path planning step, we also need to assign the weights to each edge. Considering the feasibility, smoothness, safety, and optimality of the generated trajectory, the cost function we designed is also a linear combination of the offset from the reference speed v_{ref}, acceleration and jerk penalties, and collision risk with obstacles, as shown in (29).

$$c_{speed} = c_{ref} + c_{acc} + c_{jerk} + c_{obs} \tag{29}$$

c_{ref} represents the cost term of the offset from the reference speed, which shows that we do not expect the planning speed to drift too far from the reference speed. This term can be defined as follows:

$$c_{ref} = w_{ref}(s_i' - v_{ref})^2 \tag{30}$$

where the reference speed v_{ref} depends on the speed of the vehicle ahead, traffic rules, and road structure.

c_{acc} and c_{jerk} represent the penalty terms of acceleration and jerk, respectively, which is shown as (31). These two terms can make the trajectory smoother by dampening rapid changes in the speed and acceleration profile.

$$\begin{aligned} c_{acc} &= w_{acc}(s_i'')^2 \\ c_{jerk} &= w_{jerk}(s_i''')^2 \end{aligned} \tag{31}$$

The cost term c_{obs} describes the collision risk with obstacles. In dynamic scenarios, risk assessment can be performed using some risk indicators, such as the Time-To-Collision (TTC) [36], Distance-To-Collision (DTC) [37] or Time-To-React (TTR) [38]. This paper uses a combination of the physics-based motion models and the curved lane models to predict the short-term trajectory of a vehicle. Although this prediction does not take the uncertainty of the vehicle and the error of the motion model into account, the predicted trajectory remains valid in the short term. And the computational complexity of this prediction method also makes it suitable

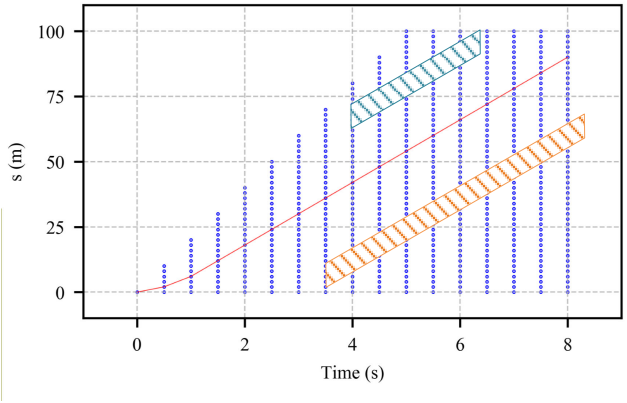


FIGURE 7. The rough speed profile in the Station-Time domain. The blue points are sampled vertices, and two parallelograms are the predicted trajectories of obstacles.

for online execution. And based on the bicycle model and the road structure, the single trajectory simulated forward is projected into the s-t graph. At last, based on the projection trajectory, we use the index DTC as our risk indicator. At a certain timestamp, we calculate the distance of the ego vehicle to obstacles and select the minimum distance as the value of the DTC . The c_{obs} is represented as follows:

$$c_{obs} = \frac{w_{obs}}{DTC + \delta} \tag{32}$$

where $\delta = 0.01$.

In the speed profile searching step, we can generate a rough speed profile using Dijkstra's algorithm as shown in Fig. 7. In this figure, the ego vehicle starts moving from rest as shown by the red curve. Two parallelograms are the predicted trajectories of obstacles.

D. SPEED PROFILE OPTIMIZATION

1) Objective Function

The generated rough speed profile is formed by the connection of many straight segments, so the kinematic constraints, dynamic constraints, and temporal smoothness are not satisfied. Obviously, a trajectory with step changes in the speed profile and acceleration profile is not feasible for a vehicle, so we also introduce an optimization method to improve the performance of the rough speed profile like path optimization. In the s-t graph, a speed profile is a one-to-one mapping between the timestamp and the station along a lane. Generally there are two optimization strategies in the s-t graph as described in [31]: one strategy is to optimize the decision variables $\mathbf{s} = [s_1, s_2, ..., s_n]^T$ based on the fixed timestamps as discussed in [35], [39], and the other strategy is to optimize the decision variables $\mathbf{t} = [t_1, t_2, ..., t_n]^T$ based on the fixed longitudinal stations as discussed in [31], [40]. For the convenience of the computation, this paper refines the rough speed profile via station optimization. And the speed optimization problem is transformed into a QP problem, so the speed optimization problem can also be solved by a local, continuous method.

The optimal speed is defined as the one that minimizes the finite sum as follows

$$j = \sum_{i=1}^{N_t} (w_{vel}(v_i - v_{ri})^2 + w_{acc}a_i^2 + w_{jerk}jerk_i^2)\varepsilon \tag{33}$$

with

$$L = w_{vel}(s_i' - v_r(t_i))^2 + w_{acc}(s_i'')^2 + w_{jerk}(s_i''')^2$$

We discretize the temporal space into the following form:

$$t_i = t_0 + i\varepsilon, i = 0, 1, ..., N_t \tag{34}$$

where ε is the step of time, and N_t indicates that the total temporal horizon T is divided into N_t discrete timestamps.

The corresponding decision variables that need to be optimized are expressed as $\mathbf{s} = [s_1, s_2, ..., s_n]^T$. The time derivatives of $\mathbf{s}$ are approximated by finite differences of the discrete waypoints.

$$\begin{aligned} s_i' &= v_i \approx \frac{s_{i+1} - s_i}{\varepsilon} \\ s_i'' &= a_i \approx \frac{s_{i+2} - 2s_{i+1} + s_i}{\varepsilon^2} \\ s_i''' &= jerk_i \approx \frac{s_{i+3} - 3s_{i+2} + 3s_{i+1} - s_i}{\varepsilon^3} \end{aligned} \tag{35}$$

Obviously, the objective function is a quadratic convex function. The first term is the penalty cost about the offset from the rough speed profile $v_r(t)$. This cost indicates that the speed of the ego vehicle should follow the rough speed profile as much as possible, which ensures that the speed optimization can converge to the global optimum. v_{ri} is the rough speed obtained from the rough speed profile. The vector $\mathbf{s}$ obtained from the rough speed profile is used as the initial value to feed QP. The remaining two terms in (33) are the penalties for acceleration and jerk, which makes the generated speed profile smoother. And for the computational convenience, we convert the objective functional (33) into a standard quadratic form (SQF) expressed by matrix manipulations. The details of the transformations can be found in Appendix A.

2) Constraint Functions

Constraints also can be separated into two classes in the QP speed optimization step, internal and external constraints. Internal constraints come from the kinematics and dynamics limitations of the vehicle, including the speed limits, acceleration limits $a_{\lim}$ and jerk limits $jerk_{\lim}$, which are essentially brought about by the physical limits of the vehicle's powertrain and the adhesion of tires. External constraints come from the driving scenarios, including the road speed limits $v_{\lim}$, collision avoidance, boundary conditions, and so on. In this paper, acceleration constraints, jerk constraints, and lane speed limits can be defined as follows:

$$\begin{aligned} |s_i''| &\le a_{\lim} \\ |s_i'''| &\le jerk_{\lim} \\ |s_i'| &\le v_{\lim} \end{aligned} \tag{36}$$

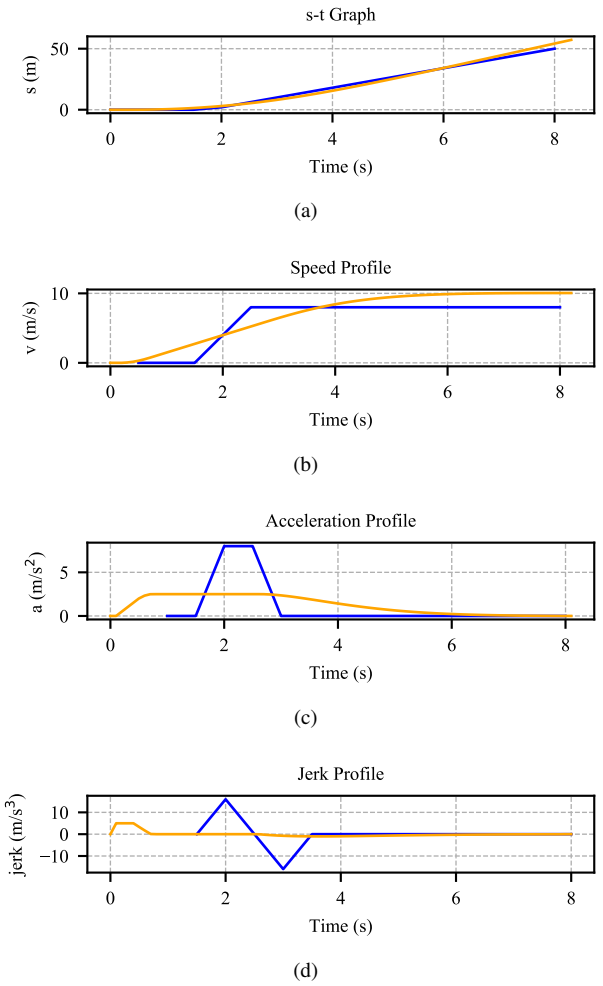


FIGURE 8. Speed profile optimization. (a) The blue curve represents the rough Station-Time profile, and the orange curve represents the post-optimized Station-Time profile. (b) The blue curve represents the rough speed profile, and the orange curve represents the post-optimized speed profile. (c) The blue curve represents the rough acceleration profile, and the orange curve represents the post-optimized acceleration profile. (d) The blue curve represents the rough jerk profile, and the orange curve represents the post-optimized jerk profile.

For each decision variable s_i, there are upper and lower bounds $(s_0, s_{\max})$ which can be expressed as:

$$s_0 \leq s_i \leq s_{\max} \tag{37}$$

And the collision avoidance constraints can be described as:

$$\begin{aligned} |s_{j,i} - s_i| &> r_{veh} + r_{dyn} \\ |s_{j,i} - s_{2i}| &> r_{veh} + r_{dyn} \\ |s_{j,i} - s_{3i}| &> r_{veh} + r_{dyn} \end{aligned} \tag{38}$$

where $s_{j,i}$ represents the coordinate of the jth obstacle at t_i in station domain. r_{dyn} represents the expansion radius of the dynamic obstacles. s_{2i} and s_{3i} represent the coordinates of the remaining vehicle discs in station domain. Due to the existence of s_{2i}, s_{3i}, the obstacle avoidance constraints become highly nonlinear constraints. To linearize these inequalities, we scale up the inequalities, which is shown in (39).

$$\begin{aligned} s_{2i} &= s_i + d\cos(\theta(s_i) - \theta_r(s_i)) \leq s_i + d \\ s_{3i} &= s_{2i} + d\cos(\theta(s_i) - \theta_r(s_{2i})) \leq s_i + 2d \end{aligned} \tag{39}$$

The amplification of s_{2i}, s_{3i} is of physical significance because this transformation increases the safe distance of the ego vehicle to obstacles in this QP step.

And the boundary value conditions form the equality constraints. The start point and the end point in the speed optimization step are $\mathbf{x}_0 = [s_0, t_0, v_0, a_0, jerk_0]^T$ and $\mathbf{x}_f = [s_f, t_f, v_f, a_f, jerk_f]^T$, respectively. And both speed and acceleration are expected to be 0.0 at the endpoint of the trajectory, which means that the ego vehicle will move at a constant speed after the optimization. Hence, the terminal conditions can be written as (40).

$$\begin{aligned} v_n &\approx \frac{s_{n+1} - s_n}{\varepsilon} \\ a_n &\approx \frac{s_{n+2} - 2s_{n+1} + s_n}{\varepsilon^2} = 0 \\ jerk_n &\approx \frac{s_{n+3} - 3s_{n+2} + 3s_{n+1} - s_n}{\varepsilon^3} = 0 \end{aligned} \tag{40}$$

At the endpoint of the trajectory, both ε and a_{n-1} are extremely small, so we can write the following equality constraints according to the boundary value conditions.

$$\begin{aligned} s_0 &= s_0 \\ s_1 &= \varepsilon v_0 + s_0 \\ s_2 &= a_0\varepsilon^2 + s_0 + 2\varepsilon v_0 \\ s_3 &= jerk_0\varepsilon^3 + 3\varepsilon v_0 + 3a_0\varepsilon^2 + s_0 \\ s_{n+1} &= 2s_n - s_{n-1} \\ s_{n+2} &= 3s_n - 2s_{n-1} \\ s_{n+3} &= 4s_n - 3s_{n-1} \end{aligned} \tag{41}$$

At last, all the above constraints (36), (37), (38) and (41) are linear, so the feasible domain is a convex space in $\mathbb{R}^{n+3}$. To facilitate numerical calculations, all constraints are also converted into the forms of matrix manipulations. The details of the transformations can be found in Appendix B. Finally, we successfully transform the speed profile optimization problem into an optimization problem with a quadratic objective function and linear constraints. This optimization problem is a typical medium-scale QP problem. We transform the objective function and the constraint functions into a standard form (SF), which is described as (42). A detailed explanation of the coefficients in (42) can be found in Appendix A and B.

$$\begin{aligned} \min \quad & j(\mathbf{s}) = \frac{1}{2}\mathbf{s}^T\mathbf{H}\mathbf{s} + \mathbf{q}\mathbf{s} + p \\ s.t. \quad & \mathbf{G}\mathbf{s} \leq \mathbf{h} \\ & \mathbf{A}\mathbf{s} = \mathbf{b} \end{aligned} \tag{42}$$

In Fig. 8 (a), the blue curve represents the rough s-t profile generated by the speed lattice searching, and the orange curve represents the refined s-t profile generated by QP. We can see that the shapes of the two curves are similar. In Fig .8 (b), (c),

(d), the profiles of speed, acceleration and jerk generated by QP are temporally smoother than those generated by lattice searching.

IV. CASE STUDIES

To evaluate the algorithm performance we proposed, we built a $1,163m$ simulated road with multiple challenging scenarios. Our simulation environment consists of static and dynamic obstacles and simulated two-lane roads. Some common driving scenarios are added to the simulated structured environment, including lane changing, following, obstacles avoidance, overtaking, and stopping. We show four cases to test the performance of the framework in this paper. In our simulation environment, the host vehicle is represented by a blue box and the surrounding vehicles are represented by boxes of other colors. The temporal information is represented by the depth of color, where deeper color represents a later time step.

In our simulator, the perception module can be removed because all the environmental information and vehicle state information can be directly acquired from the simulator. The behavior planning (decision-making) layer uses a finite state machine [41-42] model to make decisions based on the predefined behaviors and scenarios. Due to the clear logic and low computational complexity, this rule-based decision maker is fully capable of handling simple scenarios. For decision makers based on the partially observable Markov decision process (POMDP) [43], the computational complexity is too high to make it unsuitable for real-time operation. In order to validate the planning algorithm in this paper, using the state machine model as the core of the decision-making layer is a nice choice. And the behavior prediction layer combines the physical model of vehicles with the curved road model to predict the behaviors and trajectories of surrounding vehicles. Due to the existence of the nonholonomic constraints and dynamic constraints, we use the model predictive control (MPC) [44] controller to track the trajectories the decoupled planner generated and plot the historical trajectories of the host vehicle and all surrounding vehicles. In addition, compared to the pure pursuit controller [45] and the rear wheel based feedback controller [45], the MPC controller can maintain proper tracking accuracy and has good control over the host vehicle at high speed. Therefore, the trajectory planner only needs to generate an executable trajectory combining the path and the temporal information to feed into the controller.

The planner is implemented in Python3.6 scripts and runs on a laptop with 2.6GHz Intel Core i7-6500U and 8GB RAM in Ubuntu 16.04.6 LTS. The path optimizer utilizes the open source nonlinear optimization tool CasADi [46]. The solver of the interior point method is the open source optimization library IPOPT [47]. And the speed profile optimizer and the MPC controller utilizes the open source optimization library CVXOPT[2] and CVXPY[3], respectively.

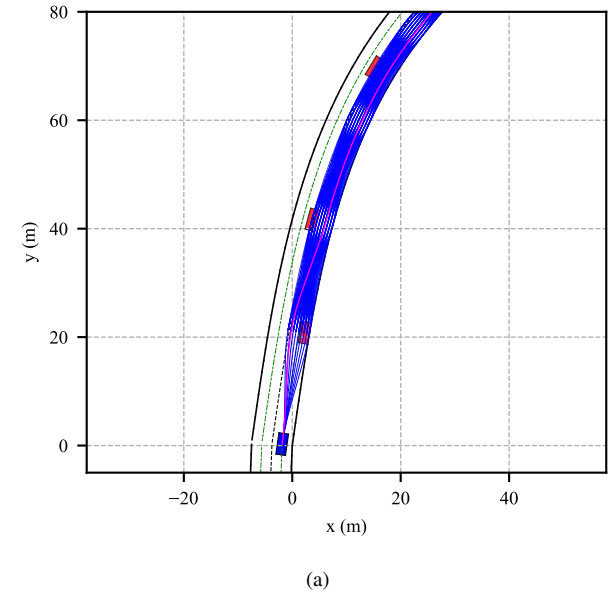
A. CASE1: STATIC OBSTACLES AVOIDANCE

Static obstacles avoidance is one of the most basic functions for trajectory planner. The host vehicle expects a spatiotemporally smooth trajectory that can bypass obstacles without collisions with obstacles and road boundaries. In this case, the ego vehicle starts moving from rest, and there are three static obstacles along the right lane. Fig. 9 (a) shows the rough path generated in the path lattice is able to avoid obstacles and looks smooth enough. The magenta curve represents the rough path, and blue curves represent the motion primitives that construct the path lattice. Fig. 9 (b) shows the rough path and the refined path, respectively. In this graph, the magenta curve represents the rough path, and the orange curve represents the refined path. Fig. 9 (c) shows the magenta curvature profile of the rough path is not smooth due to the existence of oscillations and step changes. And the orange curve represents the curvature profile of the refined path. The results show that the curvature of the path is well improved by the path optimizer. The curvature profile of the refined path removed the shocks and improved the smoothness of the path. Fig. 10 shows the speed, acceleration and jerk profiles of the ego vehicle, which are temporally smooth and Fig. 11 records the trajectory of the ego vehicle generated by the MPC controller. The static obstacles avoidance test verifies that our framework can output a spatiotemporally smooth and feasible trajectory for the trajectory tracking module to execute.

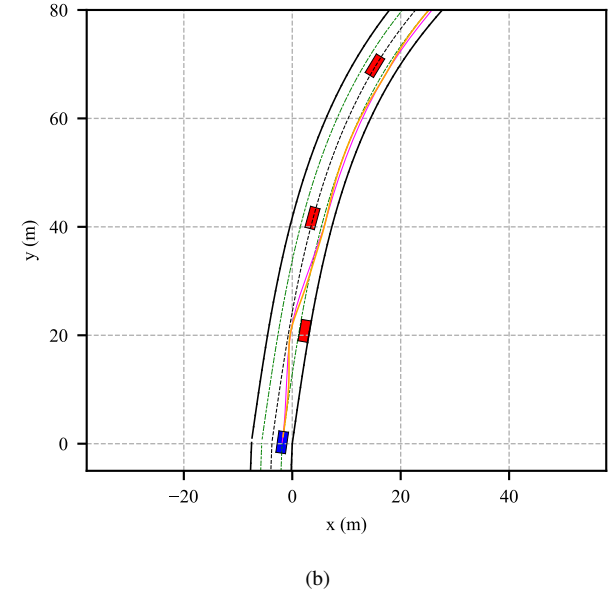
B. CASE2: AUTOMATIC LANE CHANGING

Lane changing is also one of the most common behaviors in structured environments. According to our predictive model, obstacles continue to move forward along the lane with the current detection speed. We set up slow-moving vehicles ahead to test the planner's lane changing performance, where the speed of obstacle vehicles is $3.0m/s$. Fig. 12 (a) shows the rough path in the scenario of lane changing. In Fig. 12 (b), the magenta curve represents the rough path and the orange curve represents the refined path. Fig. 12 (c) shows the curvature profiles of the rough path and the refined path. And the maximum curvature of the refined path is $0.04m^{-1}$. Fig. 13 shows that the ego vehicle's speed should be accelerated from $8m/s$ at $t = 12.4s$ to $11.3m/s$ at $t = 15.2s$ during the lane changing. We set the maximum acceleration and deceleration values to be $2.5m/s^2$ and $-2.5m/s^2$, respectively. And we set the maximum jerk value to be $5m/s^3$. In this graph, we can see that the speed profile, the acceleration profile and the jerk profile are smooth. And the speed, acceleration, and jerk did not exceed the thresholds. Fig. 14 records the results of trajectory tracking. The blue trajectory belongs to the host vehicle, and the other trajectories belong to the surrounding obstacles.

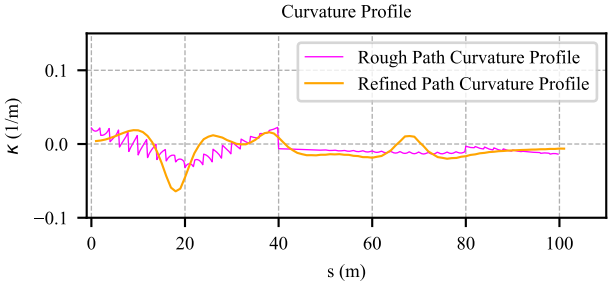
[2]http://cvxopt.org/
[3]https://www.cvxpy.org/



(a)



(b)



(c)

FIGURE 9. Static obstacles avoidance. (a) The blue lattice is constructed in the right lane. The magenta curve represents the rough path. The blue box and the red boxes represent the ego vehicle and obstacles respectively. (b) Path optimization. The magenta curve represents the rough path. The orange curve represents the refined path. The green dotted curves represent the reference line for each lane separately. (c) Curvature profile of path optimization.

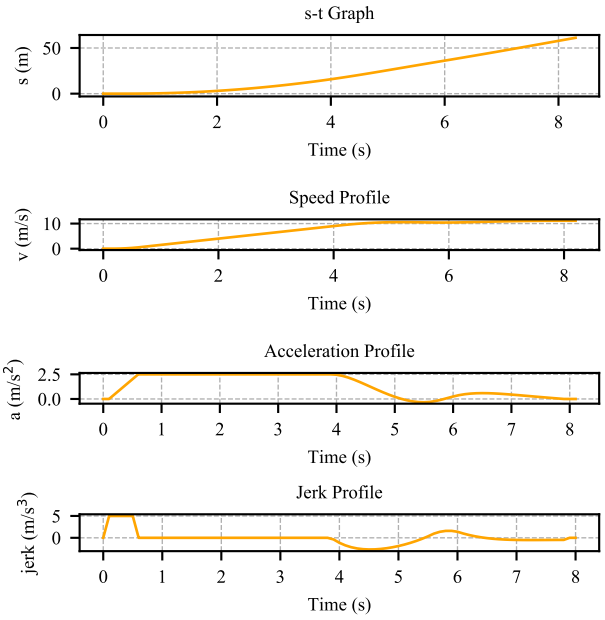


FIGURE 10. Speed profile, acceleration profile and jerk profile generated by the speed profile optimizer in the scenario of static obstacles avoidance.

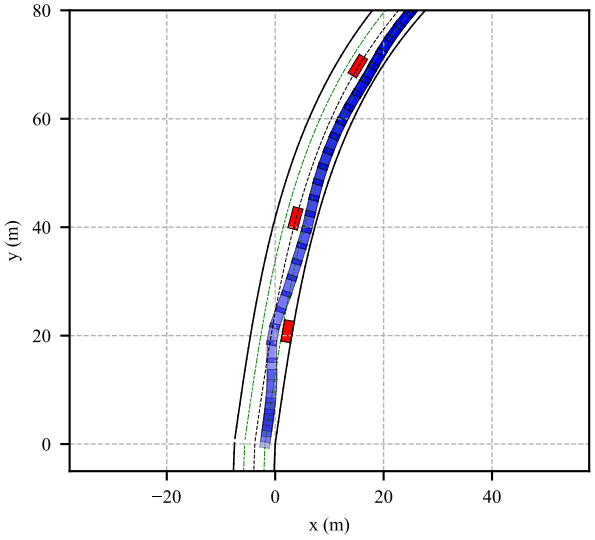


FIGURE 11. The host vehicle trajectory in the scenario of static obstacles avoidance.

C. CASE3: STOPPING

In the stopping scenario, we set a static obstacle in the left lane as shown by the orange box, and the decision making layer does not give the instruction to perform lane changing, so the ego vehicle should stop when it encounters this static obstacle as displayed in Fig. 15. Fig. 15 (a) shows the total distance horizon sampled forward of the path lattice is $80m$. For this path lattice with 4 sets of longitudinal sampling waypoints with an interval of $20m$ and 9 sets of lateral sampling waypoints with an interval of $0.5m$, Dijkstra' algorithm can

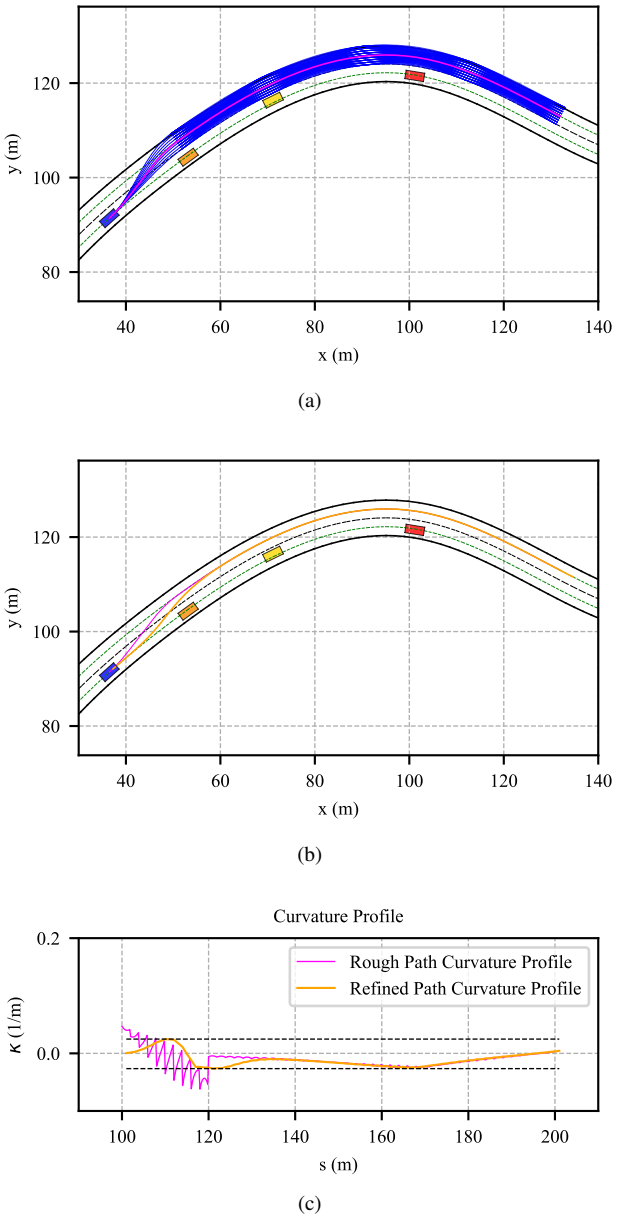


FIGURE 12. The lane change scenario. (a) The blue lattice is constructed in the left lane. The magenta curve represents the rough path. The blue box and the remaining boxes represent the ego vehicle and dynamic obstacles respectively. (b) Path optimization. The magenta curve represents the rough path. The orange curve represents the refined path. The green dotted curves represent the reference line for each lane respectively. (c) Curvature profile of path optimization.

search the path lattice to generate a rough path using $0.08s$. The magenta curve represents the rough path. Fig. 15 (b) shows the orange refined path and the magenta rough path, and the two paths are almost overlapping. Fig. 15 (c) shows the magenta curvature profile of the rough path trembles violently. The curvature profile is smooth enough when the rough path is optimized. The host vehicle detects a static obstacle ahead when $t = 24.2s$, $s = 200m$ as shown in Fig. 16. The result of the speed optimization step shows that the speed of the ego vehicle is $15.1m/s$ at $t = 24.2s$, then the

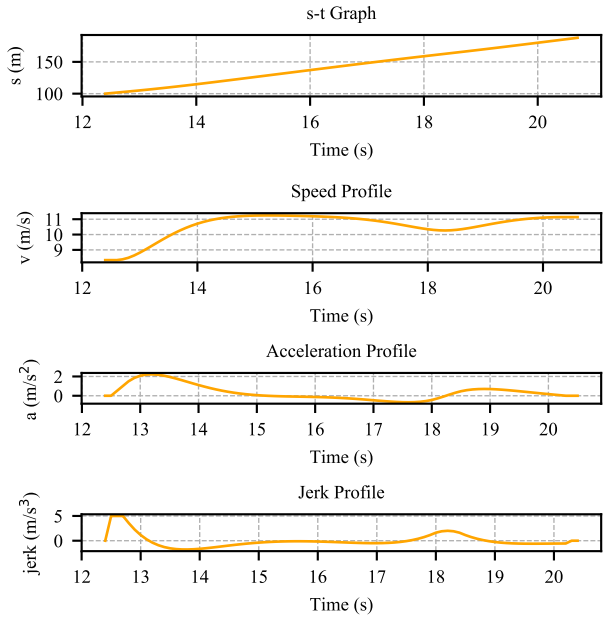


FIGURE 13. Speed profile, acceleration profile and jerk profile generated by the speed profile optimizer in the scenario of lane changing.

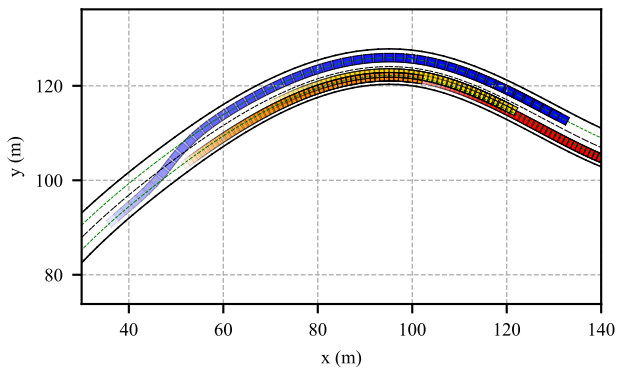


FIGURE 14. Trajectories of the host vehicle and surrounding vehicles in the scenario of lane changing.

speed is reduced to $0.0m/s$ at $t = 32.0s$. The maximum deceleration is $-2.5m/s^2$ and the maximum derivative of the deceleration is $-5m/s^3$. Fig. 17 shows the trajectories of the host vehicle and surrounding vehicles. The blue trajectory shows that the host vehicle is slowing down. The results show that of the trajectory planner can output a smooth and feasible trajectory while satisfying the kinodynamic constraints and traffic rule constraints.

D. CASE4: OVERTAKING

One of the most challenging scenarios for trajectory planning is to overtake the dynamic vehicles ahead. Due to the high speed of the ego vehicle and surrounding vehicles, the trajectory given by the planner requires reasonable and timely acceleration and steering. When $s = 400.0m$, $t = 39.2s$, the decision-making layer issues an overtaking instruction.

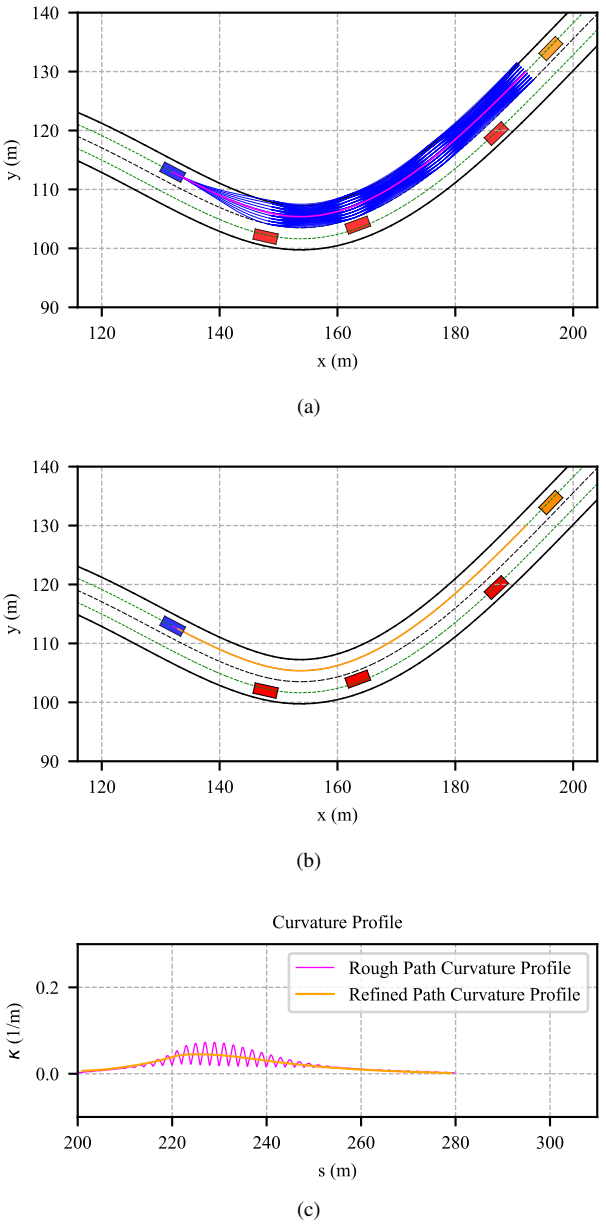


FIGURE 15. The stopping scenario. (a) The blue lattice is constructed in the current lane. The magenta curve represents the rough path. The blue box and the remaining boxes represent the ego vehicle, static and dynamic obstacles respectively. (b) Path optimization. The magenta curve represents the rough path. The orange curve represents the refined path. The green dotted curves represent the reference line for each lane respectively. (c) Curvature profile of path optimization.

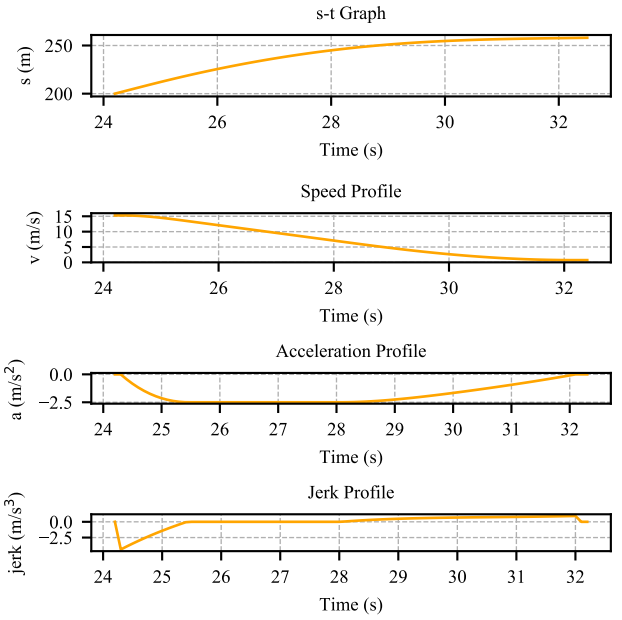


FIGURE 16. Speed profile, acceleration profile and jerk profile generated by the speed profile optimizer in the scenario of stopping.

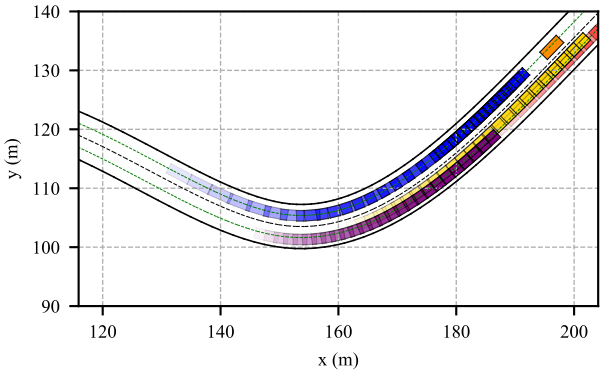


FIGURE 17. The ego vehicle trajectory of stopping.

Fig. 18 (a) displays the rough path can avoid obstacles. Fig. 18 (b) shows the magenta rough path and the orange refined path on the road. Fig. 18 (c) shows that the curvature profile of the rough path has three step changes, which indicates that the curvature of the rough path generated in the path lattice is discontinuous and not smooth. According to our experience, step changes usually occur at the junctions of the motion primitives. During the overtaking process, there are two slowly moving vehicles in the current lane, and there is a moving vehicle ahead in the left lane. Fig. 19 shows that the planning speed profile should be increased from $6.0m/s$ at $t = 39.2s$ to $15.1m/s$ at $t = 44.3s$. After passing the right vehicles, the ego vehicle changes to the right lane and starts to slow down slowly. Because the reference speed we set in this paper is $40.0km/h$, the speed of the host vehicle returns to the reference speed at $t = 50.0s$. Fig. 19 shows that the speed profile, the acceleration profile and the jerk profile generated by the speed optimizer are very smooth. Fig. 20 records trajectories of the host vehicle and surrounding vehicles in the overtaking scenario. The blue trajectory indicates the movement of the host vehicle. As can be seen from Fig. 20, the interval between the blue boxes which belongs to host vehicle's trajectory is getting larger after switching to the left lane, which indicates that the vehicle is accelerating. And after changing to the right lane, the interval of the boxes is getting smaller and smaller, which indicates that the vehicle is slowing down. Fig. 20 shows that our planning results are

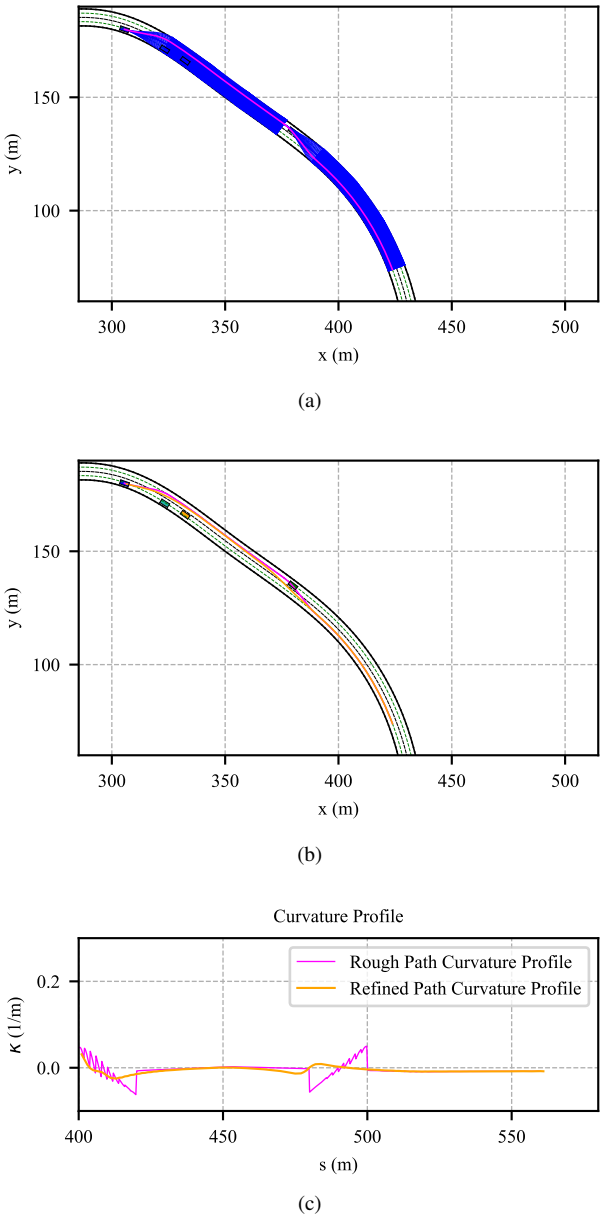


FIGURE 18. The overtaking scenario. (a) The blue lattice is constructed in two lanes. The magenta curve represents the rough path. The blue box and the remaining boxes represent the ego vehicle and dynamic obstacles respectively. (b) Path optimization. The magenta curve represents the rough path. The orange curve represents the refined path. The green dotted curves represent the reference line for each lane respectively. (c) Curvature profile of path optimization.

similar to the real on-road overtaking process.

E. PERFORMANCE ANALYSIS

We tested the running time of the path searcher and the speed profile searcher. For a path lattice with 5 sets of longitudinal sampling waypoints with an interval of $20m$ and 9 sets of lateral sampling waypoints with an interval of $0.5m$, the average search time is $0.12s$. For a speed profile lattice with 8 sets of temporal sampling nodes with an interval of $1.0s$ and 5 sets of station sampling nodes with an interval of $4.0m$, the

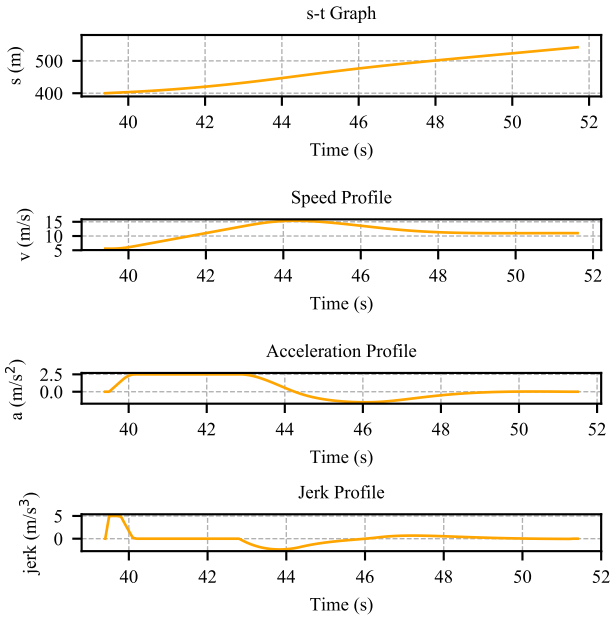


FIGURE 19. Speed profile, acceleration profile and jerk profile generated by the speed profile optimizer in the scenario of overtaking.

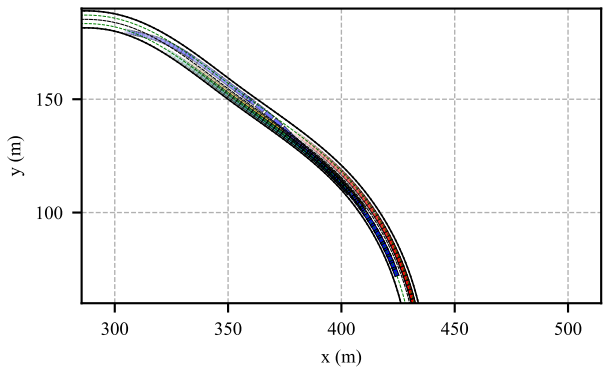


FIGURE 20. The host vehicle trajectory of overtaking.

average search time is $0.04s$. For a path optimization problem with three surrounding obstacles with a path length of $100m$, we convert it into a nonlinear optimization problem with 100 decision variables and about 2200 nonlinear constraint functions. To solve this medium scale optimization problem in this test, we tested the SQP method, interior point (IP) method, SQP-legacy method, and active-set method separately. The comparison results are shown in Table 2. The inte-

TABLE 2. Runtime of Different Solvers in the Path Optimization Step

Algorithm	Runtime (s)	Number of Iterations
SQP	4.5036	48
SQP-LEGACY	3.2448	48
ACTIVE-SET	21.0768	58
IP	0.0982	12

TABLE 3. Performance Comparison of the mainstream trajectory planning methods

Algorithm	Iteration	Time (s)	Planning Horizon	Speed Planning	Planning Strategy
Ours	7(path)+6(speed)	0.55	Long	✓	Path-speed decoupling, separate searching and optimization for path and speed
Ziegler et al.[2]	20	0.89	Long	✓	Direct optimization via waypoint coordinate optimization
Werling et al.[1]		0.11	Short	✓	Lateral trajectory - longitudinal trajectory decoupling, semi-reactive trajectory generation
Li et al.[16][17]	4	0.09	Short	✓	Path-speed decoupling, separate spiral path and trapezoidal velocity profile generation
Hybrid A* [23]		0.27	Long	×	Only path planning in grid map
Hybrid A*+CG[23]	7	0.42	Long	×	Path planning in grid map and path smoothing via conjugate gradient

rior point method is picked considering runtime performance.

In order to verify the efficiency of the trajectory planning framework, we tested the runtime performance of the proposed algorithm and other state-of-the-art algorithms in the static obstacles avoidance scenario. In the comparison, we place four static vehicle obstacles in the structured environments, and we set the planning distance horizon forward to be $100m$. The comparison results are shown in Table 3. Our contributions are explained through simulations and comparisons as follows:

- Through verifications of the previous four cases, and comparisons in Table 1 and Table 3, we can show that our algorithm performs well in the characteristics listed in Table 1 and Table 3. Compared with the decoupled method in [1], our algorithm is complete and can develop the full potential of the host vehicle. For the method in [16-17], the running time is a huge attraction. However, the short planning horizon, not smooth enough trajectory, and the limited motion potential of vehicles in discrete state space, these three drawbacks can cause tracking difficulties and bring about potential dangers for autonomous vehicles in high-speed driving. In practice, the planning distance of this method is not $100m$, but $20m$ (maybe $10m$ or $30m$, it depends on the current speed of the host vehicle and the road environments), so this nearsightedness limits the application of the algorithm in high-speed dynamic scenes. For Hybrid A* and the subsequent path smoothing via conjugate gradient [23], this method can generate a smooth path in the crowded environment, but it lacks the speed planning step. At the same time, it is not suitable for a vehicle to drive on structured roads because the temporary goal waypoint changes frequent and is difficult to determine while driving online. On the contrary, Hybrid A* is more suitable for path planning in free space. Impressively, our algorithm does take advantages of sampling-based methods and optimization-based methods, and tries to reduce the side effects of their defects. And our novel and all-sided algorithm is well balanced in terms of various constraints compliance and requirements satisfaction.
- In comparison with the direct method in [2], our algorithm can jump out of local optimum and has better flexibility to adapt to different scenarios by adjusting parameters. And fewer iterations make our algorithm converge to the optimal solution faster. And the running time is also less than the method in [2], which can be seen in Table 3. In Fig. 9, Fig. 12, Fig. 15, and Fig. 18, the optimized paths are able to avoid obstacles and their curvature profiles are spatially smooth enough. And simulations in Section IV and comparisons in Table 1 and Table 3 confirm that the path optimization over lateral offset is efficient and real-time.
- For the speed profile optimization problem with two surrounding obstacles in Station-Time domain with a time horizon $T = 8.0s$, we convert it into a standard quadratic planning problem with 80 decision variables and nearly 1000 linear constraint functions. The average number of iterations is 6 and the running time is $0.0352s$. The QP speed optimization step has excellent performance in real-time and flexibility.

V. CONCLUSION AND FUTURE WORK

In this paper, a novel decoupled trajectory planning framework is proposed and implemented in Python scripts to solve the non-convex spatiotemporal planning problem. The path searcher and the speed profile searcher can guarantee that we can search for a solution close to the globally optimal solution in the non-convex 3D state space. And the subsequent nonlinear path and speed profile optimization processes ensure that our solution is continuous, spatiotemporally smooth and optimal. And the decoupling of the spatial and temporal information also makes our solution converge to the global optimum more efficiently.

The solution the lattice searching generated is in the neighborhood of the globally optimal solution, so the rough path and the rough speed profile are used as the initial solutions of path optimization and speed optimization, respectively. This treatment reduces the number of iterations, improves the speed of convergence, and generates a globally optimal, continuous solution. Cases studies show that the framework is effective in several structured environments. For the curved roads driving, static obstacles and dynamic obstacles avoidance, lane changing, and overtaking, the trajectory planner performs well in these scenarios. So this framework is suitable for autonomous vehicles traveling online on dynamic structured roads and is able to respond accurately to the commands of the decision maker through parameter tuning.

For future work, more cases need to be done to verify the framework's performance. And the runtime is tempting if we convert these scripts into C++. We will transplant the

framework into the trajectory planning module in Autoware[4] for on-road experiments.

APPENDIX. TRANSFORMATIONS FROM SPEED PROFILE OPTIMIZATION TO STANDARD QUADRATIC PROGRAMMING

A. STANDARDIZATION OF THE OBJECTIVE FUNCTION FOR QUADRATIC PROGRAMMING

The speed cost item can be converted into the following form:

$$\sum_{i=1}^{N_t}(v_i-v_{ri})^2=\frac{1}{\varepsilon^2}\mathbf{s}_{N_t+1}^T\mathbf{H}_{vel}\mathbf{s}_{N_t+1}+\sum_{i=1}^{N_t}v_{ri}^2-\frac{2}{\varepsilon}\mathbf{q}_{vel}^T\mathbf{s}_{N_t+1} \tag{43}$$

where,

$$\begin{aligned}&\mathbf{s}_{N_t+1}=[s_1,s_2,s_3,...,s_{N_t+1}]^T\\&\mathbf{H}_{vel}=\begin{bmatrix}1&-1&0&\cdots&0\\-1&2&\ddots&\ddots&\vdots\\0&\ddots&\ddots&\ddots&0\\\vdots&\ddots&\ddots&2&-1\\0&\cdots&0&-1&1\end{bmatrix}\\&\mathbf{q}_{vel}=[-v_{r1},v_{r1}-v_{r2},\cdots,v_{r(N_t-1)}-v_{rN_t},v_{rN_t}]^T\end{aligned} \tag{44}$$

The acceleration cost item can be converted into the following form:

$$\sum_{i=1}^{N_t}a_i^2=\frac{1}{\varepsilon^4}\mathbf{s}_{N_t+2}^T\mathbf{H}_{acc}\mathbf{s}_{N_t+2} \tag{45}$$

where,

$$\begin{aligned}&\mathbf{s}_{N_t+2}=[s_1,s_2,...,s_{N_t},s_{N_t+1},s_{N_t+2}]^T\\&\mathbf{H}_{acc}=\begin{bmatrix}1&-2&1&0&\cdots&\cdots&0\\-2&5&-4&\ddots&\ddots&\ddots&\vdots\\1&-4&6&\ddots&\ddots&\ddots&\vdots\\0&\ddots&\ddots&\ddots&\ddots&\ddots&0\\\vdots&\ddots&\ddots&\ddots&6&-4&1\\\vdots&\ddots&\ddots&\ddots&-4&5&-2\\0&\cdots&\cdots&0&1&-2&1\end{bmatrix}\end{aligned} \tag{46}$$

The jerk cost item can be converted into the following form:

$$\sum_{i=1}^{N_t}jerk_i^2=\frac{1}{\varepsilon^6}\mathbf{s}_{N_t+3}^T\mathbf{H}_{jerk}\mathbf{s}_{N_t+3} \tag{47}$$

[4]https://www.autoware.org/

where,

$$\begin{aligned}&\mathbf{s}_{N_t+3}=[s_1,s_2,...,s_{N_t},s_{N_t+1},s_{N_t+2},s_{N_t+3}]^T\\&\mathbf{H}_{jerk}=\begin{bmatrix}1&-3&3&-1&0&\cdots&\cdots&\cdots&0\\-3&10&-12&6&\ddots&\ddots&\ddots&\ddots&\vdots\\3&-12&19&-15&\ddots&\ddots&\ddots&\ddots&\vdots\\-1&6&-15&20&\ddots&\ddots&\ddots&\ddots&\vdots\\0&\ddots&\ddots&\ddots&\ddots&\ddots&\ddots&\ddots&0\\\vdots&\ddots&\ddots&\ddots&\ddots&20&-15&6&-1\\\vdots&\ddots&\ddots&\ddots&\ddots&-15&19&-12&3\\\vdots&\ddots&\ddots&\ddots&\ddots&6&-12&10&-3\\0&\cdots&\cdots&\cdots&0&-1&3&-3&1\end{bmatrix}\end{aligned} \tag{48}$$

To be consistent with the dimensions of the matrix in (48), we increase the dimensions of the matrices in (44) and (46). Finally, the objective function can be organized into a standard form using the matrix manipulations as follows:

$$\underset{s_1,s_2,\cdots,s_{N_t+3}}{\arg\min}\quad j(\mathbf{s})=\frac{1}{2}\mathbf{s}^T\mathbf{H}\mathbf{s}+\mathbf{q}^T\mathbf{s}+p \tag{49}$$

where,

$$\begin{aligned}&\mathbf{H}=2\frac{w_{vel}}{\varepsilon}\begin{bmatrix}\mathbf{H}_{vel}&\mathbf{0}&\mathbf{0}\\\mathbf{0}&0&0\\\mathbf{0}&0&0\end{bmatrix}+2\frac{w_{acc}}{\varepsilon^3}\begin{bmatrix}\mathbf{H}_{acc}&\mathbf{0}\\\mathbf{0}&0\end{bmatrix}+2\frac{w_{jerk}}{\varepsilon^5}\mathbf{H}_{jerk}\\&\mathbf{q}=-2w_{vel}\begin{bmatrix}\mathbf{q}_{vel}^T&0&0\end{bmatrix}^T\\&p=w_{vel}\varepsilon\sum_{i=1}^{N_t}v_{ri}^2\\&\mathbf{s}=[s_1,s_2,\cdots,s_{N_t+3}]^T\end{aligned} \tag{50}$$

B. STANDARDIZATION OF CONSTRAINT FUNCTIONS FOR QUADRATIC PROGRAMMING

Acceleration constraints can be expressed as:

$$\begin{aligned}\mathbf{G}_{acc}\mathbf{s}_{N_t+3}&\le\mathbf{h}_{acc}\\-\mathbf{G}_{acc}\mathbf{s}_{N_t+3}&\le\mathbf{h}_{acc}\end{aligned} \tag{51}$$

where,

$$\begin{aligned}&\mathbf{G}_{acc}=\begin{bmatrix}1&-2&1&0&\cdots&0&0\\0&1&-2&1&\ddots&\ddots&\vdots\\\vdots&\ddots&\ddots&\ddots&\ddots&0&0\\0&\cdots&0&1&-2&1&0\end{bmatrix}\\&\mathbf{h}_{acc}=[\varepsilon^2a_{\lim},\cdots,\varepsilon^2a_{\lim}]^T\end{aligned} \tag{52}$$

Jerk constraints can be expressed as:

$$\begin{aligned}\mathbf{G}_{jerk}\mathbf{s}_{N_t+3}&\le\mathbf{h}_{jerk}\\-\mathbf{G}_{jerk}\mathbf{s}_{N_t+3}&\le\mathbf{h}_{jerk}\end{aligned} \tag{53}$$

where,

$$\begin{aligned}&\mathbf{G}_{jerk}=\begin{bmatrix}-1&3&-3&1&&&\\&-1&3&-3&1&&\\&&\ddots&\ddots&\ddots&\ddots&\\&&&-1&3&-3&1\end{bmatrix}\\&\mathbf{h}_{jerk}=\begin{bmatrix}\varepsilon^3jerk_{\lim}&\cdots&\varepsilon^3jerk_{\lim}\end{bmatrix}^T\end{aligned} \tag{54}$$

Lane speed limits can be expressed as:

$$\mathbf{G}_{vel}\mathbf{s}_{N_t+3} \leq \mathbf{h}_{vel} \tag{55}$$

where,

$$\mathbf{G}_{vel} = \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 & 0 & 0 \end{bmatrix} \tag{56}$$
$$\mathbf{h}_{vel} = [\varepsilon v_{\text{lim}}, \cdots, \varepsilon v_{\text{lim}}]^T$$

For each decision variable s_i, there are upper and lower bounds, which can be expressed as:

$$\begin{aligned} \mathbf{G}_{\max}\mathbf{s}_{N_t+3} &\leq \mathbf{h}_{\max} \\ -\mathbf{G}_0\mathbf{s}_{N_t+3} &\leq \mathbf{h}_0 \end{aligned} \tag{57}$$

where,

$$\begin{aligned} \mathbf{G}_{\max} &= \mathbf{G}_0 = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \mathbf{h}_{\max} &= [s_{\max}, \cdots, s_{\max}]^T \\ \mathbf{h}_0 &= [s_0, \cdots, s_0]^T \end{aligned} \tag{58}$$

For the obstacle avoidance constraints (38) and (39), we can simplify them into this expression as follows:

$$\begin{aligned} s_{j,i} - s_{3i} &> r_{veh} + r_{dyn} \\ s_i - s_{j,i} &> r_{veh} + r_{dyn} \end{aligned} \tag{59}$$

And (59) can be express as:

$$\begin{aligned} \mathbf{I}_{N_t+3}\mathbf{s}_{N_t+3} &< \mathbf{o}_j \\ -\mathbf{I}_{N_t+3}\mathbf{s}_{N_t+3} &< \mathbf{d}_j \end{aligned} \tag{60}$$

where,

$$\begin{aligned} \mathbf{o}_j &= \begin{bmatrix} s_{j,1} - r_{veh} - r_{dyn} - 2d \\ \vdots \\ s_{j,N_t+3} - r_{veh} - r_{dyn} - 2d \end{bmatrix} \\ \mathbf{d}_j &= \begin{bmatrix} -s_{j,1} - r_{veh} - r_{dyn} \\ \vdots \\ -s_{j,N_t+3} - r_{veh} - r_{dyn} \end{bmatrix} \end{aligned} \tag{61}$$

where j represents the jth obstacle. The number of obstacles is n_o, so (61) can be stacked into the following expression.

$$\mathbf{G}_{obs}\mathbf{s} < \mathbf{h}_{obs} \tag{62}$$

where,

$$\begin{aligned} \mathbf{G}_{obs} &= [\mathbf{I}, -\mathbf{I}, \cdots, \mathbf{I}, -\mathbf{I}]^T \\ \mathbf{h}_{obs} &= [\mathbf{o}_1, \mathbf{d}_1, \cdots, \mathbf{o}_j, \mathbf{d}_j, \cdots, \mathbf{o}_{n_o}, \mathbf{d}_{n_o}]^T \end{aligned} \tag{63}$$

Boundary value conditions (41) can be expressed as:

$$\mathbf{A}\mathbf{s} = \mathbf{b} \tag{64}$$

where,

$$\mathbf{A} = \begin{bmatrix} 1 & & & 0 & \cdots & 0 & & & & \\ & 1 & & 0 & \cdots & 0 & & & & \\ & & 1 & 0 & \cdots & 0 & & & & \\ & & & 0 & \cdots & 0 & 1 & -2 & 1 & 0 & 0 \\ & & & 0 & \cdots & 0 & 2 & -3 & 0 & 1 & 0 \\ & & & 0 & \cdots & 0 & 3 & -4 & 0 & 0 & 1 \end{bmatrix} \tag{65}$$
$$\mathbf{b} = \begin{bmatrix} \varepsilon v_0 + s_0 \\ a_0\varepsilon^2 + s_0 + 2\varepsilon v_0 \\ jerk_0\varepsilon^3 + 3\varepsilon v_0 + 3a_0\varepsilon^2 + s_0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

All constraint matrices can be stacked into a matrix in the form of columns as follows:

$$\begin{aligned} \mathbf{G} &= \left[\mathbf{G}_{acc}, -\mathbf{G}_{acc}, \mathbf{G}_{jerk}, -\mathbf{G}_{jerk}, \mathbf{G}_{vel}, \mathbf{G}_{\max}, \mathbf{G}_0, \mathbf{G}_{obs}\right]^T \\ \mathbf{h} &= \left[\mathbf{h}_{acc}, \mathbf{h}_{acc}, \mathbf{h}_{jerk}, \mathbf{h}_{jerk}, \mathbf{h}_{vel}, \mathbf{h}_{\max}, \mathbf{h}_0, \mathbf{h}_{obs}\right]^T \end{aligned} \tag{66}$$

(42) is the standard form of QP, which is based on the above transformations. After the objective function and the constraint functions are matrixed, the QP problem can be solved quickly by CVXOPT.

REFERENCES

[1] Werling, Moritz, et al. "Optimal Trajectories for Time-Critical Street Scenarios Using Discretized Terminal Manifolds." The International Journal of Robotics Research, vol. 31, no. 3, Mar. 2012, pp. 346–359, DOI:10.1177/0278364911423042.
[2] J. Ziegler, P. Bender, T. Dang and C. Stiller, "Trajectory planning for Bertha — A local, continuous method," 2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, 2014, pp. 450-457, DOI: 10.1109/IVS.2014.6856581.
[3] J. Ziegler et al., "Making Bertha Drive—An Autonomous Journey on a Historic Route," in IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 2, pp. 8-20, Summer 2014, DOI: 10.1109/MITS.2014.2306552.
[4] W. Lim, S. Lee, M. Sunwoo and K. Jo, "Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 2, pp. 613-626, Feb. 2018, DOI: 10.1109/TITS.2017.2756099.
[5] Liu, Changliu, Chung-Yen Lin, and Masayoshi Tomizuka. "The convex feasible set algorithm for real time optimization in motion planning." SIAM Journal on Control and Optimization56.4 (2018): 2712-2733, DOI: 10.1137/16M1091460.
[6] C. Liu, C. Lin, Y. Wang and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," 2017 American Control Conference (ACC), Seattle, WA, 2017, pp. 4177-4182, DOI: 10.23919/ACC.2017.7963597.
[7] J. Chen, W. Zhan and M. Tomizuka, "Constrained iterative LQR for on-road autonomous driving motion planning," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, 2017, pp. 1-7, DOI: 10.1109/ITSC.2017.8317745.
[8] J. Chen, W. Zhan and M. Tomizuka, "Autonomous Driving Motion Planning With Constrained Iterative LQR," in IEEE Transactions on Intelligent Vehicles, vol. 4, no. 2, pp. 244-254, June 2019, DOI: 10.1109/TIV.2019.2904385.
[9] Howard, Thomas M., et al. "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments." Journal of Field Robotics 25.6-7 (2008): 325-345, DOI:10.1002/rob.20244.
[10] Howard, Thomas M., and Alonzo Kelly. "Optimal rough terrain trajectory generation for wheeled mobile robots." The International Journal of Robotics Research 26.2 (2007): 141-166, DOI: 10.1177/0278364906075328.

[11] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 1879-1884, DOI: 10.1109/IROS.2009.5354448.

[12] M. McNaughton, C. Urmson, J. M. Dolan and J. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 4889-4895, DOI: 10.1109/ICRA.2011.5980223.

[13] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135-1145, April 2016, DOI: 10.1109/TITS.2015.2498841.

[14] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli and J. P. How, "Real-Time Motion Planning With Applications to Autonomous Urban Driving," in IEEE Transactions on Control Systems Technology, vol. 17, no. 5, pp. 1105-1118, Sept. 2009, DOI: 10.1109/TCST.2008.2012116.

[15] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma and N. Zheng, "Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving," in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 4, pp. 1961-1976, Aug. 2015, DOI: 10.1109/TITS.2015.2389215.

[16] X. Li, Z. Sun, D. Cao, Z. He and Q. Zhu, "Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications," in IEEE/ASME Transactions on Mechatronics, vol. 21, no. 2, pp. 740-753, April 2016, DOI: 10.1109/TMECH.2015.2493980.

[17] Li, Xiaohui, et al. "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles." Mechanical Systems and Signal Processing 87 (2017): 118-137, DOI: https://doi.org/10.1016/j.ymssp.2015.10.021.

[18] Wenda Xu, Junqing Wei, J. M. Dolan, Huijing Zhao and Hongbin Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, 2012, pp. 2061-2067, DOI: 10.1109/ICRA.2012.6225063.

[19] Fan, Haoyang, et al. "Baidu apollo em motion planner." arXiv preprint arXiv:1807.08048 (2018).

[20] J. Kim, K. Jo, W. Lim, M. Lee and M. Sunwoo, "Curvilinear-Coordinate-Based Object and Situation Assessment for Highly Automated Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 3, pp. 1559-1575, June 2015, DOI: 10.1109/TITS.2014.2369737.

[21] D. Betaille and R. Toledo-Moreo, "Creating Enhanced Maps for Lane-Level Vehicle Navigation," in IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 4, pp. 786-798, Dec. 2010, DOI: 10.1109/TITS.2010.2050689.

[22] Wang, Hongling, Joseph Kearney, and Kendall Atkinson. "Arc-length parameterized spline curves for real-time simulation." Proc.5th International Conference on Curves and Surfaces. Vol. 387396. 2002.

[23] Dolgov, Dmitri, et al. "Path Planning for Autonomous Vehicles in Unknown Semi-Structured Environments." The International Journal of Robotics Research, vol. 29, no. 5, Apr. 2010, pp. 485–501, doi:10.1177/0278364909359210.

[24] Kelly, Alonzo, and Bryan Nagy. "Reactive nonholonomic trajectory generation via parametric optimal control." The International Journal of Robotics Research 22.7-8 (2003): 583-601, DOI: https://doi.org/10.1177/02783649030227008.

[25] K. Chu, M. Lee and M. Sunwoo, "Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles," in IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1599-1616, Dec. 2012, DOI: 10.1109/TITS.2012.2198214.

[26] Hu, Xuemin, et al. "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles." Mechanical Systems and Signal Processing 100 (2018): 482-500, DOI: http://dx.doi.org/10.1016/j.ymssp.2017.07.019.

[27] Siciliano, Bruno, et al. Robotics: modelling, planning and control. Springer Science & Business Media, 2010.

[28] T. Gu, J. Snider, J. M. Dolan and J. Lee, "Focused Trajectory Planning for autonomous on-road driving," 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, 2013, pp. 547-552, DOI: 10.1109/IVS.2013.6629524.

[29] Lefèvre, Stéphanie, Dizan Vasquez, and Christian Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles." ROBOMECH journal 1.1 (2014): 1, DOI: https://doi.org/10.1186/s40648-014-0001-z.

[30] Ö. Ş. Taş et al., "Making Bertha Cooperate–Team AnnieWAY's Entry to the 2016 Grand Cooperative Driving Challenge," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 4, pp. 1262-1276, April 2018, DOI: 10.1109/TITS.2017.2749974.

[31] C. Liu, W. Zhan and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 154-159, DOI: 10.1109/IVS.2017.7995713.

[32] Y. Zhang et al., "Speed Planning for Autonomous Driving via Convex Optimization," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 1089-1094, DOI: 10.1109/ITSC.2018.8569414.

[33] M. Werling, J. Ziegler, S. Kammel and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, 2010, pp. 987-993, DOI: 10.1109/ROBOT.2010.5509799.

[34] Wei, Yuguang, et al. "Dynamic programming-based multi-vehicle longitudinal trajectory optimization with simplified car following models." Transportation research part B: methodological 106 (2017): 102-129, DOI: https://doi.org/10.1016/j.trb.2017.10.012.

[35] W. Zhan, J. Chen, C. Chan, C. Liu and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 632-639, DOI: 10.1109/IVS.2017.7995789.

[36] J. Kim and D. Kum, "Collision Risk Assessment Algorithm via Lane-Based Probabilistic Motion Prediction of Surrounding Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 9, pp. 2965-2976, Sept. 2018, DOI: 10.1109/TITS.2017.2768318.

[37] Katrakazas, Christos, et al. "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions." Transportation Research Part C: Emerging Technologies 60 (2015): 416-442, DOI: https://doi.org/10.1016/j.trc.2015.09.011.

[38] J. Hillenbrand, A. M. Spieker and K. Kroschel, "A Multilevel Collision Mitigation Approach—Its Situation Assessment, Decision Making, and Performance Tradeoffs," in IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 4, pp. 528-540, Dec. 2006, DOI: 10.1109/TITS.2006.883115.

[39] Tianyu Gu, J. Atwood, Chiyu Dong, J. M. Dolan and Jin-Woo Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 250-256, DOI: 10.1109/IROS.2015.7353382.

[40] C. Liu, C. Lin, S. Shiraishi and M. Tomizuka, "Distributed Conflict Resolution for Connected Autonomous Vehicles," in IEEE Transactions on Intelligent Vehicles, vol. 3, no. 1, pp. 18-29, March 2018, DOI: 10.1109/TIV.2017.2788209.

[41] Urmson, Chris, et al. "Autonomous driving in urban environments: Boss and the urban challenge." Journal of Field Robotics 25.8 (2008): 425-466, DOI: https://doi.org/10.1002/rob.20255.

[42] Montemerlo, Michael, et al. "Junior: The stanford entry in the urban challenge." Journal of field Robotics 25.9 (2008): 569-597, DOI: https://doi.org/10.1002/rob.20258.

[43] Hubmann, Constantin, et al. "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles." 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2017, DOI: 10.1109/IVS.2017.7995949.

[44] Ji, Jie, et al. "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints," in IEEE Transactions on Vehicular Technology, vol. 66, no. 2, pp. 952-964, Feb. 2017, DOI: 10.1109/TVT.2016.2555853.

[45] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," in IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, March 2016, DOI: 10.1109/TIV.2016.2578706.

[46] Andersson, Joel AE, et al. "CasADi: a software framework for nonlinear optimization and optimal control." Mathematical Programming Computation 11.1 (2019): 1-36, DOI: https://doi.org/10.5281/zenodo.1257968.

[47] Wächter, Andreas, and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." Mathematical programming 106.1 (2006): 25-57, DOI: https://doi.org/10.1007/s10107-004-0559-y.