2022 IEEE Intelligent Vehicles Symposium (IV)
June 5-9, 2022. Aachen, Germany

Robust Online Path Planning for Autonomous Vehicle Using Sequential Quadratic Programming

Yuncheng Jiang[1], Zenghui Liu[1], Danjian Qian, Hao Zuo, Weiliang He and Jun Wang

***Abstract*— In urban driving scenarios, it is a key component for autonomous vehicles to generate a smooth, kinodynamically feasible, and collision-free path. We present an optimization-based path planning method for autonomous vehicles navigating in cluttered environment, e.g., roads partially blocked by static or moving obstacles. Our method first computes a collision-free reference line using quadratic programming(QP), and then using the reference line as initial guess to generate a smooth and feasible path by iterative optimization using sequential quadratic programming(SQP). It works within a fractions of a second, thus permitting efficient regeneration.**

I. INTRODUCTION

In urban driving scenarios, autonomous vehicles should deal with static or moving obstacles that partially block the roads by generating a collision-free path. Besides, path planning is required to not only consider collision avoidance, but also path feasibility, i.e. path is comfortable, kinetically and dynamically feasible. Such requirement is a great challenge in urban scenarios where autonomous vehicles need to continuously wiggle through narrow passages. Specifically, path planning should satisfy three important requirements: first, the path should be collision-free; second, the path should be comfortable, which means it is smooth and satisfies vehicle kinematic and dynamic constraints; third, the path planning algorithm should be efficient enough to deal with continuously changing urban environments.

To address this, we propose a robust online path planning method. Firstly, we transform the planning frame from Cartesian to Frenet frame. By doing so, we can easily utilize the road geometry properties: the road geometric constraints can be easily defined. Secondly, the path planning problem is decoupled into two phases: in the first phase, a collision-free and smooth reference line is generated using QP, and it is used as initial guess for the next phase in which path is optimized iteratively; in the second phase, the path planning problem is modeled as a non-linear and convex problem, and is solved by SQP. In the first phase, we use piecewise polynomials in equidistance to represent the reference line. It consists of a sequence of quintic polynomials. The objective of reference line generation is to minimize lateral offset w.r.t road center line, and its first and second-order derivatives. Meanwhile, the reference line should be confined within the road boundaries, and its first and second-order derivatives be within vehicle kinetic and dynamic limitation. The link between each polynomial segment should also be guaranteed to be at least $\mathcal{C}^3$ continuous. The reference line generation problem is formulated as a standard QP and can be solved efficiently using commercial software. The reason we compute a reference line first is that with an high-quality initial state, the number of iteration of SQP can be greatly reduced, and in most cases, it can converge to global optima or a local optima that is very close to the global optima.

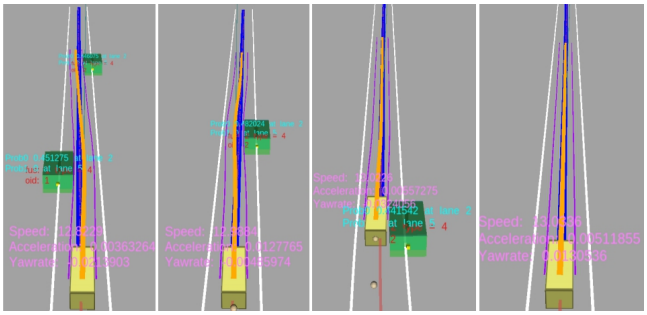

Fig. 1. A typical scenario: The yellow bounding box is ego vehicle, and the other two green ones are static obstacles. The blue line represents road center line, the orange line is planning results, and the two purple lines are the collision-free checking boundaries discussed in IV-C. Ego vehicle avoids approaching vehicles by temporary in-lane nurging.

In the second phase, we still use piecewise polynomials to ensure smoothness of the path. Moreover, the lengths of each segment are also optimization variables. To deal with collision avoidance requirement, we propose a driving corridor generation method which convert the dense road boundries into a series of sparse and inflated driving corridors, and only the link points of each two segments are constrained within the driving corridors. This method also does lazy collision checking after path generation and update driving corridor(s) by adding extra corridor constraints on those areas where collision-free checking fail. Newly added driving corridor pull the path closer to the safe reference line in those dangerous areas. The objective of pathing planning is still to minimize the offsets w.r.t reference line and its first and second-order derivatives, but with variable path segment length, the objective function is not quadratic but still convex. The path generation problem therefore becomes non-linear optimization. To improve computational efficiency, the optimization starts with an initial guess of high quality(reference line) from the first phase, and SQP is used to solve it iteratively, which uses a modified line search method, and the Hessian of Lagrangian is approximated to ensure that it is positive definite as proposed in [1].

[1]Authors who contribute equally to this paper.

All Authors are with Shanghai Automotive Industry Corporation, Ltd. 201, Anyan Road, Jiading, Shanghai, China. Corresponding author: Yuncheng Jiang jiangyuncheng@saicmotor.com

2022 IEEE Intelligent Vehicles Symposium (IV) | 978-1-6654-8821-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IV51971.2022.9827017

II. RELATED WORK

In path planning, the algorithms can be generally categorized into three classes:search-based method, sampling-based method, and optimization-based method. In this paper, the main focus is on optimization-based method.

Optimization-based methods are the most flexible methods. The work in [2] formulates the path planning problem into quadratic programming under global frame. Specifically, path and speed profiles are computed and optimized at the same time. Similar work is done in [3], where trajectory planning is performed under Frenet frame(defined w.r.t a smooth road center line). In this method, longitudinal and lateral motion planning are decoupled. Optimzition is performed on each axis, and then optimal motions are combined and transformed from Frenet frame to global frame.

In urban scenarios, a single polynomial may not be flexible enough to cope with complex road conditions[4][5], and piecewise polynomials can be a better choice. The minimum snap trajectory optimization is first proposed in [6], which is used in quadrotor trajectory planning. In this method, piecewise polynomials are first used to represent optimal path. But the main problem is that this method lacks time allocation optimization. Similar work is done in [7] where a close-form solution is proposed to solve piecewise polynomial problem. Besides, a safe geometric path is firstly generated as reference line to guide the generation of path. Further research is explored in [8], where piecewise polynomial path planning problem is formulated as quadratic programming under Frenet frame. In this work, a dynamic programming(DP) is first used to generate a safe geometric path as reference line for path optimization. Inspired by [3], the authors further develop their work in [9] and [10], where path planning and speed planning are separately optimized and combined. In [9], discrete points are used to replace piecewise polynomials such that the path is more controllable. In this method, a gradient-based method is used to generate a safe reference line. In [10], discrete points are again used to represent the optimal speed profile. The most obvious problem in [8][9][10] is that redundant and conservative constraints are applied (e.g. in path planning, high-resolution and redundant road boundary constraints are applied in optimization, greatly increasing the number of constraints, and accordingly computation time).

In the field of quadrotor, flight(driving) corridor generation has long been explored. [6] generates sparse corridor constraints, and add extra corridor constraints at the area where collision checking fails. [11] proposes efficient corridor generation in an octree-based map representation. Similar works are also done in [12][13][14] where corridors consist of convex shapes(e.g. spheres and cubes). In these methods, extrema is checked after optimal polynomial coefficients are found. The algorithm iterates by adding more constraint points to QP until no point along the path violates boundary constraints. A novel path planning method using piecewise Bezier curve is proposed in [15]. Using the convex hull and hodograph property of Bezier curve, the entire path is optimized within its corridor, and is guaranteed to be safe and dynamically feasible.

Our work is inspired by [6] and [11]. In our work, piecewise polynomials are used to represent path. Unlike [6][7], the path planning problem is formulated as non-linear programming, and each polynomial has variable length. It overcomes path length allocation problem(the same as time allocation problem in quadrotor trajectory planning) in piecewise polynomial path optimization, sacrificing a little bit computation time. We also take advantage of corridor generation methods in quadrotor trajectory optimization so that road boundary constraints are converted to sparse but safe driving corridors, by which the number of constraints are greatly reduced.

III. REFERENCE LINE GENERATION

HD map will provide a sequence of discrete points which is usually the center line of road, but it does not provide other important information such as center line curvature, heading, and yaw rate. Reference line is a prerequisite of planning in Frenet frame, as it serves as a bridge between map in Cartesian frame and path planning in Frenet frame. A high-quality reference line not only reduce the wiggling of vehicle lateral velocity and acceleration in path tracking, but also provides a good initial guess for path optimization. Therefore, in the first phase, we generate a piecewise quintic polynomial reference line by formulating a optimization problem(Fig.2).

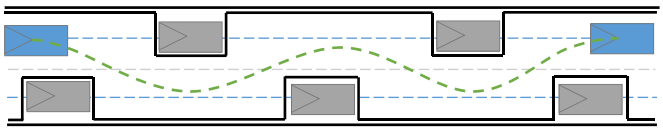

Fig. 2. QP Reference Line: Reference line that is near the road center line is generated, and serves as inital guess for the next path optimization. Since the reference line is already feasible, it enhances the robustness of the post-optimization.

A. Objective Function

The objective is to 1) minimize the wiggling of geometrical properties. The less lateral wiggling, the less lateral velocity and acceleration in path tracking control; 2) stay close to center line as much as possible. The vehicle is required to stay in lane, accordingly the reference line should be close to the center line. The objective function, therefore, consists of three terms: lateral offset w.r.t center line, the first and second-order derivative of lateral offset w.r.t spatial parameter.

B. Constraints

The reference line does not need to pass through center line accurately. However the deviations to center line should be limited so that the shape of the reference line does not deviate too much from center line. The reference line should also be confined between road boundaries. For the purpose of simplification, we only apply road boundary constraints on equally-distributed control points of each segment. The

first and second-order derivative w.r.t spatial parameter are also constrained within vehicle kinetic and dynamic limitation. Continuity constraints guarantee that the link of each polynomial segment is at least $\mathcal{C}^3$ continuous.

C. Problem Formulation

The reference line consists of m-segment and n^{th} order piecewise polynomials. We apply "many relative lines"(length of each segment always starts from zero), and the whole reference line are written as:

$$l(s) = \begin{cases} \sum_{j=0}^{n} a_{1j} s^j & 0 \le s \le s_{seg} \\ \sum_{j=0}^{n} a_{2j} s^j & 0 \le s \le s_{seg} \\ \vdots & \\ \sum_{j=0}^{n} a_{mj} s^j & 0 \le s \le s_{seg} \end{cases} \tag{1}$$

where a_{ij} is the j^{th} order polynomial coefficient of the i^{th}, and segement s_{seg} is the length of each segment. The objective function is as follows:

$$\begin{aligned} f\left(l\left(s\right)\right) = w_l \int_0^{s_e} l^2(s) ds + w_{dl} \int_0^{s_e} \left(\frac{d(l(s)}{ds}\right)^2 ds \\ + w_{ddl} \int_0^{s_e} \left(\frac{d^2(l(s))}{ds^2}\right)^2 ds \\ + w_{ref} \int_0^{s_e} (l(s) - l_{ref}(s))^2 ds \end{aligned} \tag{2}$$

where s_e is the total length of the path. When applying constraints, we only choose n equally distributed control points. Therefore the constraints are as follow:

- Inequality constraints of lateral offset:

$$l_{i,j}^{min} \le l_{ij} \le l_{i,j}^{max}$$

- Inequality constraints of the first order derivative:

$$\dot{l}_{i,j}^{min} \le \dot{l}_{ij} \le \dot{l}_{i,j}^{max}$$

- Inequality constraints of the second order derivative:

$$\ddot{l}_{i,j}^{min} \le \ddot{l}_{ij} \le \ddot{l}_{i,j}^{max},$$

- continuity constraints:

$$\frac{\mathrm{d}^k l_{i,n}}{\mathrm{d} {l_{i,n}}^k} = \frac{\mathrm{d}^k l_{i+1,1}}{\mathrm{d} {l_{i+1,1}}^k}$$

where $l_{i,j}$ means the j^{th} point of the i^{th} segment, and k means the k^{th} order derivative, where $i \in [1, 2, \ldots, m]$, $j \in [1, 2, \ldots, n]$, $k \in [1, 2, 3]$.

We formalize the problem into a standard QP problem so that it can be fast solved by quadratic programming commercial software. The expression of each segment is expanded as $l(s_i) = \boldsymbol{a_i s}$, where $\boldsymbol{a_i} = [a_{i0}, a_{i1}, a_{i2}, a_{i3}, a_{i4}, a_{i5}]$ and $\boldsymbol{s} = [1, s, s^2, s^3, s^4, s^5]^T$. The quadratization of the first objective term can be expanded as:

$$\begin{aligned} \int_0^{s_j} l^2(s) ds = \sum_{i=0}^{m} \int_0^{s} (\sum_{j=0}^{n} a_{i,j} s^j)^2 ds \\ = \sum_{i=0}^{m} \int_0^{s} \boldsymbol{a_{i,j}^T s s^T a_{i,j}^T} ds = \boldsymbol{a^T S a} \end{aligned} \tag{3}$$

where $\boldsymbol{a} = (\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_m})^T$ is polynomial coefficient of each segment. $\boldsymbol{S}$ is defined as the block diagonal matrix $diag(\int_0^s \boldsymbol{s s^T} ds, \ldots, \int_0^s \boldsymbol{s s^T} ds)$. Similarly, the rest terms can be quadratized in the same way. The objective function can be written as $\boldsymbol{a^T S a}$. The constraints can be formulated in either equality constraints $\boldsymbol{A}_{eq} a = b_{eq}$ or inequality constraints $\boldsymbol{A}_{eq} a \le b_{ieq}$, and both constraints are affine. The reference line generation problem can be modeled as a standard quadratic programming:

$$\begin{aligned} arg\ min \frac{1}{2} \boldsymbol{a^T S a} + \boldsymbol{f^T a} \\ s.t.\ \boldsymbol{A_{ieq} a \le b_{ieq}} \\ \boldsymbol{A_{eq} a = b_{eq}}, \end{aligned}$$

which can be solved efficiently by commercial software. In reference line generation phase, we do not implement nonlinear optimization. Although introducing some nonlinear terms can make the result better(e.g. curvature constraints), it consumes more computation resources. Since we will, in the second phase, apply SQP to generate an optimal path, the result from QP is good enough as an initial guess for the next optimization.

IV. TRAJECTORY GENERATION WITH CONSTRAINTS

In this section, we discuss generation of an optimal path that satisfies kinetic and dynamic constraints in driving corridors using an efficient SQP approach(IV-A). Piecewise polynomial trajectory generation is initially proposed in Baidu's prior work[8], and we make improvements in implicit segment lengths optimization(IV-B). To impose constraints on the modified nonlinear problem, we propose a driving corridor generation method, which constraints segment link points in inflated driving areas, and consider the shape and heading of ego vehicle. We also iteratively add extra constrain points that pull the path closer to safe reference line when some part of the path violates corridor constraints(IV-C). It is shown that only a finite number of extra constrain points are needed to guarantee that corridor constraints are satisfied[16], and the algorithm can converge robustly. Although the problem is formulated as a non-linear problem, we propose warm-start initialization so that a modified SQP algorithm[1] starts from an initial guess very close to global optima and converges efficiently.

A. Problem Definition

The optimal path consists of m-segment and n^{th} order piecewise polynomials. Notice that the lengths of each path segment are also optimization variables, and the whole path can be written as:

$$l(s) = \begin{cases} \sum_{j=0}^{n} p_{1j} \left(s - s_0\right)^j & s_0 \le s \le s_1 \\ \sum_{j=0}^{n} p_{2j} \left(s - s_1\right)^j & s_1 \le s \le s_2 \\ \vdots & \\ \sum_{j=0}^{n} p_{mj} \left(s - s_{m-1}\right)^j & s_{m-1} \le t \le s_m \end{cases} \tag{4}$$

where p_{ij} is the j^{th} order polynomial coefficient of the i^{th} segement. The optimization problem is modeled to search a

sequence of polynomial segments that minimize the integral of the square of the n^{th} derivatives:

$$\min(\sum_{n_i=0}^{3}\sum_{j=0}^{m}\int_0^{s_j}(\frac{d^{n_i}l(s_j)}{ds^{n_i}})^2ds + \sum_{j=0}^{m}\int_0^{s_j}(l(s_j)-s_{ref})^2ds). \quad (5)$$

where n_i denotes the order of derivative, j denotes the index of polynomial segment. The objective function is no longer quadratic with the variable length of each polynomial segment, but is still convex. We will discuss the details of the objective function formulation in IV-B, and equality and inequality constraints in IV-E.

B. Objective Function

The vehicle path should be smooth, comfortable, and pass through safe areas. To satisfy these requirements, the path should have as less lateral movements as possible, i.e. the first, second, and third order derivatives of lateral offset should be low. It can not only leads to smooth and low path curvature, accordingly reducing instability and overshoot in path tracking, but also improve driving comfort. Meanwhile, the autonomous vehicle is also encourage to stay in lane, which means lateral offset with respect to road center line is penalized. Finally, the vehicle is also encouraged to follow the reference line as close as possible. The objective function therefore, has three terms:

1) **lateral offset**. The lateral offset should be minimized so that the vehicle follow the center line as close as possible.
2) **derivatives of lateral movement**. The first order derivative of lateral movement represents how quickly the vehicle moves laterally, which is very important for path tracking controller. The second and third order derivatives of lateral movement represent the change rate of vehicle lateral velocity and acceleration, which have direct effects on driving comfort and vehicle fuel-economy.
3) **lateral offset with respect to reference line**. The reference line is a smooth and comfortable path, and the vehicle is encouraged to follow the reference line as close as possible, so long as it does not violate safe corridor and vehicle feasibility constraints.

The expression of objective function is shown in Eq. (6). Specifically, the cost of each segment can be expanded as follows:

$$\begin{aligned} f\left(l\left(s_j\right)\right) = {} & w_l\int_0^{s_j} l^2(s)ds + w_{dl}\int_0^{s_j}\left(\frac{d(l(s)}{ds}\right)^2 ds \\ & + w_{ddl}\int_0^{s_j}\left(\frac{d^2(l(s))}{ds^2}\right)^2 ds \\ & + w_{dddl}\int_0^{s_j}\left(\frac{d^3(l(s))}{ds^3}\right)^2 ds \\ & + w_{ref}\int_0^{s_j}(l(s)-\mathrm{l_{ref}}(s))ds \end{aligned} \quad (6)$$

Each segment of path has the same expression as reference line which is defined as $l(s_i) = \boldsymbol{p_i s}$, where $\boldsymbol{p_i} = [p_{i0}, p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5}]$ and $\boldsymbol{s} = [1, s, s^2, s^3, s^4, s^5]^T$. Let us use the first objective function term as an example to show its formulation:

$$\begin{aligned} \int_0^{s_m} l^2(s)ds &= \sum_{i=0}^{m}\int_0^{s_i}(\sum_{j=0}^{n}a_{i,j}s^j)^2ds \\ &= \sum_{i=0}^{m}\int_0^{s_i}\boldsymbol{a_{i,j}^T s s^T a_{i,j}^T}ds = \boldsymbol{a^T S a} \end{aligned} \quad (7)$$

where $\boldsymbol{a} = (\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_m})^T$ is polynomial coefficient of each segment. $\boldsymbol{S}$ is defined as the block diagonal matrix $diag(\int_0^{s_i}\boldsymbol{ss}^T ds, \ldots, \int_0^{s_i}\boldsymbol{ss}^T ds)$. Similarly, the rest terms can be quadratized in the same way. The objective function can be written as $\boldsymbol{a}^T\boldsymbol{S}\boldsymbol{a}$. Notice that although the expression of the first objective function term is very similar to (3), it is not quadratic form due to the variable length s_i of each path segment. However, the objective function is still convex.

C. Driving Corridor Constraints

Assume that the road boundaries are given in the form of a sequence of discrete points $\mathcal{C} = \{c_1, c_2, \ldots, c_m\} \in \mathcal{R}^{3\times m}$, where $c_i = (s_i, c_i^l, c_i^u)^T$. s_i stands for the longitudinal distance in Frenet frame. c_i^l and c_i^u denote lower bound and upper bound. Unlike the path representation in Sec.III where boundary constraints can be directly applied to each path segment, the length of each path segment is optimization variables, and therefore the implementation of boundary constraints on the variant path segments is implicit. More specifically, there is no exact match between path segments and boundary constraints. We propose a driving corridor generation method to realize implementation of boundary constraints. We choose a sequence of control points $\mathcal{J} \in \mathcal{R}^{2\times n}$ along the reference line with equidistance as the initial driving corridors, then each control point is inflated in free-space to provide larger driving corridors. The boundary constraints are converted to control point constraints. By introducing control points, we can implement boundary constraints on specified areas, and therefore the match between path segment and its boundary constraints becomes clear.

Let us assume that the total length of path is S, and there are n control points. $\mathcal{J}$ is expanded as $\mathcal{J} = \{j_1, j_2, \ldots, j_n\}, j_i \in (j_i^s, j_i^l)$, which are equally distributed with distance $S_{aver} = \frac{S}{n-1}$. In the s direction, each control point is inflated forward and backward. Since the inflated areas determines the position of the link point of each path segment, we deliberately limit the longitudinal inflating distance to ensure they are not overlapped. It not only constrains the minimum distance of each path segment, but also guarantees numerical stability of quintic polynomial. Empirically, a very short quintic polynomial can be numerically unstable, and can be greatly affect by little noise. Therefore, the minimum longitudinal distance between each driving corridor is defined as Δs_{min}, and the longitudinal inflating distance s_{lon} must satisfy $s_{lon} \leq \frac{1}{2}(\frac{S}{n-1} - \Delta s_{min})$.

Fig.3 shows the denotation of the variables. The constraints of the length of each segment s_i can be written as follows:

- The length of each segment is limited by driving corridors that have longitudinal minimum intervals:

$$\frac{S}{n-1} - s_{lon} \leq s_i \leq \frac{S}{n-1} + s_{lon}, \tag{8}$$

- Summation constraints that guarantee each link point is constrained in its driving corridor:

$$i\frac{S}{n-1} - s_{lon} \leq \sum_{i=1}^{n} s_i \leq i\frac{S}{n-1} + s_{lon}, \tag{9}$$

- Summation of all segment satisfies:

$$\sum_{i=1}^{n} s_i = S. \tag{10}$$

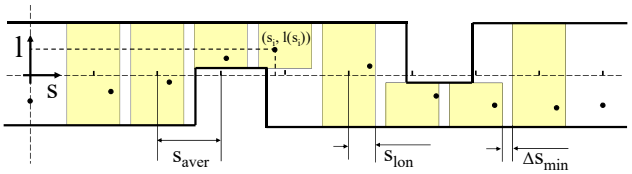

Fig. 3. Driving Corridor Generation: Equally-distributed points are choosen as original control points, driving corridors are generated at each control point by inflating longitudinally and laterally.Link points of each path segment are constrained in respective driving corridors.

To determine the lateral position constraints of each segment, we laterally inflate each driving corridor within its longitudinal limits(Eq.8) until any point reaches its upper bound or lower bound. Since the longitudinal inflating areas of each driving corridor has been determined, it is easy to extract corresponding lower and upper bounds from road boundaries which is defined as:

$$c_{s_i}^l \leq l(s_i) \leq c_{s_i}^u, \forall i = 1, 2, \ldots, m, \tag{11}$$

where $c \in \mathcal{C}$, and $l(s_i)$ is lateral offset at s_i. Combine (8) and (11), driving corridor is defined as a sequence of rectangles $\mathcal{W} = \{w_1, w_2, \ldots, w_m\} \in \mathcal{R}^{4\times m}$, where $w_i = (L_i^{lon}, U_i^{lon}, L_i^{lat}, U_i^{lat})^T$. L stands for lower bound, and U stands for upper bound. The complexity of extracting corresponding lower and upper bounds from road boundaries is $O(\frac{S}{(n-1)\delta s})$, where δs is resolution of road boundaries. The total complexity of implementing road boundaries is $O(\frac{nS}{(n-1)\delta s})$. The algorithm of driving corridor generation is shown in Alg. 1.

To consider the geometry of ego vehicle, a bicycle model is generated at each control point. Ego vehicle is simplified as a rectangle, and two circles with radius $\mathcal{R}_f$ and $\mathcal{R}_r$ are added to the front and rear axle of ego vehicle respectively. A control point constraint is converted to four rectangle vertices constraints shown in Fig.4. Let us denote the wheelbase as l, vehicle width as w, the distance from front bumper to front axle as l_f, and the distance from rear bumper to rear axle as

Algorithm 1: Driving Corridor Generation.

Input: road boundaries $\mathcal{C} \in \mathcal{R}^{3\times n}$; control points $\mathcal{J} \in \mathcal{R}^{2\times m}$; longitudinal inflating distance S_{lon}; equally distributed distance S_{aver}; path length S;

Output: driving corridor $\mathcal{W} \in \mathcal{R}^{4\times m}$;

```
k = 1 ;
while k ≤ m do
    foreach j_k ∈ J do
        j_k^s ∈ [kS_aver − S_lon, kS_aver + S_lon];
        l_k^lon = kS_aver − S_lon, u_k^lon = kS_aver + S_lon;
        foreach s_k^l ∈ j_k do
            if L_k^lat < c^l_{s_k^l} then
                L_k^lat = c^l_{s_k^l} ;
            if U_i^lat > c^u_{s_k^l} then
                U_k^lat = c^u_{s_k^l} ;
            w_k = (L_k^lon, U_k^lon, L_k^lat, U_k^lat)^T;
    k = k + 1;
```

l_r. The lateral position of the left-front corner of ego vehicle is given by:

$$l = f(s) + sin(\theta)l + R_f, \tag{12}$$

where $f(.)$ denotes for the lateral position of the rear axle. To guarantee the linearity of driving corridor constraints, Eq. (12) is simplified as:

$$l = f(s) + f^{'}(s)l + R_f. \tag{13}$$

Similarly, the constraints on the other three corners can be formulated in the same way.

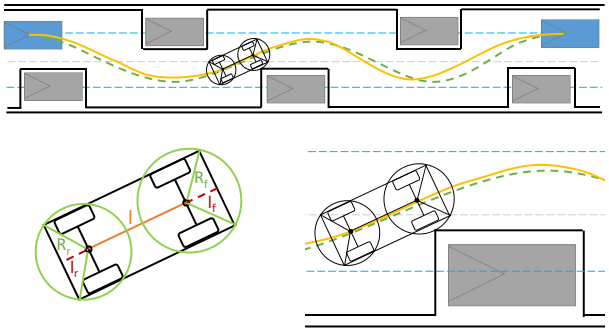

Fig. 4. Vehicle Geometry and Constraints Linearization: Ego vehicle is simplied as a rectangle and two circles. The linearized constraints ensures that inequality constraints are affine. For $\theta \leq \frac{\pi}{12}$, the estimation error is no more than 14 cm.

D. Violation of Driving Corridor Constraints

According to Eq.13, driving corridor constraints are applied to each link points. However, it only guarantees that the vehicle at link points stays within the driving corridor. It does not provide any guarantee on the safety of any

Algorithm 2: Driving Corridor Vialation Check.

Input: road boundaries $\mathcal{C} \in \mathcal{R}^{3\times n}$; control points $\mathcal{J} \in \mathcal{R}^{2\times m}$; longitudinal inflating distance s_{lon}; equally distributed distance S_{aver}; path length S; violation areas $\mathcal{J}_d$;

Output: driving corridor $\mathcal{W} \in \mathcal{R}^{4\times m}$;

foreach $pair(j_i, j_{i+1}) \in \mathcal{J}_d$ **do**
  $j^s_{new} = [0.5(j_i + j_{i+1}) - S_{lon}, 0.5(j_i + j_{i+1}) + S_{lon}]$;
  foreach $s^l_{new} \in j_{new}$ **do**
    if $L^{lat}_{new} < c^l_{s^l_k}$ **then**
      $L^{lat}_{new} = c^l_{s^l_k}$;
    if $U^{lat}_{new} > c^u_{s^l_k}$ **then**
      $U^{lat}_{new} = c^u_{s^l_k}$;
    $w_{new} = (L^{lon}_{new}, U^{lon}_{new}, L^{lat}_{new}, U^{lat}_{new})^T$;
    $\mathcal{W} = \mathcal{W} \cup w_{new}$

other points of the path. To achieve such a guarantee, one simple way is to choose a well-designed resolution of driving corridor constraints which ensures that the position constraints are dense enough to enforce all points throughout the path within the driving corridor shown in Fig.5. The weakness of this method is that we have to apply redundant position constraints on some points that does not need to be constrained, and therefore, the computation time increases with the number of constraints. The worst case is that driving corridor constraints are applied on each point of the path. We propose a novel method algorithm to enforce least driving corridor constraints by only adding extra control point(s) on those areas where path violates corridor constraints shown in Fig.6. Although the updated driving corridor constraints are slightly different from the previous constraints, we can warm start the algorithm using the solution of last loop so that it converges within a few iterations. Meanwhile, we do not need to worry about the initial feasibility problem: although the warm start solution does not satisfy the newly generated driving corridor(s), an interior-point based convex programming method does not require that the initial guess starts from feasible region. Let us assume that some areas violate collision-free checking(e.g. area $\mathcal{D}$ in Fig.6(a), and the violation areas are defined using pairs of control points $\mathcal{J}_d = \{(j_i, j_{i+1}), \dots\}$. Pair (j_i, j_{i+1}) means the violation areas are longitudinally located between control point j_i and j_{i+1}. new control points are generated at the mid points of those areas, and are inflated using the same rules in Sec.IV-C shown in Fig.6c. The algorithm is shown in Alg.2.

E. Equality and Inequality Constraints

Lateral offset constraints are converted to driving corridor constraints shown in Sec.IV-C. Other inequality constraints and continuity constraints are the same as that in reference line generation shown in Sec.III. Namely, we still have the first order derivative constraints, the second order derivative constraints, and continuity constraints.

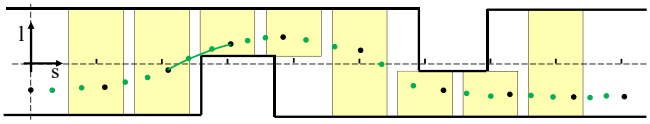

Fig. 5. Increasing Resolution of Constrained Point: The black points are original control points in equidistance. The green points are densely distributed control points to enforce position constraints. If we conservatively apply driving corridor generation on each control points, position constraints are guaranteed by sacrificing computation time.

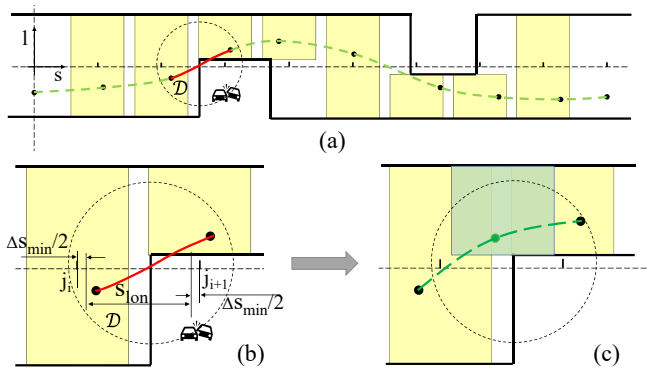

Fig. 6. Add Additional Control Points at Violation Segment(s): The area $\mathcal{D}$ is detected to violate collision-free checking. The planned path is shown in red line. Then a new control point is first generated at the middle of j_i and j_{i+1}, then an extra driving corridor is inflated from that mid points using a conservative way. The green driving corridor is newly generated, and a new path segment is optimized shown in green dashed line.

V. Results

We use an Intel Core i9-10900 2.8GHz PC with 32GB RAM to run simulation on CarMaker platform. The proposed path planning method is implemented using C++. Besides, we use the sparse linear algebra library in Eigen and an open source quadratic programming solver OSQP [17] to solve QP problem in SQP algorithm. In this section, a typical scenario test is proposed to show system performance of our method. Then we make a comparison between our method and another two state-of-the-art methods: Minimum Snap Method(MSM)[6] and Piecewise Jerk Method(PJM)[9].

A. Qualitative Results

In this section, we create a typical lane borrow scenario to qualitatively test the system performance of our method. Ego vehicle starts from 0 m on a straight line, and two static vehicles partially block current lane at longitudinally 25 m, and 55 m. Current lane width is 3.75 m, and the rear axle center of the two static vehicles have offsets of -1.0m and 1.0m respectively w.r.t the road center line. Road boundaries are generated shown in Fig.7(a).

We assume that the planned trajectory is perfectly tracked. The iteration process of our proposed method at $l = 4m$ is presented in Fig.7, which is plotted in Frenet frame. In the first iteration (Fig.7a), driving corridor constraints are applied to link points using Alg.1, which is indicated by orange rectangles. Alg.2 is used to do collision-checking, and the second, third and fifth segments fail in the collision-checking

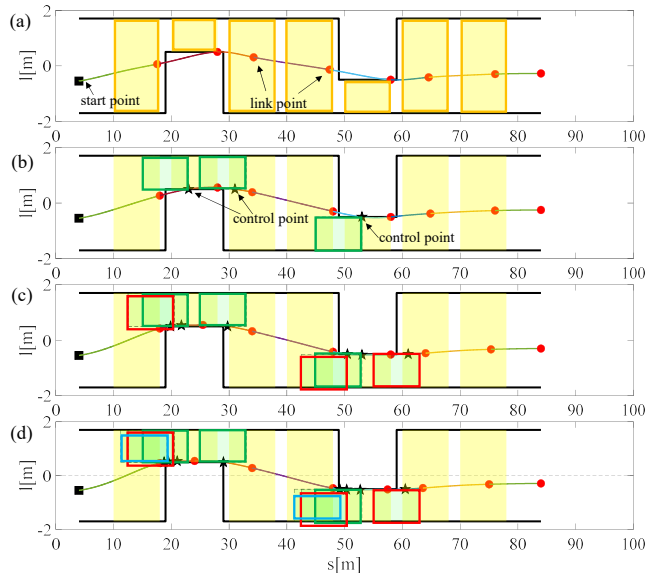

Fig. 7. Corridor Generation and Iterative Optimization: The driving corridors are updated iteratively and a collision-free path is finally obtained. The orange rectangles are initial driving corridors generated at link points. Newly added driving corridors in the second, third and fourth iterations are plotted using green, red and blue rectangles, respectively.

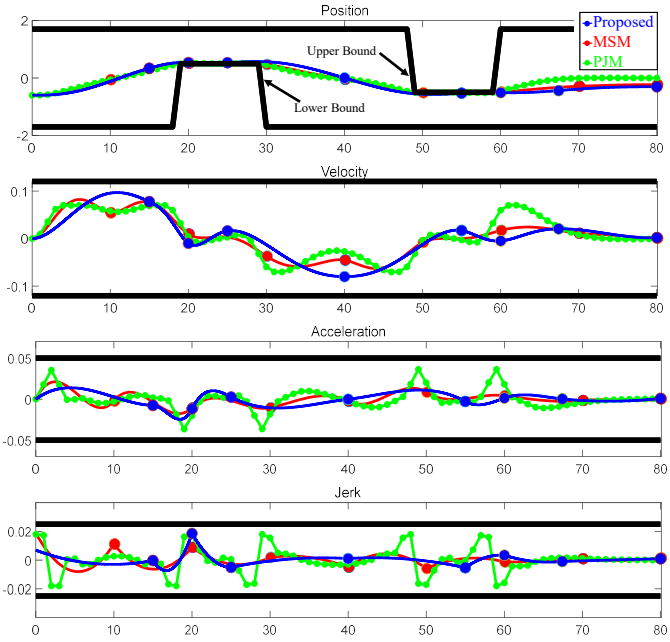

Fig. 8. Comparison Between Our Method and MSM, PJM: The optimized paths, velocity, acceleration, and jerk are plotted. Our proposed method is shown in blue curves, MSM is shown in red curves, and PJM is shown in green curves.

test. Then in the second iteration, extra driving corridors are generated in those segments using Alg.1 again, which is shown in Fig.7(b). Path is regenerated with the newly added constraints and the same process is repeated until the path passes the collision-checking test.

On average, a collision-free path is generated within 2-5 iterations. It is worth noting that in some cases, driving corridor constraints are not completely satisfied, but with enough iterations, collision-free is guaranteed. The reasons are: 1) there are buffers between road boundnaries and real obstacles, which means the path is generated conservatively; 2) after enough iterations(e.g. 5 iterations), the density of driving corridor constraints reaches $\frac{s_{aver}}{5}$, and the resolution is far enough in practice. Actually, the constraints resolution is the same as that in Fig.5, and generated path is guaranteed to be collision-free.

B. Comparison and Analysis

In this section, we use a similar scenario as shown in Fig.1 to compare our method with MSM and PJM. We simplify ego vehicle as a point, and only study the quality of generated paths of each method. The velocity, acceleration, and jerk are also calculated to qualitatively indicate the path quality shown in Fig.8. We calculate the mean and maximum absolute value of lateral offset, velocity, acceleration and jerk of the path in each method, and summarize the data in Table.I. Then we study the parameter sensitivity between our method and MSM.

As shown in Fig.8, the three methods are used to generate paths in the same scenario with the same well-tuned cost function parameters. Although there is little differences in path lateral offset, path velocity, acceleration, and jerk varies greatly. For PJM, the path is represented by a sequence of points. The geometrical information is calculated by finite differentiation between consecutive points. Since this method uses discrete points and only guarantees acceleration continuity, the jerk of the path is not differentiable. In terms of fuel economy, jerk is a key factor to indicate energy consumption, and PJM has the worst jerk quality among the three methods. In terms of velocity and acceleration, PJM still has more vibration and obvious peak values. Our method and MSM have smoother velocity and acceleration.

TABLE I
THE PATH QUALITY VARIFICATION USING COST FUNCTION PARAMETER SET 1

	Piecewise Polynomial	Piecewise Jerk	Proposed
$max(\|v\|)$	0.0209	0.0702	0.0112
$max(\|a\|)$	0.0026	0.0364	0.0017
$max(\|j\|)$	0.0014	0.0180	0.0014
$mean(\|v\|)$	0.0101	0.0339	0.0059
$mean(\|a\|)$	0.0017	0.0077	0.0010
$mean(\|j\|)$	3.6006e-04	0.0059	3.8239e-04

Although all three methods consider lateral offset, velocity, and acceleration in cost functions, the method in [6] depends heavily on cost function parameters. A well-tuned set of parameters can perform well in some scenario, while another set of parameters can result in completely different performance (Fig.9). Our method systematically deals with this issue by adding the length of each polynomial segment as optimization parameter(Eq.7). According to Fig.9, when choosing two sets completely different cost function param-

eters, the path of our method does not change too much by adjusting the length of each polynomial segment: the two blue paths almost overlap, while the path generated by MSM varies greatly: there are apparent difference between the two red paths.

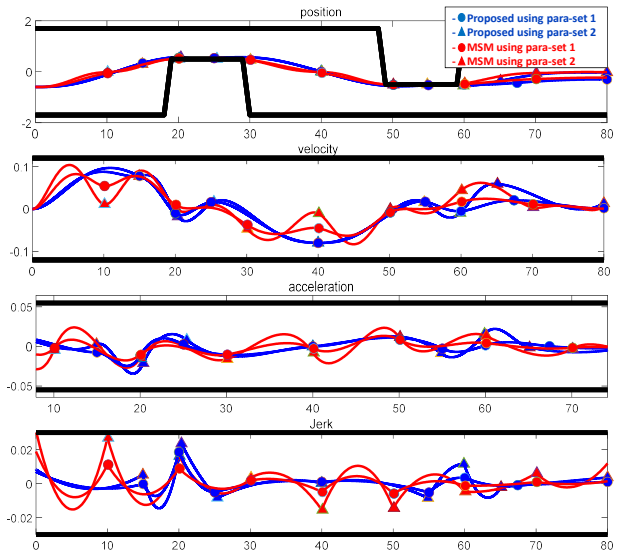

Fig. 9. Parameter Sensitivity Analysis: we choose two sets of completely different objective function parameters: the first set [1,10,100,100]; and the second set [1,100,800,10000].

The variable length also brings more flexibility in path planing. To deal with in-lane nurging scenarios, ego vehicle should deviate obviously when it approaches obstacles, thus its lateral velocity and acceleration increasing or decreasing quickly. The method in [6] uses equidistance piecewise polynoimials, and it greatly affect the shape of the path. Our method, however, can automatically adjust the length of each polynomial segment: when ego vehicle is far from obstacles, the legnth can be long, and when it approaches obstacles, the length becomes short so that the lateral velocity and acceleration of the path are smoother. Therefore, our method is much less sensitive to parameter variation.

VI. CONCLUSIONS

Our method first computes a collision-free reference line using QP, and then the path planning problem is formulated as non-linear programming, and each polynomial has variable length. We take advantage of corridor generation method in quadrotor path optimization so that road boundary constraints are converted to sparse but safe driving corridors. The path planning problem is iteratively solved to generate a smooth and feasible path by using SQP. Our method works within a fractions of a second, thus permitting efficient regeneration. Comparison with several state-of-the-art path planning methods shows that our method has better system performance.

REFERENCES

[1] Antoniou, Andreas, and Wu-Sheng Lu. Practical optimization: algorithms and engineering applications. Vol. 19. New York: Springer, 2007.

[2] Ziegler, Julius, et al. "Making bertha driveâĂŤan autonomous journey on a historic route." IEEE Intelligent transportation systems magazine 6.2 (2014): 8-20.

[3] Werling, Moritz, et al. "Optimal trajectory generation for dynamic street scenarios in a frenet frame." 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010.

[4] Jiang, Yuncheng, et al. "A Dynamic Motion Planning Framework for Autonomous Driving in Urban Environments." 2020 39th Chinese Control Conference (CCC). IEEE, 2020.

[5] Jiang, Yuncheng, et al. "DL-AMP and DBTO: An Automatic Merge Planning and Trajectory Optimization and its Application in Autonomous Driving." 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, 2021.

[6] Mellinger, Daniel, and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors." 2011 IEEE international conference on robotics and automation. IEEE, 2011.

[7] Richter, Charles, Adam Bry, and Nicholas Roy. "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments." Robotics research. Springer, Cham, 2016. 649-666.

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

[9] Zhang, Yajia, et al. "Optimal Vehicle Path Planning Using Quadratic Optimization for Baidu Apollo Open Platform." 2020 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2020.

[10] Zhang, Yajia, et al. "Optimal trajectory generation for autonomous vehicles under centripetal acceleration constraints for in-lane driving scenarios." 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019.

[11] Chen, Jing, Tianbo Liu, and Shaojie Shen. "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments." 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016.

[12] Gao, Fei, and Shaojie Shen. "Online quadrotor trajectory generation and autonomous navigation on point clouds." 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2016.

[13] Gao, Fei, et al. "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments." Journal of Field Robotics 36.4 (2019): 710-733.

[14] Ding, Wenchao, et al. "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor." IEEE Robotics and Automation Letters 4.3 (2019): 2997-3004.

[15] Gao, Fei, et al. "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial." 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018.

[16] Gao, Fei. Motion planning for autonomous flights: algorithms, systems, and applications. Diss. 2019.

[17] Stellato, Bartolomeo, et al. "OSQP: An operator splitting solver for quadratic programs." Mathematical Programming Computation 12.4 (2020): 637-672.