# TDR-OBCA: A Reliable Planner for Autonomous Driving in Free-Space Environment

**Preprint** · September 2020



**9 authors**, including:

Yu Wang
Baidu USA
**21** PUBLICATIONS **200** CITATIONS
SEE PROFILE

Shiyu Song
Central South University
**8** PUBLICATIONS **96** CITATIONS
SEE PROFILE

Qi Luo
Baidu Online Network Technology
**18** PUBLICATIONS **152** CITATIONS
SEE PROFILE

# TDR-OBCA: A Reliable Planner for Autonomous Driving in Free-Space Environment

Runxin He[1], Jinyun Zhou[1], Shu Jiang, Yu Wang, Jiaming Tao, Shiyu Song, Jiangtao Hu, Jinghao Miao, Qi Luo[2]

***Abstract*— This paper presents an optimization-based collision avoidance trajectory generation method for autonomous driving in free-space environments, with enhanced robustness, driving comfort and efficiency. Starting from the hybrid optimization-based framework, we introduces two warm start methods, temporal and dual variable warm starts, to improve the efficiency. We also reformulates the problem to improve the robustness and efficiency. We name this new algorithm TDR-OBCA. With these changes, compared with original hybrid optimization we achieve a $96.67\%$ failure rate decrease with respect to initial conditions, $13.53\%$ increase in driving comforts and $3.33\%$ to $44.82\%$ increase in planner efficiency as obstacles number scales. We validate our results in hundreds of simulation scenarios and hundreds of hours of public road tests in both U.S. and China. Our source code is available at `https://github.com/ApolloAuto/apollo`.**

## I. Introduction

Collision-free trajectory planning for nonholonomic system is vital to robotic applications, such as autonomous driving [1] [2]. One of its application areas is the autonomous driving in free-space scenarios, where the ego vehicle's starting points or destinations are off-road. Parking and hailing scenarios belong to these free-space scenarios, because the ego vehicle either goes to or starts from an off-road spot, e.g., parking spot. In free-space scenarios, an autonomous vehicle may need driving forward or backward via gear shifting to maneuver itself towards its destination. Trajectories generated by Hybrid A* [3] or other sampling/searching based methods [4] usually fail to meet the smoothness requirement for autonomous driving. Thus, an optimization-based method is required to generate feasible trajectories in order to decrease autonomous driving failures [5].

Though non-holonomic constraints, such as vehicle dynamics, can be well incorporated into Model Predictive Control (MPC) formulations, collision avoidance constraint still remains a challenge due to its non-convexity and non-differentiability [6]. To simplify the collision constraint, most trajectory planning studies approximate the ego vehicle and obstacles into disks. However, this approximation reduces problem's geometrical feasible space, which may lead to failures in scenarios with irregularly placed obstacles [7]. If obstacles and the ego vehicle are represented with full dimensional polygons, the collision avoidance MPC are derived into a Mixed-Integer Programming (MIP) problem [8]. Though a variety of numerical algorithms are available to solve MIP such as dynamical programming (DP) method [9] and the branch and bound method [10], due to its completeness, the computation time may fail to real-time requirement when the number of nearby obstacles is large [11]. To remove the discrete integer variables in the MIP, Zhang [12] presented Hybrid Optimization-based Collision Avoidance (H-OBCA) algorithm. The distances between the ego vehicle and obstacles are reformulated into their dual expressions, and the trajectory planning problem is transformed into a large-scale MPC problem with only continuous variables.

Inspired by H-OBCA, we present Temporal and Dual warm starts with Reformulated Optimization-based Collision Avoidance (TDR-OBCA) algorithm with improved robustness, driving comfort and efficiency, and integrate it with Apollo Autonomous Driving Platform [13]. Our contributions are:

1) **Robustness**: With two extra warm starts and a reformulation to cost and constraints in final MPC, we reduce failure rate by $96.67\%$, from $37.5\%$ to $1.25\%$ with respect to different initial spots in Section IV-A.1;
2) **Driving Comfort**: With additional smooth penalty terms in the cost function, we show in Section IV-A.2 the steering control outputs have reduced more than $13.53\%$.
3) **Efficiency**: We also show in Section IV-A.2 that TDR-OBCA's solving time maintains a $3.44\%$ to $81.25\%$ decrease compared with original methods as obstacles number scales in different scenarios (including both the driving region boundaries and surrounding vehicles and pedestrians), with possibility of further time reduction if we replace IPOPT [14] with powerful solvers specially for MPC, such as GRAMPC [15]. Thus make it applicable to free-space scenarios of different complexities, e.g. parking, pull over, hailing and even three point U-turn in the future.
4) **Real Road Tests**: We show in IV-B that robustness, efficiency and driving comfort of TDR-OBCA is verified on both 282 simulation cases and hundreds of hours of public road tests in both China and USA.

This paper is organized as followed: the problem statement and core algorithms are in Section II and III respectively; the results of both simulations and on-road tests are presented

[1] Authors contributed equally to this paper

[2] Corresponding author

All Authors are with Baidu USA LLC, 250 E Caribbean Drive, Sunnyvale, CA 94089 runxinhe@baidu.com, jinyunzhou@baidu.com, luoqi06@baidu.com

in Section IV.

## II. Problems Statement

TRD-OBCA aims to generate a smooth collision-free trajectory for autonomous vehicles in free spaces, i.e., parking lots or off-road regions. We formulate the collision-free and smoothness requirements as constrains of a MPC optimization problem, which is similar to H-OBCA algorithm [12].

At time $k$, the ego vehicle's state vector is $x(k) = [x_x(k), x_y(k), x_v(k), x_\varphi(k)]^T \in \mathbb{R}^4$, where $x_x(k)$ and $x_y(k)$ is vehicle's latitude and longitude position, $x_v(k)$ is the vehicle's velocity and $x_\varphi(k)$ is heading (yaw) angle in radius. The control command from steering and brake/throttle is formulated as $u(k) = [\delta(k), a(k)]^T \in \mathbb{R}^2$, where $\delta(k)$ is the steering and $a(k)$ is the acceleration. The ego vehicle's dynamic system is modeled by the kinematic of a bicycle model and defined as,

$$x(k) = f(x(k-1), u(k-1)). \tag{1}$$

Compared to H-OBCA, we have two major changes to the problem's formulation. The first is, instead of using variant time steps, the time horizon $[0, T]$ is evenly discretized into $K$ steps. The time horizon $T$ is derived based on the estimated trajectory distance and vehicle dynamic feasibility. The detailed implementation is introduced in Section III-A. The other is that we reformulate the constraints and cost function to reduce control efforts and increase trajectory smoothness, which is introduced in Section III-C. Here we first present the formulation to H-OBCA, which transforms the collision avoidance problem form a MIP to a continous MPC with nonlinear constrains,

$$\begin{aligned}
&\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\mu},\boldsymbol{\lambda}} \sum_{k=1}^{K} l\ (x(k), u(k-1)) \\
&\text{subject to:} \\
&x(0) = x_0, \\
&x(K) = x_F, \\
&x(k) = f(x(k-1), u(k-1)), \\
&h(x(k), u(k)) \leq 0, \\
&-g^T \mu_m(k) + (A_m t(x(k)) - b_m)^T \lambda_m(k) > d_{min}, \\
&G^T \mu_m(k) + R(x(k))^T {A_m}^T \lambda_m(k) = 0, \\
&\|{A_m}^T \lambda_m(k)\|_2 \leq 1, \lambda_m(k) \succeq_\kappa 0, \mu_m(k) \succeq_\kappa 0, \\
&\text{for } k = 1, \dots K,\ m = 1, \dots, M.
\end{aligned} \tag{2}$$

Where $l$ is the cost function, its detailed formulation and our corresponding reformulation are in Section III-C. The optimization is performed over $\boldsymbol{x} = [x(0), \dots, x(K)]$, $\boldsymbol{u} = [u(0), \dots, u(K-1)]$ and dual variables to the dual formulas of distances between ego vehicle and obstacles [12], $\boldsymbol{\mu} = [\mu_1(1), \dots, \mu_1(K), \mu_2(1) \dots, \mu_M(K)]$ and $\boldsymbol{\lambda} = [\lambda_1(1), \dots, \lambda_1(K), \lambda_2(1) \dots, \lambda_M(K)]$. $x_0$ and $x_F$ are ego vehicle's start and end state respectively. Constraints $h(x(k), u(k))$ express vehicle dynamic limitations, for example the box constraints to the steering angle and the acceleration. The vehicle geometry is approximated as a polygon with line segments, described by matrix $G$ and vector $g$. Obstacles are described as combinations of polygons as well, the $m$-th polygon is represented as a matrix $A_m$ and the vector $b_m$. A complicated obstacle may be expressed by several polygons with more line segments than a simple one. At time step $k$, the set distance between the ego vehicle and $m$-th obstacle polygon should be larger than $d_{min}$ for safety. Detailed descriptions to expressions of obstacles' geometry and safety distance are referred to H-OBCA [12].

## III. Core Algorithms

Due to its high non-convexity and non-linearity, solving problem (2) requires heavy computation efforts. Even in simulation, it may take minutes to generate a trajectory. We introduced the temporal profile and dual variable warm starts together with hybrid A* to speed up the convergence of final optimization problem. We also reformulate the MPC problem with new penalty terms to guarantee the smoothness of the trajectory and soft constraints to end state spot for robustness. The architecture of TDR-OBCA is shown in Fig. 1 and the highlighted blocks are the core algorithms, which are introduced in this section.

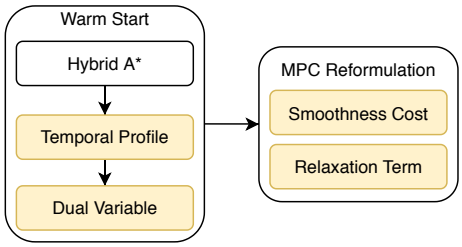



Fig. 1. TDR-OBCA Architecture.

### A. Temporal profile warm start

Hybrid A* is used to heuristically generate $x_x$, $x_y$ and $x_{phi}$ as part of a collision free trajectory to problem (2). Then, a temporal profile method estimates $x_v$ and $x_\varphi$ based on the path result from Hybrid A*. However, the temporal profile results generated by directly differentiating path points from Hybrid A* is not smooth enough for control layer. In TDR-OBCA, an optimization-based temporal profile method is introduced to improve the dynamic feasibility and smoothness of the trajectory, thus to reduce the failure rate.

The detail of the temporal profile method in TDR-OBCA has been discussed in another work of one of the authors [16]. In this process, first the Hybrid A* output path is partitioned by different vehicle gear locations, i.e., forward gear or reverse gear. Then, on each path, we optimize state space of longitudinal traverse distance and its derivatives with respect to time. In our application, the total time horizon is derived based on the ego vehicle's minimal traverse time and its dynamic feasibility. Given the ego vehicle's maximum acceleration $a_{max}$, maximum speed $v_{max}$, and maximum

deceleration $-a_{max}$ on a traverse distance of $s$, total horizon is chosen as

$$T = r\frac{v_{max}^2 + sa_{max}}{a_{max}v_{max}}, \tag{3}$$

where $r \in [1.2, 1.5]$ is a heuristic ratio.

The modified optimization problem for temporal profile is a quadratic programming (QP) problem, which can be efficiently solved by high performance numerical solvers [17].

### B. Dual variable warm start

In TDR-OBCA, we use dual variable initialization to provide better initial values to $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ for final MPC problem, such as (2), to achieve a fast convergence to the optimal value. Because the cost function in problem (2) does not contain $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$, its direct optimization-based warm start problem is not well defined. To address this issue, we first introduce a slack variable, $\boldsymbol{d}$, to the dual variable warm start problem. We define $\boldsymbol{d} = \big[d_1(1), \ldots, d_1(K), d_2(1) \ldots, d_M(K)\big]$, where $d_m(k)$ represents the negative value of a safety distance between the ego vehicle and the $m$-th obstacle polygon at time $k$. Thus, the dual warm up problem is reformulated as

$$\begin{aligned}
&\min_{\boldsymbol{\mu},\boldsymbol{\lambda},\boldsymbol{d}} \sum_{m=1}^{M}\sum_{k=1}^{K} d_m(k) \\
&\text{subject to:} \\
&-g^T\mu_m(k) + \big(A_m t(\tilde{x}^*(k)) - b_m\big)^T\lambda_m(k) \\
&\qquad + d_m(k) = 0, \\
&G^T\mu_m(k) + R(\tilde{x}^*(k))^T {A_m}^T\lambda_m(k) = 0, \\
&\big\|{A_m}^T\lambda_m(k)\big\|_2 \le 1, \\
&\lambda_m(k) \succeq 0, \mu_m(k) \succeq 0, d_m(k) < 0, \\
&\qquad \text{for } k = 1, \ldots K,\ m = 1, \ldots, M,
\end{aligned} \tag{4}$$

where $\tilde{x}^*$ is the vehicle states from Hybrid A* .

Though problem (4) is a convex Quadratic Constrained Quadratic Programming (QCQP), to improve the computation efficiency, we transform the problem (4) into a QP problem as problem (5), which can be solved parallelly using distributed computing algorithms, such as Operator Splitting and ADMM [17].

$$\begin{aligned}
&\min_{\boldsymbol{\mu},\boldsymbol{\lambda},\boldsymbol{d}} \frac{1}{\beta}\sum_{m=1}^{M}\big\|{A_m}^T\lambda_m(k)\big\|_2^2 + \sum_{m=1}^{M}\sum_{k=1}^{K} d_m(k) \\
&\text{subject to:} \\
&-g^T\mu_m(k) + \big(A_m t(\tilde{x}^*(k)) - b_m\big)^T\lambda_m(k) \\
&\qquad + d_m(k) = 0, \\
&G^T\mu_m(k) + R(\tilde{x}^*(k))^T {A_m}^T\lambda_m(k) = 0, \\
&\lambda_m(k) \succeq 0, \mu_m(k) \succeq 0, d_m(k) < 0, \\
&\qquad \text{for } k = 1, \ldots K,\ m = 1, \ldots, M.
\end{aligned} \tag{5}$$

Now we prove that the optimal solution to the relaxed problem (5) leads to an sub-optimal point to problem (4) and show one corresponding transformation formula.

**Proposition III.1.** *The feasible set of optimization problem* (4) *is a subset of* (5)*. Thus, given* $\{\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{d}}\}$ *as the optimum solution to* (4)*,* $\{\boldsymbol{\mu}^o, \boldsymbol{\lambda}^o, \boldsymbol{d}^o\}$ *as the optimum solution to* (5)*, then*

$$\begin{aligned}
&\frac{1}{\beta}\sum_{m=1}^{M}\big\|{A_m}^T\lambda_m^o(k)\big\|_2^2 + \sum_{m=1}^{M}\sum_{k=1}^{K} d_m^o(k) \le \\
&\frac{1}{\beta}\sum_{m=1}^{M}\Big\|{A_m}^T\tilde{\lambda}_m(k)\Big\|_2^2 + \sum_{m=1}^{M}\sum_{k=1}^{K} \tilde{d}_m(k).
\end{aligned} \tag{6}$$

*Moreover, the optimization solution to problem* (5) *leads to a feasible point to optimization problem* (4)*.*

*Proof.* Since optimization problem in (4) has one more type of constraints than problem (5), it is trivial to show $\{\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{d}}\}$ also satisfies the constraints to problem (5), thus inequality (6) satisfies as well.

We prove the left statement by constructing such a transformation. Assume $\{\boldsymbol{\mu}^o, \boldsymbol{\lambda}^o, \boldsymbol{d}^o\}$ is the optimal solution to (5), construct a new combination of dual variables, $\{\boldsymbol{\mu}^m, \boldsymbol{\lambda}^m, \boldsymbol{d}^m\}$, as for $k = 1, \ldots K$, and $m = 1, \ldots, M$, if $\big\|{A_m}^T\lambda_m^o(k)\big\|_2 > 1$, $\lambda_m^m(k) = \frac{1}{\big\|{A_m}^T\lambda_m^o(k)\big\|}\lambda_m^o(k)$, $\mu_m^m(k) = \frac{1}{\big\|{A_m}^T\lambda_m^o(k)\big\|}\mu_m^o(k)$, $d_m^m(k) = \frac{1}{\big\|{A_m}^T\lambda_m^o(k)\big\|}d_m^o(k)$; otherwise, $\lambda_m^m(k) = \lambda_m^o(k)$, $\mu_m^m(k) = \mu_m^o(k)$, $d_m^m(k) = d_m^o(k)$.

By the scaling transfer above, we keep $\big\|{A_m}^T\lambda_m^m(k)\big\|_2$ smaller than 1 and satisfying all the other constraints in (5), thus $\{\boldsymbol{\mu}^m, \boldsymbol{\lambda}^m, \boldsymbol{d}^m\}$ satisfies the constraints in (4). □ □

Moreover, since the above scaling is strictly larger than 1, it is able to show the further relation between optimum solutions to problem (4) and problem (5) as following,

**Proposition III.2.** *Given* $\{\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{d}}\}$ *as the optimum solution to* (4)*,* $\{\boldsymbol{\mu}^m, \boldsymbol{\lambda}^m, \boldsymbol{d}^m\}$ *as the transfer constructed in Proposition III.1 to the optimum solution to* (5)*,*

$$\begin{aligned}
&\sum_{m=1}^{M}\sum_{k=1}^{K} \tilde{d}_m(k) \le \sum_{m=1}^{M}\sum_{k=1}^{K} d_m^m(k) \le \\
&\frac{1}{\beta}\sum_{m=1}^{M}\big\|{A_m}^T\lambda_m^m(k)\big\|_2^2 + \sum_{m=1}^{M}\sum_{k=1}^{K} d_m^m(k).
\end{aligned}$$

Propositions III.1 and III.2 show the relation between the dual warm up QCQP (4) and QP (5). From the aspect of numerical computation, we apply the QP problem (5) for dual warm up since it is more efficient to solve.

### C. MPC problem reformulation

To increase trajectory smoothness and reduce control efforts, we first define the term, $l(\cdot)$, in problem (2) as

$$\begin{aligned}
&\alpha_x\big\|x(k)\big\|_2^2 + \alpha_{x'}\big\|x(k) - x(k-1)\big\|_2^2 \\
&\quad + \alpha_u\big\|u(k-1)\big\|_2^2 + \alpha_{\tilde{u}}\big\|u(k-1) - \tilde{u}(k-1)\big\|_2^2,
\end{aligned} \tag{7}$$

where $\alpha_x$, $\alpha_{x'}$, $\alpha_u$ and $\alpha_{\tilde{u}}$ are corresponding hyperparameters. In (7), the first two terms measure the trajectory's first and second order smoothness respectively, the third term

measures control energy usage and the fourth term measures differences between two consecutive control decisions, where $\tilde{u}$ represents the previous control decision at the corresponding time step. Note that the fourth term is crucial to road-test where real vehicle is involved. The control actuators, i.e., steering, braking and throttle, has limited reaction speed. Large fluctuations between consecutive control decisions may make the actuator fail to track the desired control commands.

This basic cost function in (2) is further modified in TDR-OBCA to overcome two major practical issues: 1) $d_{min}$ in equation is a hyper-parameter for minimum safety distance, but it is hard to tune for general scenarios; 2) both initial and end state constraints make the nonlinear optimization solver slow and, in extreme case, hard to find a feasible solution.

To address these issues, we introduce two major modifications to the MPC problem (2). First, we introduce a collection of slack variables, $\boldsymbol{d}$, which is defined in section III-B to the MPC problem; second the end state constraints are relaxed to soft constraints inside the cost function. The new cost function is

$$\begin{aligned} \mathcal{J}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}\right) &= \sum_{k=1}^{K} l\left(x(k),u(k-1)\right) \\ &+ \alpha_e \left\|x(K)-x_F\right\|_2^2 + \beta \sum_{m=1}^{M}\sum_{k=1}^{K} d_m(k), \end{aligned} \tag{8}$$

where $\alpha_e > 0$ is the hyper-parameter to minimize the final state's position to the target, and $\beta > 0$ is the hyper-parameter to those cost terms which maximize the total safety distances between vehicle and obstacles. The reformulated MPC problem with slack variables is formulated as,

$$\begin{aligned} &\min_{\boldsymbol{x},\ \boldsymbol{u},\ \boldsymbol{d},\ \boldsymbol{\mu},\ \boldsymbol{\lambda}} \mathcal{J}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}\right) \\ &\text{subject to:} \\ &\quad x(0) = x_0, \\ &\quad x(k+1) = f\left(x(k),u(k)\right), \\ &\quad h\left(x(k),u(k)\right) \le 0, \\ &\quad -g^T\mu_m(k) + \left(A_m t(x(k)) - b_m\right)^T \lambda_m(k) \\ &\qquad + d_m(k) = 0, \\ &\quad G^T\mu_m(k) + R\left(x(k)\right)^T {A_m}^T \lambda_m(k) = 0, \\ &\quad \left\|{A_m}^T\lambda_m(k)\right\|_2 \le 1, \\ &\quad \lambda_m(k) \succeq 0, \mu_m(k) \succeq 0, d_m(k) < 0, \\ &\qquad \text{for } k = 1,\dots K,\ m = 1,\dots,M, \end{aligned} \tag{9}$$

where the notations to $d_m(k)$ and $\boldsymbol{d}$ follow the problem (4).

## IV. Experiment Results

TDR-OBCA algorithm is validated in both simulations and real world road tests. In subsection IV-A, we start showing TDR-OBCA's robustness over different starting positions. Then, we further verify TDR-OBCA's robustness and efficiency in end-to-end scenario-based simulations. In subsection IV-B, TDR-OBCA is deployed on real autonomous vehicles, which proves the trajectories provided by TDR-OBCA have control-level smoothness as well as its efficiency and robustness in real world.

### A. Simulations

In this subsection, we present two categories of simulations.

*1) Robustness to various starting positions:* We design a no-obstacle valet parking scenario with regular curb boundaries to show the robustness of TDR-OBCA over its competitive algorithms. The simulation setups and parameters are shown in Fig. 2.

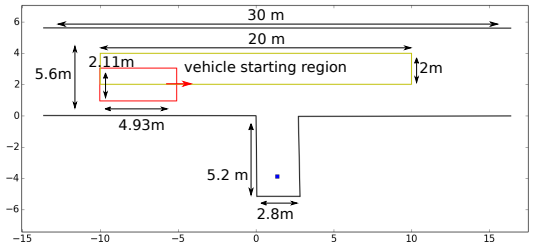



Fig. 2. Robustness experiment settings. Ego vehicle's starting positions are within the yellow box. The red box is the ego vehicle at one starting position. The vehicle is 1769 kg in weight with a 2.8 $m$ wheelbase. Its steering range is $[-0.5, 0.5]$ rad and steering rate is $[-0.5, 0.5]$ rad/$s$. The acceleration range is set to $[-1, 1]$ $m/s^2$ and the speed range is $[-1, 2]$ $m/s$.

The simulation includes 80 cases identified by ego vehicle's different starting positions. The starting positions are evenly sampled on a grid of $x \in [-10, 10]$ $m$ with interval 1.0 $m$ and $y \in [2, 4]$ $m$ with interval 0.5 $m$. The starting heading angle is set to be zero, facing right hand side in Fig. 2. The ending parking spot remains the same for all cases. The optimization problems are implemented and simulated with a i7 processor clocked at 2.6 GHz.

TABLE I

Robustness statistics for cases in Fig. 2. The failure rate is calculated by the number of cases when the algorithm fails to solve the problem divided by the total case number.

| Algorithm | Failure Rate | **Reduced Rate** |
|---|---|---|
| H-OBCA | 37.5% | \ |
| TD-OBCA | 12.5% | **66.67%** |
| TDR-OBCA | 1.25% | **96.67%** |

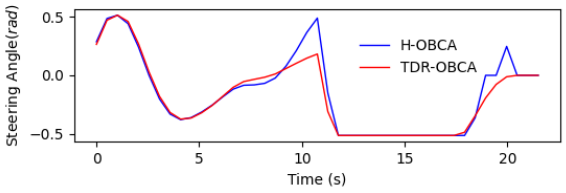
To show robustness, in Table I we compare failure rates of three algorithms. They are: 1) H-OBCA as the benchmark; ii) TD-OBAC algorithm with TDR-OBCA's warm starts proposed in Section III-A and III-B; iii) complete TDR-OBCA algorithm. The failure rate drops from benchmark's 37.5% to 1.25% by applying TDR-OBCA, where the rest of failures are due to exceeding the limit of vehicle dynamics.

Besides robustness, we also compare the optimal control output from H-OBCA and TDR-OBCA. Fig.3 shows the steering output from H-OBCA and TDR-OBCA from one of

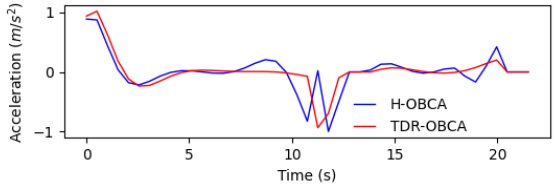
the 80 cases as a typical example. The y-axis is the steering angle in rad. The steering angle output from TDR-OBCA (the red line) has less sharp turns compared to that from H-OBCA (the blue line). Similarly, in the plots to acceleration and jerk, which is the rate of change of acceleration with time, TDR-OBCA also shows more smoothness compared to the trajectory provided by H-OBCA.

As a supplement to Fig. 3, we compare the minimum, maximum and mean control outputs along with jerks of these two algorithms in Table II. On average, TDR-OBCA reduced $13.53\%$ steering output, $1.42\%$ acceleration and $3.34\%$ jerk compared with H-OBCA. That means the steering commands generated by TDR-OBCA is smoother, the accelerations and jerks for TDR-OBCA are comparable to H-OBCA.

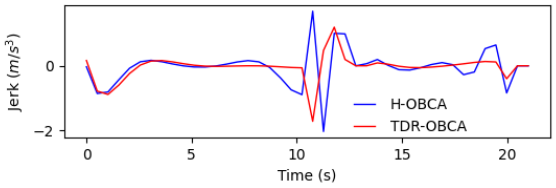
*2) End-to-end scenario-based simulations:* In this subsection, we demonstrate TDR-OBCA performance in end-to-end simulations.




(a) Steering comparison.




(b) Acceleration comparison




(c) Jerk comparison

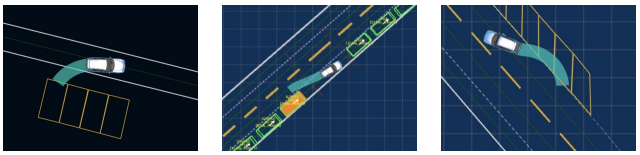
Fig. 3. Control outputs to H-OBCA and TDR-OBCA.

TABLE II

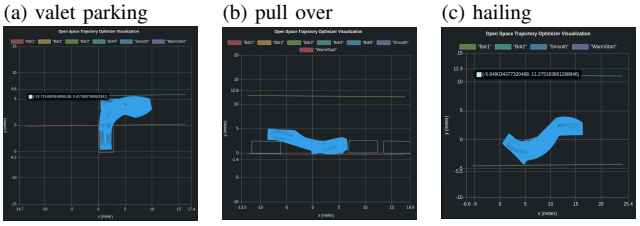
TRAJECTORY EVALUATION RESULTS OVER THE SUCCESSFUL CASES.

| | | **H-OBCA** | **TDR-OBCA** |
|---|---|---|---|
| Steering Angle ($rad$) | Mean | 0.2048 | **0.1771** |
| | Max | 3.2762 | 3.2762 |
| | Min | $-3.8644$ | $-3.8644$ |
| Acceleration ($m/s^2$) | Mean | 0.3392 | **0.3344** |
| | Max | 5.3351 | 5.3351 |
| | Min | $-0.5127$ | $-0.5127$ |
| Jerk ($m/s^3$) | Mean | 0.2663 | **0.2574** |
| | Max | 10.0583 | 10.6021 |
| | Min | $-10.2744$ | $-10.2744$ |

We introduce different types of obstacles and application scenarios (i.e., starting and ending positions). Among hundreds of cases presented at `https://azure.apollo.auto`, we chose three typical scenario types, which are valet parking, parallel parking and hailing. TDR-OBCA successfully generates trajectories for all these scenarios as shown in Fig. 4, which further verifies its robustness. With different environments, initial and destination spots, TDR-OBCA is always able to provide collision free trajectories.



(a) valet parking (b) pull over (c) hailing



(d) valet parking trajectory (e) pull over trajectory (f) hailing trajectory

Fig. 4. Planning trajectories in simulation including scenarios: 1) valet parking: 4a, 4d; 2) pull over: 4b, 4e; 3) hailing: 4c, 4e.

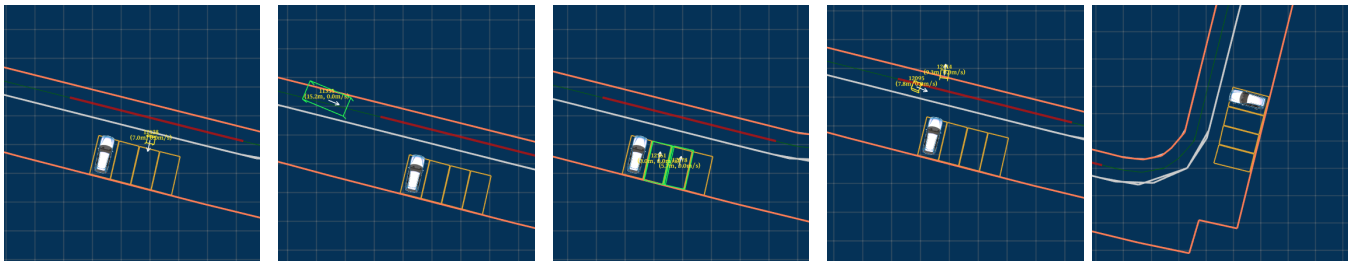
For valet parking scenario cases in Fig. 4, their settings are identified by different curb shapes and surrounding obstacles as shown in Fig. 5. We classify obstacles to two types, Type A obstacles are road boundaries, and Type B obstacles are vehicles or pedestrians. H-OBCA and TDR-OBCA's computation time and the line segment numbers of both type obstacles in each simulation are listed in Table III. Based on this table, when simulation becomes complicated from case $(a)$ to $(e)$, the total time cost by TDR-OBCA, $t_{t,TDR}$, is not increased as fast as H-OBCA, $t_{t,H}$, which validates TDR-OBCA's computation efficiency.

TABLE III

VALET PARKING, COMPUTATION TIME SCALES W.R.T. COMPLEX SIMULATIONS IN FIG. 5, WHERE $N_{OA}$ IS THE NUMBER OF TYPE A OBSTACLES, I.E., ROI BOUNDARY LINE SEGMENTS, AND $N_{OB}$ IS THE NUMBER OF TYPE B OBSTACLES, I.E., VEHICLES' OR PEDESTRIANS' BOUNDARY LINE SEGMENTS. $t_{f,H}$ AND $t_{f,TDR}$ ARE THE MEAN TIME TO GENERATE EACH PLANNING FRAME BY H-OBCA AND TDR-OBCA RESPECTIVELY. $t_{t,H}$ AND $t_{t,TDR}$ ARE THE TOTAL TRAJECTORY GENERATION TIME BY H-OBCA AND TDR-OBCA RESPECTIVELY. ALL THE TIME ARE MEASURED IN SECOND.

| Case ID | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| $N_{OA}$ | 5 | 5 | 6 | 6 | 9 |
| $N_{OB}$ | 4 | 4 | 8 | 8 | 0 |
| $t_{f,H}$ (s) | N.A. | 0.029 | N.A. | 0.021 | N.A. |
| $t_{f,TDR}$ (s) | 0.017 | 0.016 | 0.022 | 0.019 | 0.015 |
| **Improved Rate** | N.A. | **44.82%** | N.A. | **9.52%** | N.A. |
| $t_{t,H}$ (s) | N.A. | 1.80 | N.A. | 2.52 | N.A. |
| $t_{t,TDR}$ (s) | 1.08 | 1.74 | 1.97 | 1.61 | 2.29 |
| **Improved Rate** | N.A. | **3.33%** | N.A. | **36.11%** | N.A. |

Fig. 5 and Table III show TDR-OBCA's robustness and ef-



(a) Pedestrian Obstacle (b) Vehicle Obstacle (c) Two Vehicle Obstacles (d) Two Pedestrian Obstacles (e) Curved RoI Boundary

Fig. 5. Settings for end-to-end valet parking simulation cases in Table III

.

ficiency in two aspects, i) TDR-OBCA is able to handle cases where H-OBCA fails; 2) the TDR-OBCA's computation time is less than H-OBCA. Moreover, figures in Fig. 4 show TDR-OBCA's robustness in different application scenarios, such as valet parking, pull over and hailing.

*3) Simulation Summary:* TDR-OBCA is robust compared to competitive algorithms. Together with regular planner, it is able to handle different scenarios with various obstacles, which is essential for autonomous driving. Furthermore, in the next subsection with real road test results, we show the trajectories generated by TDR-OBCA meet control-level smoothness for autonomous driving.

### B. Real world road tests

We have integrated TDR-OBCA with planner module in Apollo Open-Source Autonomous Driving Platform and validated its robustness and efficiency with hundreds of hours of road tests in both USA and China. Fig. 6 shows the architecture of trajectory provider with highlighted TDR-OBCA algorithm module. The trajectory provider takes inputs from map, localization, routing, perception and prediction modules to formulate the Region of Interest and transfers obstacles into line segments. The trajectories generated by TDR-OBCA are passed into three post-processing modules before sent to the vehicle's control layer. In trajectory stitching module, each trajectory is stitched with respect to the ego vehicle's position. The safety check module guarantees the ego vehicle have no collisions to moving obstacles. The trajectory partition module divides the trajectory based on its gear location. Finally, the trajectory is sent to control module, which communicates with Controller Area Network (CAN bus) module to drive the ego vehicle based on this provided trajectory.

The three types of simulation scenarios presented in Section IV-A.2 are all selected from hundreds of hours of real road tests. For all their related road tests, the lateral control accuracy is high and in the range from 0.01 to 0.2 meter.

In Apollo planning module, we use a hybrid planer algorithm, which combines the free-space trajectory provider with regular trajectory provider. TDR-OBCA is applied when the starting point or the end point of a trajectory is off driving road. Together with the regular driving planner, TDR-OBCA aims to maneuver ego vehicle to a designated end pose. It is usually applied but not limited to low speed scenarios. The

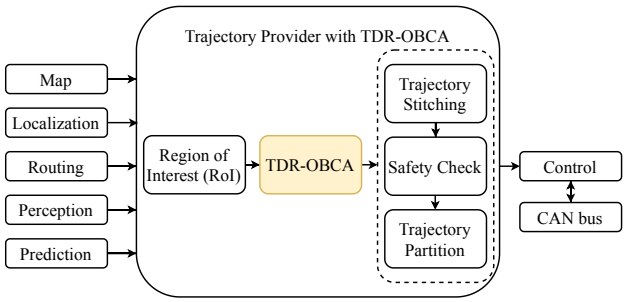



Fig. 6. TDR-OBCA application in Apollo Autonomous Driving Platform.

road-test logic is shown in Algorithm 1.

Here we only take valet parking scenario as an example to show TDR-OBCA's road-test performance in detail. Fig. 7 shows the results of three different valet parking experiments.

According to ego vehicle's status, such as the rear-wheels-to-parking-spot distance and the heading, the car with our algorithm is able to generate either direct parking trajectories or zig-zag trajectories respectively. Table IV shows control performance (including lateral errors and heading angle errors at the end pose of each trajectory) of these three scenarios, where the controller of autonomous vehicle follows the planned trajectory generated by TDR-OBCA. Our results confirm that the trajectories generated by TDR-OBCA are smooth, constrained by vehicle's kinodynamics and lead to low tracking errors in real road tests.

## V. Conclusion

In this paper, we present TDR-OBCA, a robust, efficient and control friendly trajectory generation algorithm for autonomous driving in free space. In TDR-OBCA, two new warm start methods and the optimization problem's reformulation dramatically decrease the simulation failure rate by $97\%$ and computation time by up to $44.82\%$, and increase driving comfort by reducing the steering control output more than $13.53\%$, thus make the real world application possible.

The results have been tested on various scenarios and hundreds of hours' road tests. In the future, we will focus on the further improvement to the computational efficiency.

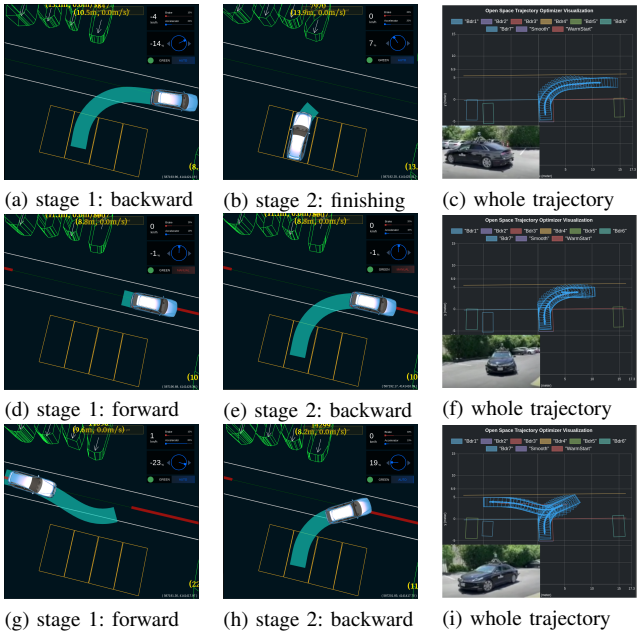
## References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving

**Algorithm 1:** Hybrid Planer with TDR-OBCA

**Result:** Maneuver the vehicle to from starting pose to end pose

Generate RoI from map, routing and localization modules;
Get obstacle shapes and locations from Perception and prediction modules;
**while** *Scenario is not completed* **do**
    Check ego vehicle current status;
    **if** *At end pose* **then**
        Break ;
    **end**
    **if** *Vehicle is on driving road* **then**
        Use on-road planner;
    **else**
        Use TDR-OBCA, adjust position and maneuver forward or backward;
        Stitch trajectory making it start from the current vehicle position;
        Check collisions with moving obstacles;
        Divide trajectory according to gear positions;
    **end**
    Generate control level smooth trajectory with speed profile;
    Generate trajectory tracking control commands with control module;
    Move vehicle with CAN bus module;
    Update current CAN bus status for control module;
**end**



(a) stage 1: backward (b) stage 2: finishing (c) whole trajectory
(d) stage 1: forward (e) stage 2: backward (f) whole trajectory
(g) stage 1: forward (h) stage 2: backward (i) whole trajectory

Fig. 7. Real road test planning trajectories including scenarios: 1) reverse parking: 7a, 7b and 7c; 2) short reverse parking: 7d, 7e and 7f; 3) zig-zag parking: 7g, 7h and 7i.

TABLE IV
VALET PARKING: TRAJECTORY END POSE ACCURACY ON REAL ROAD TESTS

| Parking Scenarios (Partitions) | Lateral Error(m) | Heading Error(deg) |
|---|---|---|
| Reverse Parking (test run 1) | 0.0337 | 0.042 |
| Reverse Parking (test run 2) | 0.0374 | 0.068 |
| Reverse Parking (test run 3) | 0.0446 | 0.098 |
| Short Reverse Parking | 0.0645 | 0.137 |
| Zigzag Parking (forward part) | 0.0331 | 1.271 |
| Zigzag Parking (backward part) | 0.0476 | 0.037 |


urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[2] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.
[3] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.
[4] G. S. Aoude, B. D. Luders, J. P. How, and T. E. Pilutti, "Sampling-based threat assessment algorithms for intersection collisions involving errant drivers," *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 581–586, 2010.
[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[6] A. Eele and A. Richards, "Path-planning with avoidance using non-linear branch-and-bound optimization," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 384–394, 2009.
[7] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowledge-Based Systems*, vol. 86, pp. 11–20, 2015.
[8] M. da Silva Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, "Collision-free encoding for chance-constrained nonconvex path planning," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, 2019.
[9] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," in *Knapsack problems*. Springer, 2004, pp. 235–283.
[10] C. A. Floudas, *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.
[11] A. Richards and J. How, "Mixed-integer programming for control," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 2676–2683.
[12] 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.
[13] J. Xu, Q. Luo, K. Xu, X. Xiao, S. Yu, J. Hu, J. Miao, and J. Wang, "An automated learning-based procedure for large-scale vehicle dynamics modeling on baidu apollo platform," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 5049–5056.
[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.
[15] B. Käpernick and K. Graichen, "The gradient based nonlinear model predictive control software grampc," in *2014 European Control Conference (ECC)*, 2014, pp. 1170–1175.
[16] Z. Yajia, S. Hongyi, Z. Jinyun, H. Jiangtao, and M. Jinghao, "Optimal trajectory generation for autonomous vehicles undercentripetal acceleration constraints for in-lane driving scenarios," in *2019 IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2019.
[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.