Autonomous Driving Trajectory Optimization With Dual-Loop Iterative Anchoring Path Smoothing and Piecewise-Jerk Speed Optimization

Jinyun Zhou, Runxin He, Yu Wang, Shu Jiang, Zhenguang Zhu, Jiangtao Hu, Jinghao Miao, *Member, IEEE*, and Qi Luo, *Graduate Student Member, IEEE*

***Abstract*—This letter presents a free space trajectory optimization algorithm for autonomous driving, which decouples the collision-free trajectory generation problem into a Dual-Loop Iterative Anchoring Path Smoothing (DL-IAPS) problem and a Piecewise-Jerk Speed Optimization (PJSO) problem. The work leads to remarkable driving performance improvements including more robust and precise collision avoidance, higher control feasibility, higher computation efficiency and stricter driving comfort guarantee, compared with other existing algorithms. The advantages of our algorithm are attributed to our fast iterative collision checks with exact vehicle/obstacle shapes, strict non-holonomic dynamic constraints and accurate kinematics-based speed optimization. It has been validated that, through batch simulation and road experiments, compared with prior works, our algorithm is with the highest robustness and capable to maintain the lowest failure rate ($\sim$7%) at nearly all test conditions, achieves 10x faster computational speed than other planners, fulfills 100% driving-comfort standards in complex driving scenarios, and does not induce significant time increase as boundaries or obstacles scale up.**

***Index Terms*—Intelligent transportation systems, autonomous agents, nonholonomic motion planning.**

I. Introduction

In RECENT years, autonomous driving technology is making a huge progress on handling numerous lane cruising scenarios including lane following, lane changing, stopping at traffic lights, etc [1], [2]. However, many of the motion planning algorithms drive the vehicle to strictly follow the lane or disallow backward driving [3]. Such restrictions degrade the vehicle's capability to handle parallel and perpendicular parking or scenarios when the car needs backward driving or passing a semi-structured area. Thus free space motion planning algorithm with the ability to both forward and backward gear is essential for expanding the vehicles' geo-fenced operation areas and enabling curb-to-curb operation. Compared with relatively standardized on-lane planning, the free space motion planning asks for more effective algorithms to conquer more "open-ended" questions and address: 1) *non-holonomic vehicle dynamic constraints*, because under free space scenarios there are lots of chances for maneuvering the car to move at the way close to its physical limits, 2) *precise obstacle collision avoidance*, to seek out a narrow but really "passable" path when passing-by other obstacles with small gaps (10 − 20 cm) in between, 3) *real time computation*, and 4) *driving comfort*. These technical barriers make free space planning a challenging topic in the autonomous driving field [4].

Historically, two major approaches of general free space motion planning have been researched. Path/speed coupled method jointly solves the path and speed optimization problem. Hierarchical Optimization-Based Collision Avoidance (H-OBCA) [5] applies Nonlinear Model Predictive Control for obstacle avoidance. Timed Elastic Bands (TEB) [6] is based on the flat system model and now it is extended to Lie group [7] with ego-circle representation [8]. These approaches are able to jointly consider vehicle dynamics and obstacle avoidance into an one-shot optimization problem but usually have large computational burden or low robustness [9]. The decoupled methods, for example Convex Elastic Band Smoothing (CES) [10] and Convex Feasible Smoothing [11], derive and decouple the path/speed planning [12]. Recently, due to their computation efficiency, the relative optimization methods in this category are widely applied in many other robotic fields such as unmanned aerial vehicles [13], [14] and 7R robots [15]. Although with better computational efficiency, decoupled methods usually lack control feasibility and cannot guarantee path or speed smoothness in some extreme cases, for example, full-steer turn in a narrow aisle in parking lot or obstacle avoidance in an alley with a "zig-zag" path [4], [16].

In this letter, we propose a novel path/speed decoupled method that both leverages the inherent high computational efficiency and effectively smooths the path/speed trajectories. We first employ a searching based path planning method, such as Hybrid A* [16], to generate a global jerky and coarse path as original reference, then we decouple the trajectory optimization problem into two hierarchical steps including, iterative curvature constrained path smoothing (i.e., DL-IAPS) and comfortable minimum-time piecewise-jerk speed optimization (i.e., PJSO) to address the above mentioned issues with following advantages:

Manuscript received August 24, 2020; accepted December 6, 2020. Date of publication December 21, 2020; date of current version January 4, 2021. This letter was recommended for publication by Associate Editor S. Manzoor and Editor Y. Choi upon evaluation of the reviewers' comments. *(Jinyun Zhou and Runxin He have contributed equally to this work.) (Corresponding author: Qi Luo.)*

The authors are with the Baidu USA LLC, 1195 Bordeaux Drive, Sunnyvale, California 94089 USA (e-mail: zhoujiny404@gmail.com; runxinhe@hotmail.com; wangyu59@gmail.com; shujiang2016@gmail.com; zzgthk@163.com; boris.hu@gmail.com; miaojinghao@baidu.com; luoqi06@baidu.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2020.3045925, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3045925

1) **Robust and with High Successful Rate**: We show in Section IV-A that our algorithm successfully maintains a low failure rate (6.25%−7.50%) during batch test and outperforms other selected planning algorithms across different planning time resolutions, especially when vehicle starts from some challenging initial conditions that other algorithms fail. The high robustness of our algorithm is attributed to two features as follows:
   a) **Precise Collision Avoidance**: Prior works [17], [18] either made a linear approximation to the collision avoidance constraints or over-simplified the ego car's shape as a circle with an extra buffer added. These estimations are too conservative to exploit the search space and thus usually fail planning in narrow space due to reduced search space. We perform iterative collision checks with precise obstacle shapes and polygon-like ego vehicle shape, to expand search space and reduce planning failure rates.
   b) **Control Feasibility**: Unlike some prior works which neglected to incorporate the maximum curvature/acceleration constraints introduced by non-holonomic vehicle dynamics [11], [19] or only had valid curvature constraints when the smoothed trajectory was close to original one [10], our approach strictly enforces non-holonomic constraints with modified Sequential Convex Optimization (SCP) path planning. We show this can eliminate the possible planning failures induced by violating non-holonomic constraints.
2) **High Computational Efficiency**: In the DL-IAPS + PJSO algorithm, we perform iterative collision checks with polygon-like ego vehicle shape within an average time of 0.088−0.127 sec with no obstacle added. This is 10x faster compared with other algorithms as shown in IV-A. We also show our planner achieves 0.183−0.206 sec computational time with complex obstacles/boundaries for various trajectories (with full-length of 9−14 sec), which demonstrates that our algorithm will not induce significant time increase as boundaries or obstacles scale up, as shown in IV-B.
3) **Driving Comfort and Minimum Traversal Time**: Inspired by T. Lipp and S. Boy [20] and W. Lim *et al.* [21], We formulate the speed profile optimization in a form where both minimal traversal time and driving comfort are included in the optimization objectives as hard constraints, to provide better driving experiences in the Robotaxi (i.e., self-driving taxi operated for a ridesharing service). We limit the jerk bounds according to prior driving comfort study in [22] and show that the DL-IAPS + PJSO optimized trajectories are 100% within the comfort bounds in Section IV-A.

Our proposed planner together with benchmark planners are integrated in the Apollo Autonomous Driving Platform.[1] We validate our work by 80 numeric simulation cases with different initial conditions, 208 simulation scenarios extracted from real world, and 400 hours on-road tests in US and China to confirm its efficiency and robustness in different free space driving scenarios.

[1]Source for all planners in comparison available at https://github.com/ApolloAuto/apollo, except TEB algorithm from https://github.com/rst-tu-dortmund/teb_local_planner.

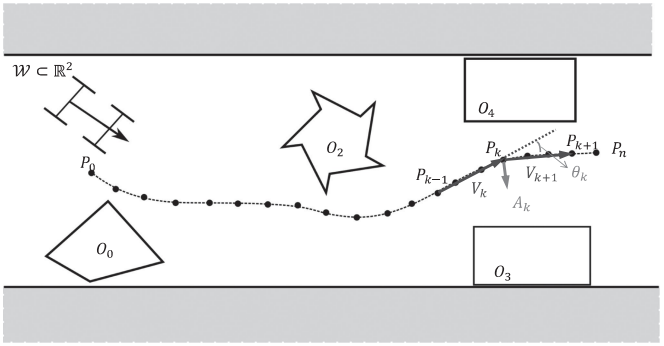


Fig. 1. Illustration to problem statement.

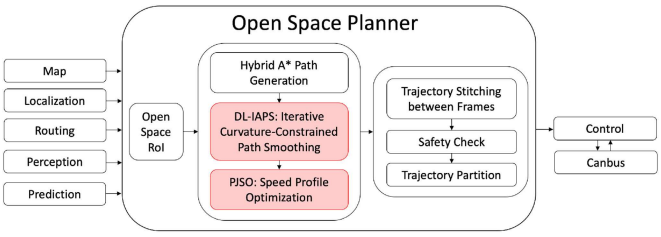


Fig. 2. Open Space Planner Architecture.

II. Problem Statement

As shown in Fig. 1, $\mathcal{W} \subset \mathbb{R}^2$ denotes the work space for a vehicle, and $\mathcal{O} = \{O_i\}_{i=1}^m$ denotes the collection of obstacles. At time step k, the ego vehicle's state can be described with location $P_k = {}^{x_k, y_k}$, heading angle ϕ_k and unit heading vector $\hat{u}_{\phi_k}$. Also, as shown in Fig. 1, given two consecutive step $k-1$ and k, ego vehicle position change vector can be defined as $V_k = P_k - P_{k-1}$, and its change rate vector defined as $A_k = V_{k+1} - V_k$, the included angle between two consecutive position change vector V_{k+1} and V_k is defined as θ_k. For point P_k, the longitudinal traverse distance, speed and acceleration are defined as ${}^{s_k, \dot{s}_k, \ddot{s}_k} \in \mathbb{R}^3$.

With the proposed algorithm, a complete autonomous driving planning module architecture is designed and implemented as shown in Fig. 2. A trajectory planner, named Open Space Planner, contains three consecutive modules:

1) **Region of Interest (RoI)**: This module receives information from map and perception, filters out far away or fast-moving obstacles and defines a task specific end position and collision free area for later modules
2) **Trajectory Generation**: This module contains three parts: collision free Hybrid A* path searching, iterative curvature-constrained path smoothing (i.e., DL-IAPS) and speed profile optimization (i.e., PJSO).

3) **Trajectory Post-processing**: This module contains three parts: trajectory stitching (with trajectory from last planning cycle), collision check (for fast-moving obstacles), and trajectory partition that splits trajectory into forward/backward pieces.

This letter mainly focuses on designing and implementing the iterative curvature constrained path smoothing and speed profile optimization (highlighted in red in Fig. 2), with the assumption that the upstream searching-based planner has generated a collision free reference path $\mathcal{P} = \{P_i\}_{i=1}^{n}$.

III. Path Speed Decoupled Trajectory Optimization

The proposed path/speed decoupled trajectory planning contains two parts: path smoothing design in III-A and speed profile optimization in III-B.

A. Dual-Loop Iterative Anchoring Path Smoothing

In this subsection, we introduce a Dual-Loop Iterative Anchoring Path Smoothing (DL-IAPS) for collision avoidance and path smoothing. The overall algorithm is shown in Algorithm 1. The inner loop starts from the referenced collision free trajectory via Hybrid A* and smooths the path with curvature constraint via SCP, and the outer loop checks collision avoidance and shrinks the corresponding state feasible region conditionally. Outer iteration terminates when the smoothed path passes collision check to all obstacles.

1) Inner Loop for Curvature Constrained Path Smoothing: Inspired by the elastic band approach path smoothing method [10], [6], with vehicle turning radius R and its minimum value $R_{\min}$, a relation in (1) between position vector change rate A_k and maximum path curvature $1/R_{\min}$ can be approximated based on the assumption that P_k are uniformly and densely spread over the path,

$$\begin{aligned}\|A_k\| &= \|V_{k+1} - V_k\| \approx 2 * \|V_k\| * \sin(\theta_k/2) \\ &\approx \|P_k - P_{k-1}\|^2/R \leq \|P_k - P_{k-1}\|^2/R_{\min} \end{aligned} \tag{1}$$

It is worth mentioning that the work in Algorithm CES deals with the curvature constraint in (1) with an assumption that the length of the smoothed path, $\|P_k - P_{k-1}\|^2$ in (2f), is nearly the same as that of the input reference path, $\|P_k^{ref} - P_{k-1}^{ref}\|^2$, so that the order of curvature constraint can be reduced from quartic to quadratic [10]. However, the curvature performance comparison in Section IV-A shows Algorithm CES's approach tends to invalidate the maximum curvature constraint when the smoothed path is much shorter than the reference path in extreme cases, which affects control feasibility.

With above path curvature constraint being a quartic constraint, the nonlinear path smoothing optimization problem is formulated as:

$$\min_{P} \sum_{k=1}^{n-2} \|2P_k - P_{k-1} - P_{k+1}\|^2 \tag{2a}$$

subject to:

$$P_0 = P_{0_{ref}} \text{ and } P_{n-1} = P_{n-1_{ref}}, \tag{2b}$$

Algorithm 1: DL-IAPS Path Planning.

Variables:

f : cost function as in (2a)
P_k : vehicle positions $[x_k, y_k]$
k : time step, $k = 0, \ldots, n-1$
g : inequality constraint on curvature in (2f)
s : slack variable with respect to g
t : trust region size
μ : penalty coefficient
$\mathcal{B}$: state bubble size

Parameters:

α : penalty scaling factor
ρ : trust region adaptation threshold
γ^+, γ^- : trust region change ratio
f_{tol} : cost function convergence threshold
x_{tol} : decision variables convergence threshold
c_{tol} : constraint satisfaction threshold
β : state bubble change ratio

```
▷ begin of outer loop for collision avoidance
for Collision Check iteration = 1, 2, ... do
    ▷ begin of inner loop for path smoothing
    for Penalty iteration = 1, 2, ... do
        for Sub-problem iteration = 1, 2, ... do
            ĝ = linearization(g, P_{last−iteration})
            for Trust Region iteration = 1, 2, ... do
                P ← arg min_P f(P) + μ Σ_{i=0}^{m_{inequ}} s_i
                if TrueImprove/ModelImprove> ρ then
                    t ← t * γ^+ ; break
                else
                    t ← t * γ^−
                end
                if t < x_tol then
                    break
            end
            if converge according to x_tol or f_tol then
                break
        end
        if constraints satisfied to tolerance c_tol then
            break
        else
            μ ← α * μ
        end
    end
                                ▷ end of the inner loop
    for Obstacle number = 1, 2, ... do
        for path point = 1, 2, ... do
            if Full dimension collision detected then
                B_k ← β * B_k
            else
                continue
            end
        end
    end
end
                                ▷ end of the outer loop
```

$$P_1 = P_{0_{ref}} + \|P_1 - P_0\| * \hat{u}_{\phi_0}, \tag{2c}$$

$$P_{n-2} = P_{n-1_{ref}} + \|P_{n-1} - P_{n-2}\| * \hat{u}_{\phi_{n-1}}, \tag{2d}$$

$$P_k \in \mathcal{B}_k, \text{ for } k = 2, \ldots n-3, \tag{2e}$$

$$g(P) = \|2P_k - P_{k-1} - P_{k+1}\|^2 - \frac{\|P_k - P_{k-1}\|^4}{R_{\min}^2} < 0,$$
$$\text{for } k = 1, \ldots n-2, \tag{2f}$$

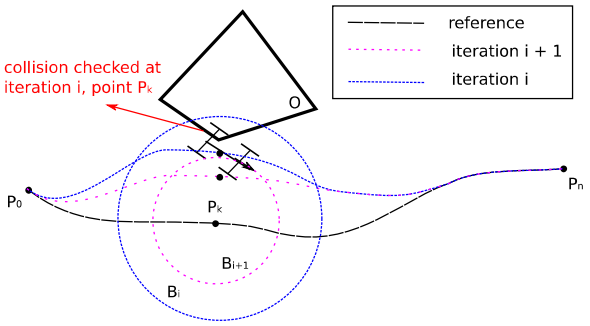


Fig. 3. Illustration of collision check and $\mathcal{B}_k$ updates.

The notations in (2) are defined in Section II. Each optimization cost term in (2a) tries to reduce the geometrical difference between the current path point and its neighboring points along the new trajectory. Such cost term encourages every three consecutive points in a straight line therefore minimizes the curvature. ϕ_0 and ϕ_{n-1} are headings of path's initial and end points respectively, which are the same as the reference path initial and end points headings. $\hat{u}_\phi$ is the unit norm vector along the direction ϕ. $\mathcal{B}_k$ as shown in Fig. 3 is state bubble constraining the feasible region of a point's position in the optimization problem.

Equation (2) is hard to solve due to the non-linearity in constraints (2e) and (2f), so we leverage SCP to solve it. SCP approximates the original problem as a convex quadratic programming sub-problem around its current iteration point to solve, and repeats the above process until the original problem fully solved [17]. The corresponding approximated convex sub-problem is reformulated as (3):

$$\min_{P,\,d} \sum_{k=1}^{n-2} \|2P_k - P_{k-1} - P_{k+1}\|^2 + \mu \sum_{k=1}^{n-1} s_k \tag{3a}$$

subject to:

$$P_0 = P_{0_{ref}} \text{ and } P_{n-1} = P_{n-1_{ref}}, \tag{3b}$$

$$P_1 = P_{0_{ref}} + \|P_1 - P_0\| * \hat{u}_{\phi_0}, \tag{3c}$$

$$P_{n-2} = P_{n-1_{ref}} + \|P_{n-1} - P_{n-2}\| * \hat{u}_{\phi_{n-1}}, \tag{3d}$$

$$Lx_k \le x_k \le Ux_k, \text{ for } k = 2, \dots n-3, \tag{3e}$$

$$Ly_k \le y_k \le Uy_k, \text{ for } k = 2, \dots n-3, \tag{3f}$$

$$x_k^{pre} - t \le x_k \le x_k^{pre} + t, \text{ for } k = 2, \dots n-3, \tag{3g}$$

$$y_k^{pre} - t \le y_k \le y_k^{pre} + t, \text{ for } k = 2, \dots n-3, \tag{3h}$$

$$\hat{g}(P_k^{pre}, P_{k-1}^{pre}, P_{k+1}^{pre}, P_k, P_{k-1}, P_{k+1}) - s_k < 0, \quad \text{for } k = 1, \dots n-1, \tag{3i}$$

$$s_k \ge 0, \text{ for } k = 1, \dots, n-2. \tag{3j}$$

State feasible bubble $\mathcal{B}_k$ in constraint (2e) is approximated as an inscribed box with upper limits, Ux_k, Uy_k, and lower limits, Lx_k, Ly_k, in constraints (3e), (3f). Trust region state constraints with respect to previous iteration are shown in constraints (3g), (3h). Nonlinear constraints (2f) are transformed to linearized constraints (3i) around previous iteration path points P_k^{pre} via Euler method. Trust Region method [23] is then applied afterward to guarantee the approximation quality between iterative steps. TrueImprove/ModelImprove in Algorithm 1 is the ratio between true improvement to objective and constraint violation of original problem 1 to those of the sub-problem [17]. We enlarge or shrink trust region size for each sub-problem according to this ratio.

2) Outer Loop for Collision Avoidance: Following the path smoothing in the inner loop, we check whether the generated path trajectory collides with obstacles. The precise shapes of obstacles and the ego vehicle are considered during the collision check. If collision is detected in the kth path point, we shrink the corresponding bubble $\mathcal{B}_k$ size by a ratio $\beta < 1$. The detailed procedure is illustrated in Fig. 3.

Although some prior works [10], [19] avoided using precise collision check to speed up the trajectory generation process, we find it is essential to ensure vehicle's safety operation, especially in some scenarios with narrow spaces, such as pull over and parallel parking. It is also worthy mentioning that, instead of directly anchoring the point back to related reference points as in [16], we shrink the state space around the collision path point iteratively, with the purpose of avoiding over-sacrifice the path smoothness (which is critical for Robotaxi) due to collision avoidance.

B. Piecewise-Jerk Speed Optimization

In this subsection, we introduce the Piecewise-Jerk Speed Optimization (PJSO) method to generate longitudinal speed profile along the path generated from Subsection III-A. As the path generated by the DL-IAPS oftentimes comprises both forward and back vehicle movement, the speed optimization is done separately on each piece assuming the vehicle always comes to a complete stop at the gear shifting position for better driving comfort.

We treat the speed profile optimization problem as longitudinal traversal distance smoothing along a time horizon $T_{horizon}$ discretized by Δt. The decision variables includes $[s_k, \dot{s}_k, \ddot{s}_k]$ for $k = i, \dots, n-1$, where $n = T_{horizon}/\Delta t$, and $s_k, \dot{s}_k, \ddot{s}_k$ are the longitudinal traversal distance, speed and acceleration. We use a cubic polynomial as the state dynamics between $[s_k, \dot{s}_k, \ddot{s}_k]$ and $[s_{k+1}, \dot{s}_{k+1}, \ddot{s}_{k+1}]$, assuming the jerk (i.e.,rate of change of acceleration) is constant from time t_k to t_{k+1} (which is so-called "piecewise" jerk). The dynamics are considered as constraints in the final optimization problem and shown in (4):

$$\begin{aligned} \dot{s}_{k+1} &= \dot{s}_k + \ddot{s}_k \Delta t + \frac{1}{2} \dddot{s}_{k,k+1} \Delta t^2 \\ &= \dot{s}_k + \frac{1}{2} \ddot{s}_k \Delta t + \frac{1}{2} \ddot{s}_{k+1} \Delta t, \end{aligned} \tag{4a}$$

$$\begin{aligned} s_{k+1} &= s_k + \dot{s}_k \Delta t + \frac{1}{2} \ddot{s}_k \Delta t^2 + \frac{1}{6} \dddot{s}_{k,k+1} \Delta t^3, \\ &= s_k + \dot{s}_k \Delta t + \frac{1}{3} \ddot{s}_k \Delta t^2 + \frac{1}{6} \ddot{s}_{k+1} \Delta t^2. \end{aligned} \tag{4b}$$

Such problem formulation makes it flexible to set constraints for feasibility. $s, \dot{s}, \ddot{s}$ and $\dddot{s} = \frac{\ddot{s}_{j+1} - \ddot{s}_j}{\Delta t}$ are set to be constrained by vehicle parameters $\mathcal{S} \subset \mathbb{R}^4$. The curvature-induced speed constraint on $\dot{s}$ can be added to the problem as (5) with the actual path curvature function $\kappa(s)$, but in order to keep it as a quadratic programming form, we approximate the speed constraint induced by path curvature as a linear constraint in (6d), with maximum lateral acceleration $lateral\_a_{\max}$ and maximum curvature $\kappa(s)_{\max}$ along the generated path. This approximated constraint would over limit the speed by using a fixed all-time maximum curvature, but still be a proper one, as the problem setting of the algorithm is not a racing competition but relatively slow free space maneuvering, like parallel parking.

$$\dot{s}_j < \sqrt{a_{lateral\_max}/\kappa(s_j)}, \quad \text{for } j = 0, \ldots, n-1 \tag{5}$$

With the optimization constraints been set up, the optimization step horizon n, which decides the time horizon by $T_{horizon} = n\Delta t$, is initialized in (6f). As n can't be too short to traverse through the entire path, we first estimate its feasible lower bound $n_{\min}$ based on the vehicle dynamics. Given the maximum acceleration $a_{\max}$, maximum speed $v_{\max}$ and the total traverse path distance s_f, we have $n_{\min} = \frac{v_{\max}^2 + s_f a_{\max}}{a_{\max} v_{\max} \Delta t}$, with an infinite jerk assumption so that the vehicle is able to accelerate by $a_{\max}$ to peak speed $v_{\max}$ and decelerates by $-a_{\max}$ to zero speed. Then, with a constrained jerk, the horizon is multiplied by a heuristic expansion ratio r as $n = \lfloor r * n_{\min} \rfloor$, where r is selected in range of $[1.2, 1.5]$. Higher ratio gives more dynamic feasibility for this fixed-distance speed optimization, but an over-estimated r may bring in unnecessary computation time as it increases the dimension of decision variables.

To minimize the traversal time to path end at s_f, we set a cost term $\sum_{k=0}^{n-1}(s_k - s_f)^2$ in objective to penalize the distance gap between every-step state and final state. In addition to that, to balance the driving comfort, penalties on $\ddot{s}$ and $\dddot{s}$ are included.

The complete quadratic programming optimization formulation with weighting hyperparameter w_{s_f}, $w_{\dddot{s}}$ and $w_{\ddot{s}}$ is presented in (6) as:

$$\min_{\mathbf{s},\dot{\mathbf{s}},\ddot{\mathbf{s}}} \mathcal{J}_c\,(\mathbf{s}, \dot{s}, \ddot{s}) = w_{s_f} \sum_{k=0}^{n-1} (s_k - s_f)^2 + w_{\dddot{s}} \sum_{k=0}^{n-2} ((\ddot{s}_{k+1} - \ddot{s}_k)/\Delta t)^2 + w_{\ddot{s}} \sum_{k=0}^{n-1} \ddot{s}_k^2, \tag{6a}$$

subject to:

$$[s_0, \dot{s}_0, \ddot{s}_0] = [0.0, 0.0, 0.0]\,, \tag{6b}$$

$$\left[s_j, \dot{s}_j, \ddot{s}_j, \frac{\ddot{s}_{j+1} - \ddot{s}_j}{\Delta t}\right] \in \mathcal{S}, \tag{6c}$$

$$\dot{s}_j < \sqrt{a_{\text{lateral\_max}}/\kappa(s)_{\max}}, \tag{6d}$$

$$s_k,\ \dot{s}_k \text{ follow the dynamics in Eq. (4)}, \tag{6e}$$

$$\text{for } k = 0, \ldots, n-2,\ j = 0, \ldots, n-1,$$

$$\text{and } n = \left\lfloor r \frac{v_{\max}^2 + s_f a_{\max}}{a_{\max} v_{\max} \Delta t} \right\rfloor. \tag{6f}$$

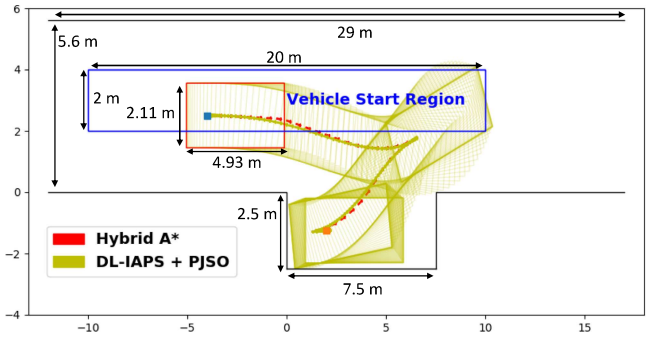


Fig. 4. Illustration of standard parallel parking simulation environment. Simulation vehicle parameters: wheelbase: 2.8 m; kinematic-feasible path curvature (m^{-1}): $[-0.2, 0.2]$; desired speed range (m/s): $[-1, 2]$; acceleration range (m/s^2): $[-1, 1]$; jerk range (m/s^3): $[-1, 1]$; Optimizer set-up: hybrid A* step size(m): 0.5; hybrid A* steering resolution(rad): 0.026; path/speed optimization Δt(sec): 0.1–0.5. One example with Hybrid A* path reference and DL-IAPS smoothing ($\Delta s = 0.1$ m) is presented.

IV. Experimental Setup and Results

In this section, we present both numerical simulations and real-world vehicle testing results on Apollo Autonomous Driving Platform [24], [25], to demonstrate the overall performance of our proposed algorithm.

A. Numerical Simulations: Performance Comparisons

The ideal free-space planning algorithm should be *robust enough* to tackle various tasks (i.e, from arbitrary realistic starting position/pose to desired ending position/pose), and always capable to generate a *collision-free*, *control feasible* (i.e., smooth enough to strictly satisfy the kinematic constraints) and *driving comfortable* (i.e., less jerky) planning trajectory, with *high computation efficiency*. To demonstrate the remarkable performance of our proposed DL-IAPS plus PJSO algorithm according to the above metrics, we evaluate our planner with batch simulation tests and compare it with other aforementioned state-of-the-art planners.

We set up a standard numeric testing environment for parallel parking scenario as Fig. 4. Parallel parking scenario often requires the vehicle to do multiple forward/backward pose adjustments in a narrow space with possibly irregular obstacles/boundaries. Therefore, it is utilized as a good test case to evaluate the free space planner at various extreme situations. The proposed algorithms are implemented in a simulation environment with an i7 processor clocked at 2.6 GHz. The quadratic programming problem in both path and speed optimization are solved by a QP solver, OSQP [26].

Under this test environment, we implement our path/speed decoupled planner together with another decoupled CES planner [10] (note: for test purpose the CES path planner is implemented with the IPOPT solver [27] and combined with our PJSO speed optimizer, since its own source code is not published), and coupled planners including H-OBCA planner [5] and TEB planner [6], as follows:

1) Robustness (involved With Collision Avoidance and Control Feasibility): First, the batch simulation results (consisting

TABLE I
PERFORMANCE SUMMARY ON FAILURE RATE AND COMPUTATION TIME, THROUGH 80 PULL OVER CASES WITH DIFFERENT STARTING POSES

δt	Algorithms	Failure Rate*	Computation Time (sec)** mean	min	max
0.5	H-OBCA	8.75%	0.284	0.108	1.429
	TEB	100.00%	–	–	–
	CES + PJSO	23.75%	1.407	0.062	6.125
	DL-IAPS + PJSO	**7.50%**	**0.088**	0.009	0.390
0.4	H-OBCA	26.25%	0.453	0.163	1.142
	TEB	76.25%	0.454	0.122	0.892
	CES + PJSO	23.75%	1.419	0.063	6.281
	DL-IAPS + PJSO	**7.50%**	**0.098**	0.011	0.434
0.3	H-OBCA	98.75%	0.842	0.842	0.842
	TEB	13.75%	0.802	0.191	1.438
	CES + PJSO	22.50%	1.365	0.070	5.991
	DL-IAPS + PJSO	**6.25%**	**0.102**	0.012	0.433
0.2	H-OBCA	100.0%	–	–	–
	TEB	10.00%	1.330	0.557	1.996
	CES + PJSO	22.50%	1.298	0.072	6.005
	DL-IAPS + PJSO	**6.25%**	**0.101**	0.015	0.435
0.1	H-OBCA	100.0%	–	–	–
	TEB	**0.00%**	2.343	1.579	3.115
	CES + PJSO	22.50%	1.415	0.092	6.179
	DL-IAPS + PJSO	6.25%	**0.127**	0.030	0.458

*Failure rate is the ratio of failed cases (when the algorithm fails to solve the problem) within the total 80 cases with different starting poses. **Computation time (in average through successful cases within 80 cases). Reference path is generated via hybrid A* with extra time cost of 0.4 sec.

of 80 tests with uniformly distributed stating poses within the vehicle start region in Fig. 4) demonstrate the robustness of the planners by means of their failure rates through tests, as shown in Table I. Failed test cases are induced by two reasons: 1) Missing the theoretically-existing but narrow "passable" zone and hence, failing to avoid collision; 2) Failing to meet the vehicle kinematic constraint, which makes the generated path non-executable by vehicle controller when applied in road-test (i.e., control infeasible). In principle, the lower the failure rate is, the more capable the planner is to seek out the control-feasible planning trajectories from challenging starting poses. Table I shows the failure rates among planners at different optimization time resolutions δt (for better execution in road-test, the fine time resolution is preferred and $\delta t = 0.5$ sec is the acceptable upper limit). Benefiting from the accurate vehicle shape and strict curvature constraint modeling, our DL-IAPS plus PJSO planner achieves the lowest failure rate of $\sim$7%; while the CES plus PJSO planner only achieves a $\sim$23% failure rate due to its theoretical curvature constraint violation which is explained in [12]. The TEB algorithm applies approximation on turning radius in optimization formulation, which could cause curvature constraint violation especially when δt is large [6]; when small δt is adopted, the TEB planner reaches very low failure rate but its computation time roars up to an unacceptable level (>2 sec). In the contrary, the H-OBCA planner only maintains a low failure rate with large δt, but fully fails at fine sampling due to the increased decision variables' dimension (more rigorous kinematic constraints and complex collision avoidance constraints), which makes it harder to solve the underlying nonlinear problem.

Fig. 5 shows the control-feasible performance comparisons at one specific starting point. The yellow areas denote the

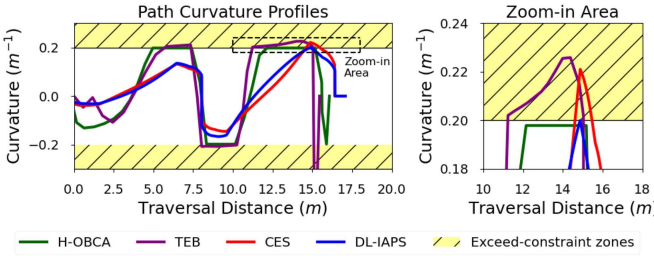


Fig. 5. Path curvature comparison of optimized trajectories by the DL-IAPS, CES, TEB and H-OBCA with starting point at x = 0, y = 3.5 and $\delta t = 0.5$\;s

TABLE II
PERFORMANCE SUMMARY ON DRIVING COMFORT, THROUGH 80 PULL OVER CASES WITH DIFFERENT STARTING POSES

δt	Algorithms	Longitudinal Jerk Exceed Bound*	Max	Mean	Lateral Jerk Exceed Bound	Max	Mean
0.5	H-OBCA	6.40%	2.011	0.325	0.00%	0.245	0.068
	TEB	–	–	–	–	–	–
	CES + PJSO	**0.00%**	1.000	0.252	0.00%	0.284	0.049
	DL-IAPS + PJSO	**0.00%**	1.000	0.252	0.00%	0.302	0.050
0.4	H-OBCA	7.81%	2.307	0.331	0.00%	0.291	0.083
	TEB	1.69%	1.113	0.070	0.00%	0.183	0.028
	CES + PJSO	**0.00%**	1.000	0.241	0.00%	0.267	0.050
	DL-IAPS + PJSO	**0.00%**	1.000	0.242	0.00%	0.286	0.051
0.3	H-OBCA	12.82%	2.726	0.435	0.00%	0.642	0.169
	TEB	1.90%	1.468	0.050	0.00%	0.145	0.016
	CES + PJSO	**0.00%**	1.000	0.224	0.00%	0.249	0.047
	DL-IAPS + PJSO	**0.00%**	1.000	0.226	0.00%	0.271	0.048
0.2	H-OBCA	–	–	–	–	–	–
	TEB	1.09%	2.261	0.044	0.00%	0.109	0.010
	CES + PJSO	**0.00%**	1.000	0.214	0.00%	0.222	0.046
	DL-IAPS + PJSO	**0.00%**	1.000	0.215	0.00%	0.240	0.047
0.1	H-OBCA	–	–	–	–	–	–
	TEB	1.14%	6.121	0.089	0.00%	0.064	0.007
	CES + PJSO	**0.00%**	1.000	0.204	0.00%	0.199	0.045
	DL-IAPS + PJSO	**0.00%**	1.000	0.204	0.00%	0.214	0.046

*Longitudinal and lateral jerk comfort bound are set as 1.0 m/s^3 [22].

forbidden zones in which the path curvatures are beyond the control-feasible thresholds. With our DL-IAPS planner, the path curvature is well constrained by the bound decided by (2); the H-OBCA planner also performs well. However, both the CES and TEB planners exceed the curvature constraints (i.e., even 100% steering cannot meet the path curvature) and eventually lead to the planning failure.

2) Computation Efficiency: Still from Table I, the batch simulation results demonstrate a much better computation efficiency with our algorithm. The total time with our path and speed optimization is only around $90-130$ msec in average because of path inner loop SCP and speed QP formulation, which is acceptable to most real-time applications; however, with highly similar simulation setups, the H-OBCA, TEB and CES algorithms ask for hundreds of to $1000+$ msec running time, which is almost one order of magnitude more than our planner.

3) Driving Comfort: The driving comfort test results are shown in Table II. Due to the lack of the driving-comfort related optimization, both H-OBCA and TEB planners present large longitudinal jerks; while the planners with PJSO always maintain the jerks not higher than the empirical bounds reported

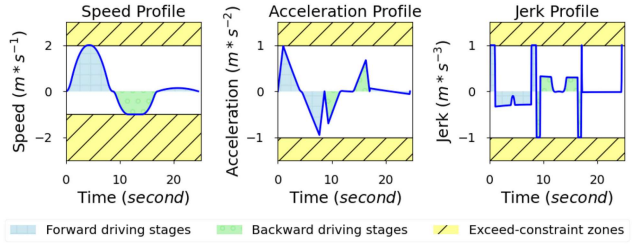
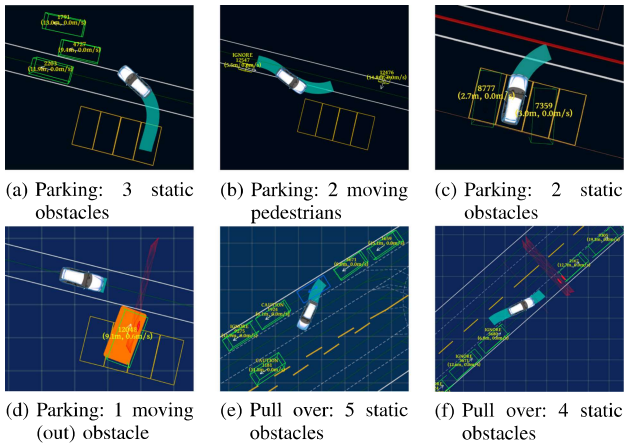


Fig. 6. Optimized speed, acceleration and jerk profiles of PJSO smoother.



(a) Parking: 3 static obstacles (b) Parking: 2 moving pedestrians (c) Parking: 2 static obstacles

(d) Parking: 1 moving (out) obstacle (e) Pull over: 5 static obstacles (f) Pull over: 4 static obstacles

Fig. 7. Various simulation test cases with multiple numbers of boundaries and obstacles (with same vehicle parameters/dynamic constraints as in Fig. 4 except speed profile optimization Δt(s): 0.5).

in [22], to avoid inducing the bad human sensation. To better highlight our algorithm, the PJSO optimized speed, acceleration and jerk profiles are shown in Fig. 6, in which all the profiles are well bounded and a good balance between driving comfort and minimal traversal time is maintained.

B. Numerical Simulations: Expand Performance Validation With Complex Boundaries and Obstacles

To scale the computation efficiency and validate the robustness of the DL-IAPS plus PJSO algorithm to cope with complex obstacles and boundaries, we perform a large-scale, end-to-end simulation on Apollo online simulation platform that carries out totally 208 different free space test scenarios. Fig. 7 shows some of these free space test cases. With the purpose of identifying the sensitivity of computation time to the amount of the boundaries/obstacles, multiple obstacles are intentionally inserted into typical test cases.

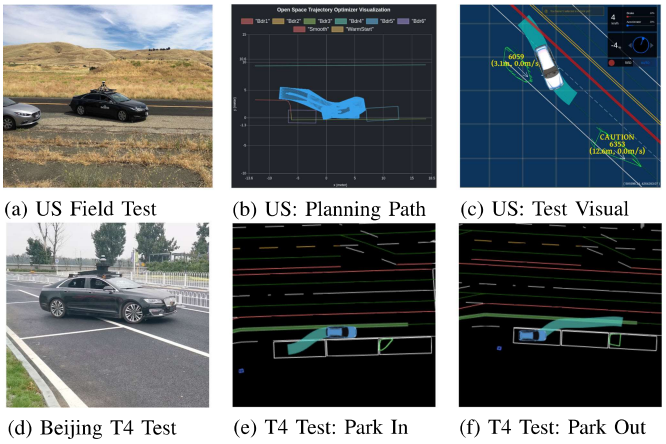
Table III demonstrates that for typical valet parking and pull over scenarios, the total computation time including path smoothing and speed optimization only slightly increases as the numbers of boundaries and obstacles increases and therefore, prevent the computation time from explosively growing induced by extensive obstacles or serpentine boundaries.

C. On-Road Experimental Implementation and Results

To further validate the control feasibility and real-world executability of our DL-IAPS plus PJSO algorithm, we tested it

TABLE III
COMPUTATION TIME OF VALET PARKING AND PULL OVER TEST CASES WITH MULTIPLE BOUNDARIES AND OBSTACLES, IN (SEC)

Cases	Number of (Boundary, Obstacle)	Path Smoothing	Speed Profile	Total Time	Smooth Points	Time per Point
Parking	(6 , 0)	0.087	0.096	**0.183**	162	**0.001130**
	(6 , 1)	0.107	0.081	**0.188**	162	**0.001160**
	(6 , 2)	0.106	0.082	**0.188**	162	**0.001160**
	(6 , 3)	0.106	0.078	**0.184**	162	**0.001136**
Pull Over	(4 , 2)	0.104	0.100	**0.204**	258	**0.000791**
	(4 , 3)	0.103	0.098	**0.201**	253	**0.000794**
	(4 , 4)	0.103	0.099	**0.202**	253	**0.000798**
	(4 , 5)	0.101	0.105	**0.206**	253	**0.000814**



(a) US Field Test (b) US: Planning Path (c) US: Test Visual

(d) Beijing T4 Test (e) T4 Test: Park In (f) T4 Test: Park Out

Fig. 8. US Field Test and China Beijing T4 Test with Data Visualization.

on real autonomous vehicles under complex road environment that asks for subtle maneuvers involving collision avoidance and backward driving. The detailed vehicle controller design is presented in [28]. As shown in Fig. 8, we conducted both US field tests and China Beijing T4 tests. In particular, the China T4 test is currently considered as the most difficult autonomous driving test, due to its strict criteria on position/speed precision, robustness and passing rate. This test is derived from the Chinese official guidance document [29] released in 2018 in which the autonomous driving tests are divided into 5 levels from basic T1 to T5 (with most complex scenarios). Our planner makes a crucial contribution to overcome 10 T4-level free space scenarios, and facilitates Baidu to be the first and so far the only company which passes the entire T4 test in China.

Fig. 8(a)–(c) show the US field test environment, overall optimized planning trajectory, and underway test visualization of the pull over scenario, respectively. Fig. 8(d)–(f) show the China Beijing T4 test environment and underway stage-by-stage test visualization of the zig-zag parallel parking scenario, respectively. Table IV summarizes the experimental performance data in the pull over and parallel parking tests including multiple forward-driving and backward-driving stages. The high-precision planning and control performance is demonstrated by the very low lateral errors and heading angle errors at every stage through the entire test scenario.

TABLE IV
ON-ROAD TEST PERFORMANCE SUMMARY

Tests	Testing Stages	Control Errors at End	
		Lateral in (m)	Heading in (deg)
Beijing T4	1. Approaching	0.043	1.439
	2. Parking In	0.089	0.907
	3. Parking Out	0.054	1.557
	Mean Value	**0.062**	**1.301**
US Field	1. Approaching	0.0238	1.589
	2. Pose Adjustment	0.0256	0.431
	3. Parking In (Backward)	0.0476	4.863
	4. Parking In (Forward)	0.0819	0.939
	Mean Value	**0.0447**	**1.956**

Overall, the experiment results demonstrate that the optimized planning trajectory establishes a kinematic-smooth and kinodynamic-feasible reference for the control module, so as to enable an accurate autonomous vehicle control.

V. CONCLUSION

In this letter, we present a novel path/speed decoupled trajectory optimization algorithm which has the advantages of real-time computational efficiency, robust and precise collision avoidance, strict path curvature constraint and comfortable minimum-time speed profile. Through the exhaustive numeric simulations (80 standardized test cases and 208 simulation scenarios extracted from real world) and more than 400 hours real-world tests including the US and China Beijing T4 test, we have proved that our algorithm maintains the lowest failure rate ($\sim$7%) across (almost) all planning time resolutions (0.1$-$0.5 sec), 10x faster solving time, 100% guaranteed driving-comfort compared with some prior works, without inducing significant time increase as boundaries or obstacles scale up.

We plan to extend our work to other higher speed complex applications including: 1) avoid brushing against opposite vehicles along narrow roads and 2) three-point turn in the dead-end alleyway. One drawback is that, although being well within the lateral jerk comfort bounds, our algorithm has higher lateral jerks compared with others and therefore has a larger chance of hitting these bounds when being extended to higher speed applications. We will focus on lowering this bounds hitting chance as speed increases in future research.

REFERENCES

[1] 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," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[2] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, pp. 187–210, 2018.

[3] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenèt frame," in *IEEE Int. Conf. Robot. Automat.*, 2010, pp. 987–993.

[4] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, 2018, Art. no. 3170.

[5] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *Proc. IEEE Conf. Decis. Control.*, 2018, pp. 4327–4332.

[6] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 5681–5686.

[7] J. Deray, B. Magyar, J. Solà, and J. Andrade-Cetto, "Timed-elastic smooth curve optimization for mobile-base motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 3143–3149.

[8] J. S. Smith, R. Xu, and P. Vela, "egoteb: Egocentric, perception space navigation using timed-elastic-bands," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2703–2709.

[9] T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl, "An nmpc approach using convex inner approximations for online motion planning with guaranteed collision avoidance," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3574–3580.

[10] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. 54th IEEE Conf. Decis. Control.*, 2015, pp. 835–842.

[11] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," in *Proc. IEEE Conf. Decis. Control.*, 2018, pp. 4327–4332.

[12] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.

[13] P. Wu, F. Xiao, C. Sha, H. Huang, and L. Sun, "Trajectory optimization for uavs efficient charging in wireless rechargeable sensor networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4207–4220, Apr. 2020.

[14] R. Chai, A. Savvaris, A. Tsourdos, Y. Xia, and S. Chai, "Solving multiobjective constrained trajectory optimization problem by an extended evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1630–1643, Apr. 2020.

[15] Q. Li, H. Ju, P. Xiao, F. Chen, and F. Lin, "Optimal trajectory optimization of 7r robot for space maintenance operation," *IEEE Access*, early access, Jul. 20, 2020.

[16] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010.

[17] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.

[18] B. Alrifaee, J. Maczijewski, and D. Abel, "Sequential convex programming mpc for dynamic vehicle collision avoidance," in *IEEE Conf. Control Technol. Appl.*, 2017, pp. 2202–2207.

[19] J. Chen, C. Liu, and M. Tomizuka, "Foad: Fast optimization-based autonomous driving motion planner," in *Proc. IEEE Amer. Control Conf.*, 2018, pp. 4725–4732.

[20] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *Int. J. Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[21] 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," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 613–626, Feb. 2018.

[22] I. Bae, J. Y. Moon and S. Kim, "Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles," 2020, *arXiv:2001.03908*.

[23] M. Huang and D. Pu, "A trust-region sqp method without a penalty or a filter for nonlinear programming," *J. Comput. Appl. Math.*, vol. 281, pp. 107–119, 2015.

[24] J. Xu *et al.*, "An automated learning-based procedure for large-scale vehicle dynamics modeling on baidu apollo platform," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5049–5056.

[25] K. Xu, X. Xiao, J. Miao, and Q. Luo, "Data driven prediction architecture for autonomous driving and its application on apollo platform," 2020, *arXiv:2006.06715*.

[26] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.

[27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[28] Y. Wang *et al.*, "A learning-based tune-free control framework for large scale autonomous driving system deployment," 2020, *arXiv:2011.04250*.

[29] "Self-driving car road tests in china – how to get on the road to progress," [Online]. Available: https://www.kwm.com/en/cn/knowledge/insights/self-driving-car-road-tests-in-china-20170928, 2017.