2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Motion Planning by Search in Derivative Space and Convex Optimization with Enlarged Solution Space

Jialun Li[1], Xiaojia Xie[2], Qin Lin[3], Jianping He[1] and John M. Dolan[4]

***Abstract*— To efficiently generate safe trajectories for an autonomous vehicle in dynamic environments, a layered motion planning method with decoupled path and speed planning is widely used. This paper studies speed planning, which mainly deals with dynamic obstacle avoidance given a planned path. The main challenges lie in the optimization in a non-convex space and the trade-off between safety, comfort, and efficiency. First, this work proposes to conduct a search in second-order derivative space for generating a comfort-optimal reference trajectory. Second, by combining abstraction and refinement, an algorithm is proposed to construct a convex feasible space for optimization. Finally, a piecewise Bézier polynomial optimization approach with trapezoidal corridors is presented, which theoretically guarantees safety and significantly enlarges the solution space compared with the existing rectangular corridors-based approach. We validate the efficiency and effectiveness of the proposed approach in simulations.**

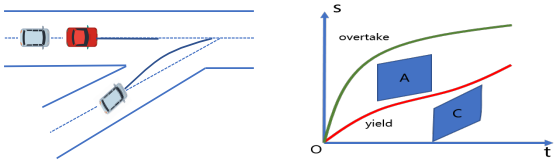
***Index Terms*— Speed planning, derivative space search, Bézier polynomial, safe trapezoidal corridors**

I. Introduction

Autonomous vehicles are promising to revolutionize transportation systems and change the ways in which people travel. To interact with other agents on road, a self-driving car adjusts its path and speed over time constantly based on perception information. This task can be formulated as a problem to optimize a trajectory in terms of comfort and energy saving while satisfying safety and dynamic feasibility constraints. However, solving the original constrained optimization problem is intractable in real-time.

To fulfill the real-time performance requirement, there are two major trajectory generation frameworks: spatio-temporal planning [1]–[3] and path-speed decoupled planning (also called layered planning) [4]–[7]. These two approaches share the same hierarchical ideas, i.e., finding a heuristic solution as a reference first and optimizing to refine later.

Spatio-temporal planning considers spatial and temporal maneuvers simultaneously. The search and optimization processes are completed in three-dimensional space (i.e., the 2D position dimensions plus the time dimension). Correspondingly, the decoupled method decomposes a 3D planning problem into two stages: path planning and speed planning. In the first stage, path planning is executed to generate a path to avoid static, oncoming, and low-speed obstacles. In the second stage, we use speed planning to adjust a vehicle's speed to keep a safe distance from dynamic obstacles which block the formed path. Although the layered approach is prone to be suboptimal with the appearance of dynamic obstacles compared to the 3D optimization approach, its separated design process is more flexible. In addition, the computational complexity can be significantly reduced, as discussed in [4]–[6].



(a) (b)

Fig. 1: Examples of yielding and overtaking maneuvers in the merging scenario and its S-T graph (ego vehicle : red).

In the layered planning framework, speed planning plays a critical role in avoiding dynamic vehicles. For example, when another vehicle merges into the same lane as the ego vehicle (see Fig. 1(a)), the ego vehicle may follow the lane-changing vehicle while maintaining a safe distance. During this process, the speed of the ego vehicle is expected to conduct an evasive maneuver, e.g., slowing down first and then accelerating smoothly.

The most common approach to speed planning is to use an S-T graph describing the relationship between station and time. However, the S-T graph is more suitable as a position control tool for collision avoidance. The first research question is that *can we directly control acceleration and jerk for generating a comfort-optimal trajectory?* To address this question, we propose to use an $\ddot{S}$-T graph instead. A multistage graph can be established to transform the original comfort-optimality problem into a shortest pathfinding problem, which is solvable by off-the-shelf tools, e.g. greedy algorithms and depth first search (DFS) strategies.

Another common drawback of existing speed planning methods is that the safety constraints are indeed imposed at discretized time instants. However, we expect to assure safety for the whole planning horizon. Many works tackle this problem in a straightforward way by refining the time interval. However, this solution leads to more decision variables, higher computation costs, and a lack of theoretical guarantee. The second research question we address in this paper is that *how can we provably ensure safety in*

This work is supported by the NSF of China under Grant 61973218.

[1]: the Department of Automation, Shanghai Jiao Tong University, and Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China {jialunli, jphe}@sjtu.edu.cn.

[2]: Megvii (Face++) Technology Inc., Beijing, China xiexiaojia@megvii.com.

[3]: the Electrical Engineering and Computer Science Department, Cleveland State University, Cleveland, OH, USA q.lin80@csuohio.edu.

[4]: the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA jdolan@andrew.cmu.edu.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981961

continuous-time space?, i.e., the safety between any two consecutive sampling timestamps must be guaranteed as well. To address this problem, we leverage the appealing convex hull property of Bézier polynomials to enforce that the continuous trajectory always fall into a safe region. In addition, such an optimization problem's solution space is enlarged via our proposed trapezoidal corridors.

The pipeline of our framework can be briefly described as i) we generate a comfort-optimal trajectory as a reference using the $\ddot{S}$-T graph; ii) we construct convex regions by using a novel convexification algorithm; iii) trapezoidal corridors are constructed from the convex regions for Bézier polynomial-based trajectory optimization. Our work has a superior trade-off among comfort, safety, feasibility, and efficiency. a) Comfort and safety: Since an autonomous vehicle is a safety-critical system, safety should be always prioritized. The comfort-optimal trajectory serves as a reference and compromises safety in the final trajectory optimization stage when collision avoidance has been taken into account. However, the degree of compromise is manageable since in the final optimization, we penalize the deviation from the comfort-optimal reference. b) Feasibility: The solution space is maximized by using our trapezoidal corridors' construction. c) Efficiency: Our proposed planner fulfills the real-time requirement. For a 7s planning horizon, the average runtime costs are around 6ms and 12ms in numeric simulations and the CommonRoad simulations [8], respectively.

The contributions of our work are summarized as follows:

- We propose to use a $\ddot{S}$-T graph for the optimization of a comfort reference trajectory. A multistage graph can be established to transform the original comfort-optimal finding problem into a shortest path finding problem, which is easily solvable.
- We propose an efficient convexification algorithm through combining abstraction and refinement to construct a safe convex space for optimization.
- We provide a sufficient condition on coefficients of the Bézier polynomials to guarantee trajectory's safety in trapezoidal corridors theoretically. Compared with existing rectangular corridors [1], the condition is relaxed and the solution space is significantly enlarged, which leads to a higher chance of finding an optimal solution. Moreover, existing condition of the solution shows that one piece of the speed profile by a trapezoidal corridor is hardly achieved by multiple rectangular corridors.

The remainder of this paper is structured as follows. We review related works in Sec. II. We introduce necessary notations and background materials on the S-T graph for speed planning and Bézier polynomials in Sec. III. The heuristic search for reference trajectory and the convex safe region construction are presented in Sec. IV. In Sec. V, we present our optimization formulation. The simulation results and analysis can be found in Sec. VI. We make concluding remarks in Sec. VII.

II. Related Works

Speed Planning. The techniques of (speed) planning approaches can be classified into three categories: (i) search and optimization; (ii) sampling lattices and selecting the minimum cost trajectory; (iii) approximated optimization. The methods in the first category search the best candidate speed profile and construct convex regions for smoothing by optimization. Xu et al. first present a method of selecting the best lattice solution and conducting a post-optimization [5]. Baidu EM motion planner uses dynamic programming for search and piecewise monomial polynomials for optimization [9]. With a similar search approach, [10] optimizes speed using piecewise jerk speed optimization. Most of works in the first category conduct search directly on an S-T graph. In this work, we argue that using an $\ddot{S}$-T graph can directly control and minimize the acceleration and jerk to generate comfort-optimal trajectories (more details will be discussed in Sec. IV). Besides, many works pose safety constraints in discredited time domain. However, we expect the trajectory to have provable safety in continuous time domain. To tackle this challenge, we propose to use trapezoidal corridors-based Bézier polynomials optimization. Some control-related optimization methods are also in this category, such as model predictive control (MPC) [11] and constrained iterative linear quadratic regulator (CiLQR) [12]. The advantage of these approaches is that they mitigate the planning and control inconsistency problem, since the dynamic model has already been considered in the planning layer. However, the disadvantage is the high computation cost. Our proposed planner fulfills the real-time requirement. For a 7s planning horizon, the average runtime costs are 5ms and 10ms in numeric simulation and the CommonRoad simulation, respectively. For the second category, different speed lattices are sampled and combined with path lattices. The generated local spatial-temporal trajectories are evaluated and the best one with the minimum cost is selected. Related works see [4], [6]. For the third category, the formulated non-convex trajectory optimization problem is solved by off-the-shelf solvers or approximated by sequential convex problems [13]–[15].

Bézier Polynomials-Based Planning. In the area of unmanned aerial vehicles (UAVs), Bézier polynomials are widely used with rectangular corridors for trajectory optimization [16], [17]. Ding et al. borrow this idea and propose to use similar approaches for unmanned ground vehicles (UGVs) [1]. However, due to dynamic traffic participants, the safe regions of UGVs are time-variant and different from scenarios considered for UAVs. Accordingly, corridors for Bézier curves are also time-variant and a common convex hull property does not hold. Therefore, it becomes non-trivial to directly transfer corridor construction methods from UAVs to UGVs. The challenges lie in generations of collision-free convex corridors and enforcing Bézier curves in these time-dependent corridors for safety. In our work, we propose to use time-dependent trapezoidal corridors and give sufficient safety condition for Bézier curves. It is theoretically proved that the trapezoidal corridors can enlarge the solution space

for improved optimization.

III. S-T Graph and Trajectory Representation

A. S-T Graph

Speed planning decides *when* an autonomous vehicle should reach a point from a planned path. To do so, we introduce Frenét coordinates to describe the position of a vehicle with respect to a road for analytical convenience. In the Frenét frame, S represents the longitudinal displacement along the road. L represents lateral position with respect to the reference, e.g., the road centerline. An L-S graph and an S-T graph are commonly used in practice. L-S graph describes the lateral coordinate against longitudinal coordinate. S-T graph illustrates the change of longitudinal position with respect to time. The vehicle's trajectory is determined by combining planning results from the S-L and the S-T graphs. In planning process, obstacles are projected onto the L-S graph and a collision-free path is generated. Then, the dynamic obstacles that block the path at certain time interval are projected onto the S-T graph.

As an example, in Fig.1(b), two dynamic obstacles considered are projected onto S-T graph as a set of blue rectangles. The width of each rectangle equals to the length of the obstacle vehicle plus half length of ego vehicle. To ensure safety, the feasible space of the station curve should not have any overlap with the regions projected from obstacles. The solution space is non-convex in general. The station profile of the ego vehicle on the S-T graph reflects its distances from obstacles with respect to time and its decisions such as yielding, overtaking, following, etc. For example, the red curve between the blue areas implies yielding to the car in front, while the green curve illustrates overtaking.

B. Bézier Polynomials and Properties

A Bézier polynomial is a polynomial function represented by a linear combination of Bernstein bases. The n-th order Bézier polynomial is written as

$$B(t) = c_0 b_n^0(t) + c_1 b_n^1(t) + \cdots + c_n b_n^n(t),$$

where the Bernstein basis is $b_n^i(t) = C_n^i \cdot t^i \cdot (1-t)^{n-i}, t \in [0,1]$. The coefficients of the polynomial c_i $(i = 0, 1, \ldots, n)$ are called control points. Compared with a monomial polynomial, a Bézier curve has the following properties.

- The Bézier polynomial starts at the control point $B(0) = c_0$ and ends at the control point $B(1) = c_n$.
- **Convex hull property:** $B(t)$ is confined within a convex hull consisting of control points, i.e.

$$B(t) \in \left\{ \sum_{i=1}^{n} \mu_i c_i \,\middle|\, \mu_i = b_n^i(t) \right\}.$$

- **Hodograph property:** The derivative of $B(t)$, denoted by $\dot{B}(t)$, can also be written as a Bézier polynomial with control points $c_i^{(1)} = n \cdot (c_{i+1} - c_i), i = 0, 1, \ldots, n-1$. Generally, the recurrence of control points between the l-th-order derivative $\frac{\mathrm{d}^l B(t)}{\mathrm{d}t^l}$ and $(l+1)$-th-order derivative $\frac{\mathrm{d}^{l+1} B(t)}{\mathrm{d}t^{l+1}}$ of $B(t)$ can be established by

$$c_i^{(l+1)} = (n-l)(c_{i+1}^{(l)} - c_i^{(l)}).$$

C. Trajectory Representation using Bézier Polynomials

With the appealing convex hull property, the trajectory $B(t)$ is guaranteed to fall into the convex hull of control points. If the convex hull are generated from a collision-free region, then $B(t)$ is guaranteed to be safe as well. Note that such a provable safety guarantee applies to the whole continuous planning horizon.

To mitigate the numerical instability issue, piecewise Bézier polynomials with lower orders are used instead of using a high-order Bézier polynomial for the whole planning horizon. Note that $B(t)$ is defined on a fixed time interval $[0,1]$. For a whole trajectory with $m+1$ pieces, in each piece $[T_k, T_{k+1}]$ $(k = 0, 1, \cdots, m, T_0 = 0)$, we use a scaling transformation and a translation transformation in the time domain to map it into the interval $[0,1]$ [16]. Then, the whole piece-wise trajectory can be represented as

$$s(t) = h_k B\left(\frac{t - T_k}{h_k}\right), t \in [T_k, T_{k+1}], \tag{1}$$

where h_k is the scaling transformation factor and T_k is the translation transformation factor for $k = 0, 1, \cdots, m$.

IV. Heuristic Searching and Corridor Generations

In this section, we introduce our proposed method for generating a comfort-optimal reference trajectory and the efficient convexification algorithm for the construction of convex safe regions.

A. Search in Derivate Space

In most of the existing literature on layered motion planning, an S-T graph is directly used for searching reference trajectories for further optimization. However, even if sample intervals and fitting methods are carefully designed, the second-order (acceleration) and the third-order derivatives (jerk) of lateral and longitudinal displacements may still be uncomfortable to passengers. Indeed, we can reduce search resolution and introduce more variables for consecutive (T, S) and similarly treat acceleration and jerk. However, it is imaginable that we will introduce excessive variables for search. Instead, in this work, we propose to search in an $\ddot{S}$-T graph for a direct control for acceleration and jerk. This motivation can be explained by an analogy with position control and force control in a robot control problem. The position control (by analogy with the S-T optimization) is more suitable for a direct collision-avoidance optimization, while the force control (by analogy with the $\ddot{S}$-T) is more suitable for directly generating an acceleration-smooth and jerk-minimized trajectory.

B. Search for Longitudinal Planning

In an $\ddot{S}$-T graph, feasible interval of $\ddot{s}, [\ddot{s}_{\min}, \ddot{s}_{\max}]$, are discretized with a user-defined step $\Delta\ddot{s}$ with Δt_s as time interval. For search, we first illustrate how to generate a child node with a father node. Suppose that the state of the father node is $(s_{\text{father}}, \dot{s}_{\text{father}}, \ddot{s}_{\text{father}})$. The second-order derivate component of child node, $\ddot{s}_{\text{child}}$, is selected from

uniformly sampled candidate values in the $\ddot{S}$-T graph. Then, we generate n_s fitting points between $\ddot{s}_{\text{father}}$ and $\ddot{s}_{\text{child}}$ on the $\ddot{S}$-T graph by minimizing longitudinal acceleration and jerk (maximizing comfort) as

$$\min_{\ddot{s}[1],\dots,\ddot{s}[n_s]} w_2 \sum_{k=1}^{n_s} (\ddot{s}[k])^2 + w_3 \sum_{k=0}^{n_s} (\ddot{s}[k+1] - \ddot{s}[k])^2 \quad \text{s.t.} \quad \ddot{s}[0] = \ddot{s}_{\text{father}},\ \ddot{s}[n_s+1] = \ddot{s}_{\text{child}}, \tag{2}$$

where w_2 and w_3 are tunable parameters. Once the second-order derivate $\ddot{s}$ of fitting nodes are achieved, the states of n_s fitting nodes and child node are calculated by, for $k = 0, \dots, n_s$,

$$\dot{s}[k+1] = \dot{s}[k] + \frac{1}{2}\ddot{s}[k]\Delta t + \frac{1}{2}\ddot{s}[k+1]\Delta t,$$
$$s[k+1] = s[k] + \dot{s}[k]\Delta t + \frac{1}{3}\ddot{s}[k](\Delta t)^2 + \frac{1}{6}\dot{s}[k+1](\Delta t)^2,$$

with $\dot{s}[0] = \dot{s}_{\text{father}}$, $s[0] = s_{\text{father}}$ and $\Delta t = \frac{\Delta t_s}{n_s+1}$. The state of child node is $(s[n_s+1], \dot{s}[n_s+1], \ddot{s}[n_s+1])$. Then, the cost of child node is calculated by

$$cost_{\text{child}} = cost_{\text{father}} + cost_{\text{comfort}} + cost_{\text{obstacle}} + cost_{\text{v}_{\text{ref}}} + cost_{\text{heuritsic}}. \tag{3}$$

The search procedure is summarized in Algorithm 1.

Algorithm 1: Search for Heuristic s-t Profile

Input: Intial state x_0, end state x_{end} of ego vehicle and S-T graph
Output: s_{ref}, $\dot{s}_{\text{ref}}$ and $\ddot{s}_{\text{ref}}$

```
Initialize root node_0 with x_0
Calculate s̈ of fitting nodes by solving Eq. (2)
priority_queue.push(node_0)
while !priority_queue.empty( ) do
    node_father = priority_queue.pop( ) //pops the node with the lowest cost
    if node_father.t == t_end then
        break
    end if
    for s̈_child in S̈_candidate do
        Calculate node_child and node_fitting with S̈-T graph
        if IsCollisionFree(node_child) and IsCollisionFree(node_fitting) then
            Caculate cost_child with Eq. (3)
            priority_queue.push(node_child)
        end if
    end for
end while
return s_ref, ṡ_ref and s̈_ref
```

In Proposition 1, we show that the solution of the problem in Eq. (2) also satisfies the constraints of $\ddot{s} \in [\ddot{s}_{\min}, \ddot{s}_{\max}]$.

Proposition 1. *Suppose that $\ddot{s}_{father}, \ddot{s}_{child} \in [\ddot{s}_{min}, \ddot{s}_{max}]$. Then, the solution of problem (2) satisfies the constraints of $\ddot{s}$, i.e., $\ddot{s}[1], \dots, \ddot{s}[n_s] \in [\ddot{s}_{min}, \ddot{s}_{max}]$.*

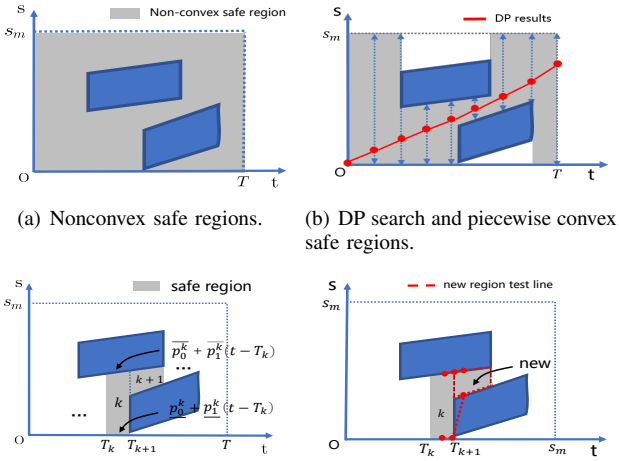
The detailed proof can be found the supplement materials [18]. We derive a closed-form solution of Eq. (2) instead of leveraging an optimization tool. Let the gradient of the objective function with respect to $\ddot{s}[k]$ be zero and we obtain

$$\ddot{s}[k] = \frac{w_3}{w_2 + 2w_3}(\ddot{s}[k-1] + \ddot{s}[k+1]) \tag{4}$$

where $k = 1, 2, \cdots, n_s$. We have the boundary conditions $\ddot{s}[0] = s_{\text{father}}$ and $\ddot{s}[n_s+1] = s_{\text{child}}$. Let $\lambda = \frac{w_3}{w_2+2w_3}$, and via solving a set of linear equations, we have

$$\begin{bmatrix} \ddot{s}[1] \\ \vdots \\ \ddot{s}[n_s] \end{bmatrix} = \begin{bmatrix} 1 & -\lambda & & \\ -\lambda & 1 & \ddots & \\ & \ddots & \ddots & -\lambda \\ & & -\lambda & 1 \end{bmatrix}^{-1} \begin{bmatrix} \lambda & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} \ddot{s}[0] \\ \ddot{s}[n_s+1] \end{bmatrix}.$$

The matrix under the inverse operation is tridiagonal. The time complexity of solving such a matrix is $O(n)$, which is significantly more efficient than $O(n^{2.8})$ of solving a general matrix inverse problem [18]. The resulting linear-time complexity implies that the optimization of fitting two nodes in the $\ddot{S}$-T has superior efficiency.



(a) Nonconvex safe regions. (b) DP search and piecewise convex safe regions.

(c) Representations of safe regions. (d) Generations of safe regions.

Fig. 2: Safety Regions and their Generations on S-T graph.

C. Piecewise Convex Safe Regions Representations

One of the main challenges of motion planning is that the free space is nonconvex (see the grey region in Fig. 2(a)). Suppose that the region can be divide into $m+1$ pieces with time intervals $[T_0, T_1], \dots, [T_m, T]$ and $T = T_{m+1}$ (see the piecewise-convex quadrilaterals in Fig. 2(b)). The details of such a convexification algorithm will be introduced in Sec. IV.C. As shown in Fig.2(c), the k-th convex safe region can be represented as

$$\mathcal{S}_k = \{(t_i, s_i) | \underline{p_0^k} + h_k \underline{p_1^k} \frac{t_i - T_k}{h_k} \le s_i \le \overline{p_0^k} + h_k \overline{p_1^k} \frac{t_i - T_k}{h_k}, t_i \in [T_k, T_{k+1}]\}, \tag{5}$$

where $\underline{p_0^k}, \underline{p_1^k}$ are the bias and the skew of the lower bound, $\overline{p_0^k}, \overline{p_1^k}$ are the bias and the skew of the upper bound. h_k denotes the length of the k-th time interval and satisfies $h_k = T_{k+1} - T_k$, $k = 0, 1, \dots, m$.

Then, the whole safe region is the union of a set of piecewise-safe sub-regions: $\mathcal{S} = \mathcal{S}_0 \cup \cdots \cup \mathcal{S}_m$ The speed planning is safe if $\forall t_0 \in [0, T], s(t_0) \in \mathcal{S}$, or for $t_0 \in [T_k, T_{k+1}]$, $s(t_0) \in \mathcal{S}_k$, $k = 0, 1, \dots, m$, i.e.

$$\underline{p_0^k} + h_k \underline{p_1^k} \frac{t_0 - T_k}{h_k} \le s(t_0) \le \overline{p_0^k} + h_k \overline{p_1^k} \frac{t_0 - T_k}{h_k}. \tag{6}$$

D. Construction of Piecewise Convex Safe Regions

Alg. 2 is the main routine for the construction of piecewise-convex safe regions. The original nonconvex space shown in Fig.2(a) is evenly chunked into $nums$ pieces (called meta-pieces) with identical time duration Δt. The upper boundaries ub and lower boundaries lb are from the free space in meta-pieces, of which the size is also $nums$. Note that for the space divided by obstacles, we select the unique space enclosing the comfort-optimal reference trajectory. $regions$ is the resulting convex regions. The first meta-piece is considered as the first convex region appended to $regions$ (see Line 1, Alg. 1). As a subroutine in Alg. 3, SingleRegionGeneration( ) computes the region's bias and skews of the lower bound and the lower bound, respective. In the loop of Alg. 2 (Lines 4-15), we iteratively evaluate two consecutive meta-pieces. If they can form a new skew (of upper bound or lower bound), which has a significant difference between the previously region (see the condition in Line 7, Alg. 2), a new single region will be established (see the illustration in Fig. 2(d)).

RegionsSplit( ) is used to check the length of each region. If it is above a user-defined threshold (e.g., $1s$ in our simulation setting), it will be split into multiple sub-regions, of which the time intervals are all below the threshold. This refinement operation aims to avoid underfitting. Conversely, the abstraction operation *RegionsMerge( )* aims to merge the small regions into a larger one, which aims to avoid overfitting and speed up the process.

Note that the linear lower and upper edges of the blue regions in Fig. 2 are formed due to the constant obstacle's velocity assumption. In practice, we suggest that when the predicted velocity is not constant, an extra fitting into lines or an overapproximation into a larger quadrilateral can be done in a pre-processing step.

Algorithm 2: Piecewise Convex Regions Generation

Input: $lb, ub, nums, \Delta t, \varepsilon$
Output: $regions$

1: **Initialize**: $i = 0,\ j = 1$ // i and j are counters for longitudinal boundaries and resulting convex regions, respectively
2: region = SingleRegionGeneration($lb, ub, 1$) // $i = 1$
3: $regions$.append($region$)
4: **for** $i = 2$ **to** $nums - 1$ **do**
5: $lskew = (lb[i] - lb[i-1])/\Delta t$ // lower meta-piece skew
6: $uskew = (ub[i] - ub[i-1])/\Delta t$ // upper meta-piece skew
7: **if** $\|lskew - regions[j-1].lskew\| > \varepsilon$ **or** $\|uskew - regions[j-1].uskew\| > \varepsilon$ **then**
8: $regions[j-1].t_{end} = i - 1$
9: $regions[j-1].t = (regions[j-1].t_{end}$
10: $-regions[j-1].t_{beg})\Delta t$
11: $region$ = SingleRegionGeneration(lb, ub, i)
12: $regions$.append($region$)
13: $j \leftarrow j + 1$
14: **end if**
15: **end for**
16: $regions[j-1].t_{end} = nums - 1$
17: $regions[j-1].t = (regions[j-1].t_{end}$
18: $-regions[j-1].t_{beg})\Delta t$
19: RegionsSplit($regions$)
20: RegionsMerge($regions$)
21: **Return** $regions$

Algorithm 3: SingleRegionGeneration(lb, ub, i)

Input: lb, ub, i
Output: $region$

1: $region.t_{beg} = i - 1$
2: $region.lskew = (lb[i] - lb[i-1])/\Delta t$
3: $region.lbias = lb[i-1]$
4: $region.uskew = (ub[i] - ub[i-1])/\Delta t$
5: $region.ubias = ub[i-1]$
6: **return** $region$

V. Piecewise Bézier Polynomial Optimization

In this section, we introduce our proposed trapezoidal corridors generation compared with the commonly used rectangular corridors, which have feasibility limitation. Then, the formulation of quadratic optimization using the trapezoidal corridors will be introduced.

A. Limitations of Enforcement in Rectangular Corridors

To enforce that the trajectory in the S-T graph stays in the safe region $\mathcal{S}$, the convex hull property of the Bézier function is used and a corridor is defined as follows.

Definition 1. *Let the coefficients of the Bézier Polynomial be $c_i \in \Omega, i = 0, 1, \ldots, n$, and these control points in the safe region $\mathcal{S}$ form a subset $\mathcal{S}^{cor} \subseteq \mathcal{S}$. Then, the subset is called a corridor.*

Rectangular corridors are the most commonly used. Constraints of the control points of rectangular corridors are given by the following proposition.

Proposition 2. *If a trajectory has control points in each time interval satisfying $c_i^k \in \Omega_{rec}^k$, where $\Omega_{rec}^k = \{c^k | \underline{p_0^k} + h_k \underline{p_1^k} \leq c^k \leq \overline{p_0^k}, i = 0, \ldots, n, k = 0, \ldots, m\}$, $s(t)$ is guaranteed to be safe, and the upper bounds and lower bounds form rectangular corridors $\mathcal{S}^{rec}$.*

Proof. Suppose that for $i = 0, \ldots, n$, $\underline{p_0^k} + h_k \underline{p_1^k} \leq c_i^k \leq \overline{p_0^k}$. According to the convex hull property of Bezier polynomial, we have

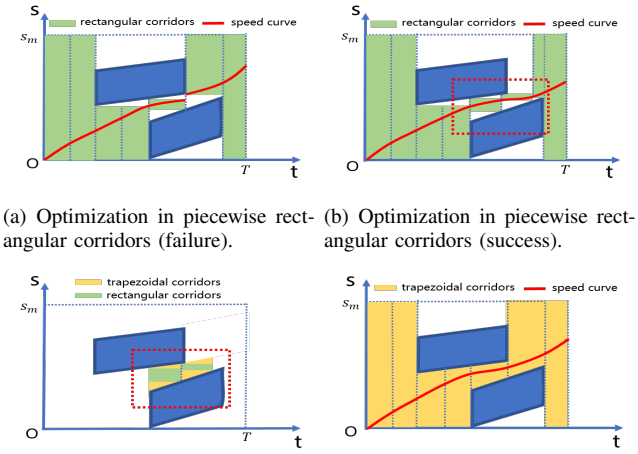
$$B(t) \leq \left(\sum_{i=1}^{n} b_n^i(t) \right) \overline{p_0^k} = \overline{p_0^k}. \tag{7}$$

Similarly, we have $B(t) \geq \underline{p_0^k} + h_k \underline{p_1^k}$. With conditions $c_i^k \in \Omega_1^k$, for $\forall t \in [T_k, T_{k+1}]$, it follows that

$$\begin{aligned} s(t) \in \mathcal{S}_k^{rec} = \{(t,s) |\ & \underline{p_0^k} + h_k \underline{p_1^k} \leq s_i \leq \overline{p_0^k}, \\ & t \in [T_k, T_{k+1}]\} \subseteq \mathcal{S}_k. \end{aligned} \tag{8}$$

Further, we have $\mathcal{S}^{rec} \subseteq \mathcal{S}$. □

Note that when we have $\underline{p_0^k} + h_k \underline{p_1^k} > \overline{p_0^k}$ (i.e., the lower bound is greater than the upper bound), there is no feasible solution for the optimization. In order to avoid this case, the time intervals of k-th corridor must satisfy $h_k \leq \frac{\overline{p_0^k} - \underline{p_0^k}}{\underline{p_1^k}}$.



(a) Optimization in piecewise rectangular corridors (failure).

(b) Optimization in piecewise rectangular corridors (success).

(c) Suboptimal of optimization in rectangular corridors.

(d) Optimization in piecewise trapezoidal corridors.

Fig. 3: Optimization in trapezoidal and rectangular corridors.

Besides, if $\exists\ \underline{p_1^k} > 0$ or $\overline{p_1^k} > 0$, the safe regions are not fully covered by rectangular corridors. As a result, constraints on control points to enforce the station curve in rectangular corridors are overtighten with reduced solution space (see the illustrations in Fig. 3(a) and 3(c)). In the next subsection, we will introduce our proposed trapezoidal corridors with enlarged solution space.

B. Safety Enforcement in Trapezoidal Corridors

The sufficient conditions of control points c_i to keep the longitudinal trajectory in our proposed trapezoidal corridors and safe are built upon the following lemma (see proof in [18]).

Lemma 1. *Let $M \in \mathbb{R}^{(n+1)\times(n+1)}$ denote the transition matrix from the Bernstein basis $\{b_n^0(t), b_n^1(t), \ldots, b_n^n(t)\}$ to the monomial basis $\{1, t, \ldots, t^n\}$. We have $M_{i,0} = 1,\ 0 \le M_{i,j} \le 1,\ i = 0, 1, \ldots, n,\ j = 0, 1, \ldots, n$.*

With Lemma 1, the constraints on control points are provided by Theorem 1 to guarantee one piece of trajectory in time-dependent trapezoidal corridor (see proof in [18]).

Theorem 1. *For a trajectory, if it has control points in each time interval satisfying $c_i^k \in \Omega^k$, where $\Omega^k = \{c^k | \underline{p_0^k} + h_k \underline{p_1^k} M_{i,1} \le c_i^k \le \overline{p_0^k} + h_k \overline{p_1^k} M_{i,1}, i = 0, 1, \ldots, n, k = 0, 1, \ldots, m\}$, $s(t)$ is guaranteed to be safe. The upper bounds and lower bounds form a trapezoidal corridor $\mathcal{S}^{tra}$.*

In Theorem 1, conditions on c_i is $\underline{p_0^k} + h_k \underline{p_1^k} M_{i,1} \le c_i^k \le \overline{p_0^k} + h_k \overline{p_1^k} M_{i,1}$. Compared to safety enforcement in rectangular corridors in Proposition 2, we have $\underline{p_0^k} + h_k \underline{p_1^k} M_{i,1} \le \underline{p_0^k} + h_k \underline{p_1^k}$ and $\overline{p_0^k} + h_k \overline{p_1^k} M_{i,1} \ge \overline{p_0^k}$. The advantage of having trapezoidal corridors is twofold:

- **Existence of solution:** In the trapezoidal safe regions represented by Eq. (8), by setting $t = T_k + h_k M_{i,1}$, we have $\underline{p_0^k} + h_k \underline{p_1^k} M_{i,1} < \overline{p_0^k} + h_k \overline{p_1^k} M_{i,1}$. This means that the lower boundaries are guaranteed to be smaller than the upper boundaries all the time. Then, there always exists control points c_i to satisfy the safety constraints. Recall that for the rectangular corridors, we need to always check $h_k \le \frac{\overline{p_0^k} - p_0^k}{\underline{p_1^k}}$.
- **Optimality of solution:** The constraints are relaxed, therefore the solution space is enlarged compared with the rectangular corridors (see the illustration for the comparison in Fig. 3).

C. Trajectory Optimization Formulation

The objective function is established as

$$J = w_1 \sum_{k=1}^{m} (c_n^k - s_{\text{ref}}[\sum_{l=1}^{k} m_l])^2 + w_2 \int_0^T (\dot{s}(t) - \dot{s}_{\text{ref}})^2 \,\mathrm{d}t$$
$$+ w_3 \int_0^T \ddot{s}(t)^2 \mathrm{d}t + w_4 \int_0^T \dddot{s}(t)^2 \mathrm{d}t + w_5 \left(c_n^m - s_{\text{ref}}[\sum_{l=1}^{m} m_l] \right)^2,$$

where $s_{\text{ref}}(t)$ is the reference longitudinal trajectory as the output from heuristic search and $\dot{s}$ is the longitudinal reference velocity. The first term penalizes the deviation from the reference. The second one penalizes the deviation between the actual and reference speed. The third and fourth terms penalize acceleration and jerk, respectively. The last term penalizes the deviation of the ending station from the reference.

The optimization considers the following constraints.

- Boundary Constraints. The piecewise curve starts from fixed position, speed, and acceleration, i.e.

$$c_i^{0,l} h_k^{(1-l)} = \left. \frac{\mathrm{d}^l s(t)}{\mathrm{d}t^l} \right|_{t=0}, \quad l = 0, 1, 2,$$

  where $c_i^{k,l}$ is the control point for the l-th-order derivative of the k-th Bézier curve.

- Continuity Constraints. The precewise curve must be continuous at the connected time points for position, speed, and acceleration.

$$c_n^{k,l} h_k^{(1-l)} = c_0^{k+1,l} h_{k+1}^{(1-l)},\ l = 0, 1, 2; k = 0, \ldots, m-1.$$

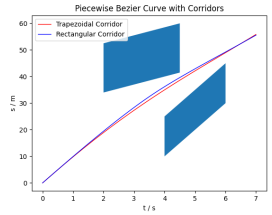
- Safety Constraints. With our proposed trapezoidal corridors, safety constraints can be represented as

$$\underline{p_0^k} + h_k \underline{p_1^k} M_{i,1} \le c_i^{k,0} \le \overline{p_0^k} + h_k \overline{p_1^k} M_{i,1}, k = 0, \ldots, m.$$

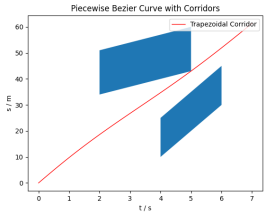
- Physical Constraints. The physical constraints under consideration include the limit of a vehicle's velocity, acceleration, and jerk. We can use the hodograph property (iv) of a Bézier curve to calculate velocity, acceleration, and jerk. The constraints are given by

$$\underline{\beta^{k,1}} \le c_i^{k,l} \le \overline{\beta^{k,1}}; \underline{\beta^l} \le c_i^{k,l} \le \overline{\beta^l},\ l = 2, 3,$$

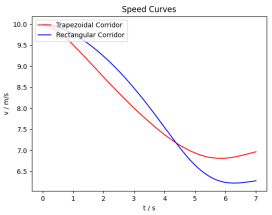
  where $k = 0, \ldots, m$ and it follows that $c_i^{k,l+1} = (n-l)(c_{i+1}^{k,l} - c_i^{k,l})$. The upper bound $\overline{\beta^{k,1}}$ is determined by the speed limit on road. Let a_{cm} be the maximum acceleration and κ_k be the maximum curvature of the path for $t \in [T_k, T_{k+1}]$ (see [19] for details). The lateral acceleration constraints are $c_i^{k,l} \le \overline{\beta^{k,1}} = \sqrt{\frac{a_{cm}}{\kappa_k}}$.



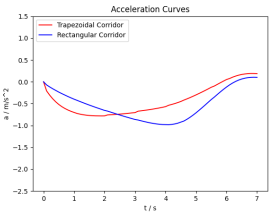
(a) Bézier curves with obstacles (Both types of corridors have solutions).



(b) Bézier curves with obstacles (Only trapezoidal corridors have the solution).



(c) Speed profiles.



(d) Acceleration profiles.

Fig. 4: Bézier curves within trapezoidal and rectangular corridors.

Finally, the trajectory optimization process can be formulated as a quadratic programming (QP) problem. We refer readers to the detailed formulation process in [18]. This problem can be solved in real-time by a modern solver such as OSQP [20].

VI. Simulations and Results Analysis

Our framework has been implemented using C++11. All simulations are carried out on a personal computer with a dual-core 2.90GHz Intel i5-4210H processor.

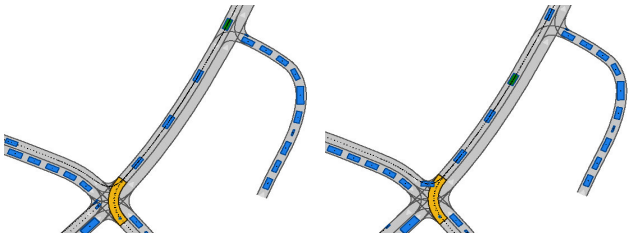
A. Numerical Simulations

We conduct numerical simulations to validate the optimality and the low failure rate of the proposed approach compared to Bézier polynomial with rectangular corridors. In the simulation, we consider a similar scenario in Fig.1 (see the similar S-T graphs in 1(b), 4(a), and 4(a)). The initial velocity and the acceleration of the ego vehicle runs are $v(0) = 10.0m/s$ and $a(0) = 0m/s^2$, respectively. The parameters are set to be $w_1 = 0.1$, $w_2 = 0.1$, $w_3 = 10.0$, $w_4 = 5.0$ and $w_5 = 3.0$.

Fig.4(a) and Fig.4(b) show Bézier curves generated by using rectangular and trapezoidal corridors in a normal case and a corner case. We observe that in the corner case in which the planned trajectory is close to the obstacle, rectangular corridor-based approach fails to find a solution. Fig. 4(c) and 4(d) illustrate corresponding speed profile and acceleration profile for the normal case from Fig. 4(a). They clearly show that the trajectory generated by the trapezoidal corridors is smoother. Table I summarizes the maximum accelerations and the average accelerations for two cases.

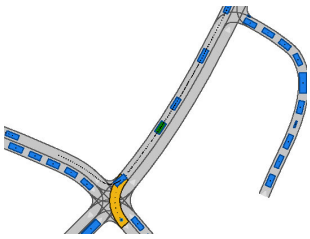
TABLE I: Comfort of planners by different corridors.

	case in Fig. 4(a)		case in Fig. 4(b)	
	Max. Acc.	Ave. Acc.	Max. Acc.	Ave. Acc.
Rectangular	0.95	0.62	-	-
Trapezoidal	**0.78**	**0.54**	**0.96**	**0.59**

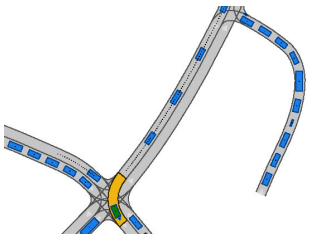


(a) $t = 0$.

(b) $t = 2s$.



(c) $t = 4s$.



(d) $t = 8s$.

Fig. 5: Illustrations of the slowing down and yielding scenario.

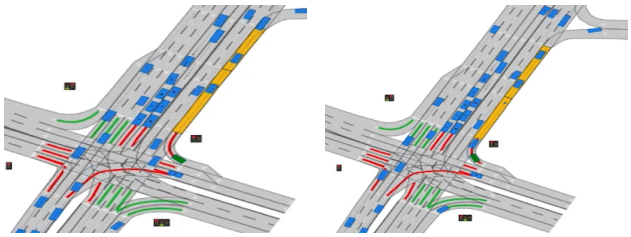
B. Simulations in CommonRoad platform

The simulations in this part are conducted in CommonRoad, which is a platform provides interactive simulated and non-interactive real traffic data. As an example shown in Fig. 5, the ego vehicle is illustrated as a green box and other traffic participants are blue boxes. A given scenairo of problem is considered as "solved" when the ego vehicle reaches the yellow region and no collision happens.

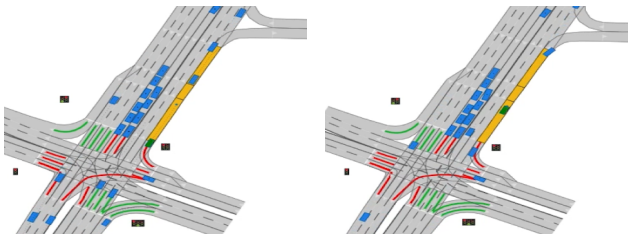
- **Case 1: Slowing down and yielding scenario**
  In this scenario, there is a front vehicle running ahead of the ego vehicle at the same lane (see Fig. 5(a)). When the front vehicle gets closer to the intersection, it slows down and turns right (Fig. 5(b), 5(c)). To keep a safe distance from the front and the behind vehicles, the ego vehicle also slows down(Fig. 5(c)). The ego vehicle safely reaches the target region and turns left (Fig. 5(d)).
- **Case 2: Merging scenario**
  In this scenario, the ego vehicle turns right at the intersection (see Fig. 6(a)). A vehicle moves at high speed and merges into the same lane from the left rear of the ego vehicle. The ego vehicle first slow down to avoid the vehicle (see Fig. 6(b)). When the vehicle passes in front of the ego vehicle. The ego vehicle accelerates to reach the goal region (see Fig. 6(c), 6(d)).
- **Case 3: Lane changing scenario**
  In this scenario, the ego vehicle changes lane at the intersection. It adjusts its speed to keep a safe distance from the front vehicle (see Fig. 7).

VII. Conclusion

In this paper, we investigate speed planning for autonomous vehicles. We first use dynamic programming to search a comfort-optimal reference trajectory. Then, a convexification algorithm generating trapezoidal corridors is proposed. We provide the sufficient conditions of control points in the trapezoidal corridors to provebly guarantee the safety

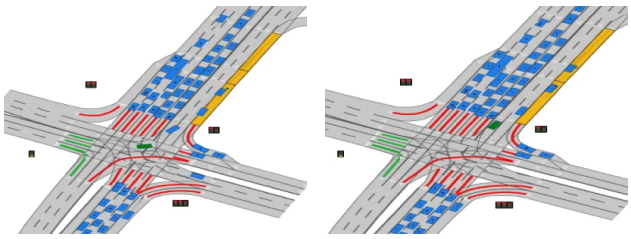


(a) $t = 2s$. (b) $t = 4s$.

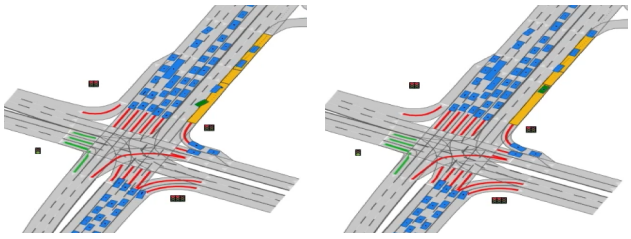


(c) $t = 6s$. (d) $t = 8s$.

Fig. 6: Illustrations of the merging scenario.



(a) $t = 2s$. (b) $t = 4s$.



(c) $t = 6s$. (d) $t = 8s$.

Fig. 7: Illustrations of the lane changing scenario.

of trajectories represented by Bézier polynomials. Compared with the existing rectangular corridors-based method, this work is proved to have enlarged solution space. Finally, we formulate the trajectory optimization as a solvable QP problem. The numeric simulations show that proposed approach is superior in terms of optimality and low failure rates. We also validate our method in real complex traffic environment in the CommonRoad platform.

REFERENCES

[1] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE RAL*, 2019.

[2] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—a local, continuous method," in *2014 IEEE IV*.

[3] T. Mercy, W. Van Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," in *2016 European Control Conference (ECC)*.

[4] T. Gu, J. M. Dolan, and J.-W. Lee, "Runtime-bounded tunable motion planning for autonomous driving," in *2016 IEEE IV*.

[5] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE ICRA*.

[6] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Trans. on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2015.

[7] C. Qu, J. He, J. Li, C. Fang, and Y. Mo, "Moving target interception considering dynamic environment," *2022 American Control Conference (ACC)*.

[8] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.

[9] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo EM motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[10] J. Zhou, R. He, Y. Wang, S. Jiang, Z. Zhu, J. Hu, J. Miao, and Q. Luo, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robotics and Automation Letters*, 2020.

[11] S. Khaitan, Q. Lin, and J. M. Dolan, "Safe planning and control under uncertainty for self-driving," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 9826–9837, 2021.

[12] Y. Pan, Q. Lin, H. Shah, and J. M. Dolan, "Safe planning for self-driving via adaptive constrained iLQR," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2377–2383.

[13] C. Liu, W. Zhan, and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 154–159.

[14] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

[15] H. Wang, X. Ding, J. He, K. Margellos, and A. Papachristodoulou, "Safety-aware optimal control in motion planning," *arXiv preprint arXiv:2204.13380*, 2022.

[16] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 344–351.

[17] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[18] J. L. et al., "Motion Planning by Search in Derivative Space and Convex Optimization with Enlarged Solution Space," https://github.com/JialunLi/Derivate-Space-Search-Supplement-Material-, 2022, [extended version].

[19] Y. Zhang, H. Sun, J. Zhou, J. Hu, and J. Miao, "Optimal trajectory generation for autonomous vehicles under centripetal acceleration constraints for in-lane driving scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3619–3626.

[20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, pp. 1–36, 2020.