2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Autonomous Parking using Optimization-Based Collision Avoidance

Xiaojing Zhang*, Alexander Liniger*, Atsushi Sakai and Francesco Borrelli

Abstract— **We present an optimization-based approach for autonomous parking. Building on recent advances in the area of optimization-based collision avoidance (OBCA), we show that the autonomous parking problem can be formulated as a smooth non-convex optimization problem. Unfortunately, such problems are numerically challenging to solve in general and require appropriate warm-starting. To address this limitation, we propose a novel algorithm called *Hierarchical OBCA* (H-OBCA). The main idea is to first use a generic path planner, such as Hybrid A*, to compute a coarse trajectory using a simplified vehicle model and by discretizing the state-input space. This path is subsequently used to warm-start the OBCA algorithm, which optimizes and smoothens the coarse path using a full vehicle model and continuous optimization. Our studies indicate that the proposed H-OBCA parking algorithm combines Hybrid A*'s global path planning capability with OBCA's ability to generate smooth, collision-free, and dynamically feasible paths. Extensive simulations suggest that the proposed H-OBCA algorithm is robust and admits real-time parking for autonomous vehicles. Sample code is provided at `https://github.com/XiaojingGeorgeZhang/H-OBCA`.**

I. Introduction

Parking a car is a difficult task and can be stressful for a human driver. Today, (semi-)autonomous parking systems are commercially available from several manufacturers. More recently, automated valet parking has attracted the industry's attention, where a car autonomously drives to a parking spot and parks itself [1].

Despite extensive research, the problem of generating obstacle-free trajectories for vehicles in a cluttered parking environment remains a difficult task, especially in tight environments. The main challenges arise from the non-linear and non-holonomic vehicle dynamics and the non-convexity of the free space. Indeed, it has been shown that the task of finding a collision-free path is, in general, NP-hard [2]. Therefore, an ideal all-purpose parking algorithm does not exist, and commercial parking assistance systems are tailored towards "standard" parking scenarios [3].

Recently, optimization-based path planning algorithms, such as Model Predictive Control (MPC), have attracted significant attention, with application ranging from (unmanned) aircraft to robots to autonomous cars [4]–[19]. This can be attributed to the increase in computational resources, the availability of robust numerical algorithms for solving optimization problems, as well as MPC's ability to systematically encode system dynamics and safety constraints inside its formulation. The main challenge in optimization-based approaches is that the obstacle avoidance constraints induce a non-convex optimization problem, often in the form of integer variables, rendering the resulting optimization problem computationally difficult to solve [20]. By introducing auxiliary decision variables, obstacle avoidance constraints can be reformulated as a set of smooth constraints, allowing the use of mature gradient- and Hessian-based numerical solvers [7], [17]. Unfortunately, it has been observed that, due to the non-holonomic dynamics and the non-convexity of the obstacle-free space, the solution quality of these optimization problems in parking problems critically depend on the initial guess provided to the numerical solver, see e.g., [11, Section III.C] and [17, Section 6.3].

In this paper, we propose a novel method for overcoming this issue. In particular, the contribution of this paper can be summarized as follows:

- We propose a hierarchical parking algorithm that combines the classical Hybrid A* path planner [21] with the optimization-based collision avoidance (OBCA) algorithm of [17]. Specifically, on the higher level, we use Hybrid A* and a simplified vehicle model to quickly generate a (coarse) path that approximately satisfies the vehicle dynamics. This path is subsequently passed onto the lower level to initialize the OBCA algorithm, which uses a full vehicle model to generate a high-quality collision-free parking path.
- We demonstrate through extensive numerical simulations that the proposed hierarchical parking algorithm is robust and computationally efficient. Furthermore, we show empirically that the obtained path is smooth, and can be accurately tracked by a simple low level path following controller.

The source code of the proposed parking algorithm is provided at `https://github.com/XiaojingGeorgeZhang/H-OBCA`. In the interest of space, we refer the reader to [22]–[24] for an overview of existing methods for generating collision-free trajectories.

II. Technical Background

A. Obstacle and Car Modeling

Let $x = (X, Y, \varphi, v) \in \mathbb{R}^4$ be the state of the vehicle, where (X, Y) is the center of the rear axis, φ is the heading angle, and v is the longitudinal velocity. For a given state x, we denote by $\mathbb{E}(x) \subset \mathbb{R}^2$ the "space" occupied by the car. Throughout this paper, we model the car to be parked as a

X. Zhang and F. Borrelli are with the MPC Lab, UC Berkeley, USA; A. Liniger is with the Automatic Control Laboratory, ETH Zurich, Switzerland; A. Sakai is with the Komatsu Ltd.; E-mails: liniger@control.ee.ethz.ch, {xiaojing.zhang,francesco.borrelli}@berkeley.edu, atsushi_sakai@global.komatsu.

* These authors contributed equally to this work.

rotated and translated rectangle, i.e.,

$$\mathbb{E}(x) = R(x)\mathbb{B} + t(x), \quad \mathbb{B} := \{y \colon Gy \leq g\}, \tag{1}$$

where $\mathbb{B} \subset \mathbb{R}^2$ is the "initial" rectangle, $R \colon \mathbb{R}^4 \to \mathbb{R}^{2\times 2}$ is a rotation matrix, and $t \colon \mathbb{R}^4 \to \mathbb{R}^2$ is the translation vector. The matrices (G, g) define our initial rectangle $\mathbb{B}$ and are assumed known.

We consider $M \geq 1$ obstacles $\mathbb{O}^{(1)}, \mathbb{O}^{(2)}, \ldots, \mathbb{O}^{(M)} \subset \mathbb{R}^2$ of the form

$$\mathbb{O}^{(m)} = \{y \in \mathbb{R}^2 \colon A^{(m)} y \leq b^{(m)}\}, \tag{2}$$

where $A^{(m)} \in \mathbb{R}^{l_m \times 2}$, $b^{(m)} \in \mathbb{R}^{l_m}$ are known matrices, and l_m is the number of faces of the mth obstacle. Representation (2) is fairly generic since most (also non-convex) obstacles can be approximated as the union of polytopes[1]. Throughout, we assume that the sets $\mathbb{B}, \mathbb{O}^{(1)}, \mathbb{O}^{(2)}, \ldots, \mathbb{O}^{(M)}$ have non-empty relative interior.

The control objective is to move ("park") the ego car from an initial state x_0 to a final state x_F, while avoiding the obstacles (2). Formally, the collision avoidance condition can be expressed as

$$\mathbb{E}(x) \cap \mathbb{O}^{(m)} = \emptyset, \quad \forall m = 1, \ldots, M. \tag{3}$$

It is well-known that the collision avoidance constraint (3) is non-convex and non-differentiable in general, rendering it difficult to use in optimization-based methods such as Model Predictive Control [7], [25]. In the following, we review a recently proposed method for remodeling (3) that preserves both continuity and differentiability, and hence allows the use of mature gradient- and Hessian-based numerical algorithms.

B. Smooth Reformulation of (3)

A popular way of analytically formulating collision avoidance is based on the notion of *signed distance* [25]

$$\text{sd}(\mathbb{E}(x), \mathbb{O}) := \text{dist}(\mathbb{E}(x), \mathbb{O}) - \text{pen}(\mathbb{E}(x), \mathbb{O}), \tag{4}$$

where, to simplify the forthcoming discussion, the superscript "(m)" in $\mathbb{O}^{(m)}$ has been omitted. The terms $\text{dist}(\cdot,\cdot)$ and $\text{pen}(\cdot,\cdot)$ in (4) denote the distance and penetration function, respectively, and are defined as

$$\text{dist}(\mathbb{E}(x), \mathbb{O}) := \min_t \{\|t\| \colon (\mathbb{E}(x) + t) \cap \mathbb{O} \neq \emptyset\}, \tag{5a}$$

$$\text{pen}(\mathbb{E}(x), \mathbb{O}) := \min_t \{\|t\| \colon (\mathbb{E}(x) + t) \cap \mathbb{O} = \emptyset\}, \tag{5b}$$

where $\|\cdot\|$ is the Euclidean distance. It follows that $\text{sd}(\cdot,\cdot)$ between two sets is positive if the two sets do not intersect, and negative if they intersect. Hence, (3) is equivalent to requiring $\text{sd}(\mathbb{E}(x), \mathbb{O}) > 0$. Unfortunately, directly enforcing $\text{sd}(\mathbb{E}(x), \mathbb{O}) > 0$ as a constraint inside an optimization problem is generally difficult since (i) it is non-convex and non-differentiable in general [25], and (ii) an explicit representation of $\text{sd}(\cdot,\cdot)$ is required for optimization algorithms to be numerically efficient.

[1]We refer the interested reader to [17] for obstacles that can be described as general convex sets.

Recently, the authors of [17] have shown that, by introducing additional decision variables, the condition $\text{sd}(\mathbb{E}(x), \mathbb{O}) > 0$ can be reformulated as follows:

Theorem 1 ([17, Theorem 2]): Assume that the obstacles and the car are given as in (2) and (1), respectively. Then, for any $d \in \mathbb{R}$, we have:

$$\begin{aligned} &\text{sd}(\mathbb{E}(x), \mathbb{O}) > d \\ &\iff \exists \lambda \geq 0, \mu \geq 0 \colon -g^\top \mu + (At(x) - b)^\top \lambda > d, \\ &\qquad G^\top \mu + R(x)^\top A^\top \lambda = 0, \ \|A^\top \lambda\| = 1. \end{aligned} \tag{6}$$

Since $\mathbb{E}(x) \cap \mathbb{O} = \emptyset$ is equivalent to $\text{sd}(\mathbb{E}(x), \mathbb{O}) > 0$, collision avoidance is ensured by setting $d = 0$ in (6). We point out that Theorem 1 provides a *smooth* and *exact* characterization of the signed distance. Intuitively speaking, any (μ, λ) satisfying the right-hand-side of (6) provides a certificate for satisfying the condition $\text{sd}(\mathbb{E}(x), \mathbb{O}) > d$. We will see in the next section how Theorem 1 allows us to encode the collision avoidance condition (3) as a set of smooth constraints.

III. Autonomous Parking

In this section, we describe the proposed parking algorithm. We begin by introducing the system model in Section III-A, and formulating the parking problem as a constrained optimal control problem in Section III-B. Finally, Section III-C presents the proposed parking algorithm.

A. System Dynamics, Constraints, and Cost

We model the vehicle dynamics using a standard kinematic bicycle model of the form $\dot{x} = f(x, u)$, which is suited for vehicles at low speeds [26]. The state x is given as in Section II-A, and the input $u := (\delta, a)$ consists of the steering angle δ and the longitudinal acceleration a. The continuous time system is discretized using a second-order Runge-Kutta method such that $x_{k+1} = x_k + \tau\, f(x_k + 0.5\tau f(x_k, u_k), u_k)$, where $\tau > 0$ is the sampling time, and x_k, u_k are the state and input at time step k. Our control objective is to park the car as quickly as possible, while minimizing the control effort. Therefore, we consider a cost of the form $J(\mathbf{u}, \tau) := \kappa\tau + \sum_{k=0}^{N-1} u_k^T Q u_k + \Delta u_k^T Q_\Delta \Delta u_k$, where $\mathbf{u} := [u_0, \ldots, u_{N-1}]$, Q and Q_Δ are positive semi-definite weighting matrices, $\Delta u_k := (u_k - u_{k-1})/\tau$ and $\kappa \geq 0$ is a weight that trades off control effort and minimum-time. Motivated by [27], we do not assume that τ is fixed but rather that it is a decision variable over which we optimize. This is motivated by the fact that the final time is given by $N\tau$; hence, minimizing τ also minimizes the execution time $N\tau$. Furthermore, optimizing over τ has the additional benefit that the maneuver duration does not need to be fixed a priori, which in practice helps avoid infeasibility issues that are caused by short maneuver lengths.

B. Optimization-based Collision Avoidance (OBCA)

Given an initial state x_S and target state x_F, the parking problem consists of navigating the vehicle from x_S to x_F, while avoiding the obstacles $\mathbb{O}^{(1)}, \mathbb{O}^{(2)}, \ldots, \mathbb{O}^{(M)}$ and minimizing the control input. By combining (1), (2) and (6) with

the system dynamcics, the parking problem can therefore be formulated as the following optimal control problem

$$\begin{aligned}\min_{\tau,\mathbf{x},\mathbf{u},\boldsymbol{\lambda},\boldsymbol{\mu}} \quad & \kappa\tau + \sum_{k=0}^{N} u_k^\top Q u_k + \Delta u_k^\top Q_\Delta \Delta u_k \\ \text{s.t.} \quad & x_0 = x_S,\ x_{N+1} = x_F, \\ & x_{k+1} = x_k + \tau\, f(x_k + 0.5\tau f(x_k, u_k), u_k), \\ & h(x_k, u_k) \leq 0, \\ & -g^\top \mu_k^{(m)} + (A^{(m)}\, t(x_k) - b^{(m)})^\top \lambda_k^{(m)} > 0, \\ & G^\top \mu_k^{(m)} + R(x_k)^\top {A^{(m)}}^\top \lambda_k^{(m)} = 0, \\ & \|{A^{(m)}}^\top \lambda_k^{(m)}\| = 1,\ \lambda_k^{(m)} \geq 0,\ \mu_k^{(m)} \geq 0, \\ & \forall k = 0, \dots, N,\ \forall m = 1, \dots, M, \end{aligned} \tag{7}$$

where $\lambda_k^{(m)}$ and $\mu_k^{(m)}$ are the (auxiliary) variables associated with the obstacle $\mathbb{O}^{(m)}$ at step k, and $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the collection of all $\lambda_k^{(m)}$ and $\mu_k^{(m)}$, respectively. The constraint $h(\cdot,\cdot) \leq 0$ encodes input and state constraints as detailed later in Section IV, and collision avoidance is ensured by means of the last five constraints, see Theorem 1. We refer to problem (7) as OBCA, optimization-based collision avoidance. Notice that, by virtue of Theorem 1, OBCA (7) *exactly* encodes the obstacle avoidance constraint (3), i.e., no approximations are involved. If the function $h(\cdot,\cdot)$ is smooth, then OBCA (7) is a smooth optimization problem that can be solved with existing off-the-shelf gradient- or Hessian-based algorithms.

C. Hierarchical Parking Algorithm (H-OBCA)

Recall that (7) is a non-convex optimization problem, and hence computationally intractable to solve in general. In practice, one has to satisfy oneself with a local optima that, for instance, satisfies the KKT conditions [28], [29]. Furthermore, it is well-known that those solutions critically depend on the initial guess (also known as "warm-starting point") that is provided to the solvers, and that different initial guesses can lead to different (local) optima. Unfortunately, computing a good initial guess is often difficult, especially in parking applications where the initial guess should approximately satisfy the non-holonomic vehicle dynamics [17].

In the following, we propose a *hierarchical* parking algorithm where we first use the Hybrid A* algorithm to compute a coarse parking trajectory, which we then use to warm-start OBCA (7)[2]. The individual steps are described in detail next.

(1) Hybrid A*: Hybrid A* is a kinodynamic path planning algorithm that is able to generate paths in clustered environments using a simplified vehicle model [21]. It is a tree search algorithm that consists of two main steps: (i) First, starting from the "best" expanded node, measured in terms of a heuristic function and a user-defined cost function, Hybrid A* uses a simplified vehicle model to create candidate nodes (ii) Second, all collision-free candidate nodes are connected to the goal node using analytical node expansion. If the path is collision-free, Hybrid A* terminates; otherwise it goes to step (i). In the following, we briefly outline the idea of the heuristic function, and also describe steps (i) and (ii) is more detail. The interested reader is referred to [21] for a complete description of Hybrid A*.

Heuristic Function and Cost Function: The "best" expanded node chosen in step (i) depends on a heuristic function and a user-defined cost function. Typically, the cost function penalizes of path length, number of switchback points and steering angle. The heuristic function is used to speed up computation and guides the node expansion direction. In our paper, the heuristic function is the solution of the A* algorithm, which is a method for computing shortest path problems on grids [30], [31]. Together with the cost function, the solution of the A* algorithm guides Hybrid A* by choosing the "best" expanded node in step (i).

Step (i): Hybrid A* uses a simplified kinematic bicycle model that neglects the velocity state v and the acceleration input a. These two quantities are replaced with a discrete input that decides if the car moves forward or backwards. Additionally, the steering angle is discretized. From a given start node, which is determined by the heuristic function and the user-defined cost function, these inputs are used to generate candidate nodes. If multiple candidate nodes enter the same grid cell, only the one with the best (user-defined) cost is kept ("pruning").

Step (ii): For all newly expanded nodes, the Hybrid A* algorithm performs a so-called "analytical node expansion" step, where all possible Reeds-Shepp paths [32] are generated between each node and the goal node. If any of those paths is collision-free, then a feasible path has been found; the node is not further expanded, and the algorithm is stopped[3]. Otherwise, if none of the Reeds-Shepp paths is collision-free, the Hybrid A* will proceed to step (i).

In the following, we denote the path returned by the Hybrid A* algorithm as $\{X_k^{\text{HA}}, Y_k^{\text{HA}}, \varphi_k^{\text{HA}}\}_k$.

(2) Warm-Starting OBCA: We use the solution $\{X_k^{\text{HA}}, Y_k^{\text{HA}}, \varphi_k^{\text{HA}}\}_k$ provided by the Hybrid A* algorithm from step *(1)* to warm-start OBCA (7) as follows: First, we down-sample the path provided by Hybrid A* to obtain the horizon N. Then, using an estimate for the sampling time τ^{ws}, we compute a smooth velocity profile $\{v_k^{\text{ws}}\}_k$ from the Hybrid A* path that satisfies the acceleration limits[4]. Given $\{v_k^{\text{ws}}\}_k$ and $\{\varphi_k^{\text{HA}}\}_k$, we extract the inputs profiles $\{a_k^{\text{ws}}, \delta_k^{\text{ws}}\}_k$. This allows us to use $\{X_k^{\text{HA}}, Y_k^{\text{HA}}, \varphi_k^{\text{HA}}, v_k^{\text{ws}}\}_k$ to warm-start the states $\mathbf{x}$, $\{\delta_k^{\text{ws}}, a_k^{\text{ws}}\}_k$ to warm-start the inputs $\mathbf{u}$, and τ^{ws} to warm-start the sampling time τ. Finally, the auxiliary variables $\lambda_k^{(m)}, \mu_k^{(m)}$ in (7) are initialized as the solution of the following optimization problems $(\lambda_k^{\text{m,ws}}, \mu_k^{\text{m,ws}}) := \max_{\lambda,\mu}\{-g^\top \mu + (A^{(m)} t(X_k^{\text{HA}}, Y_k^{\text{HA}}) - b^{(m)})^\top \lambda\colon G^\top \mu + R(\varphi_k^{\text{HA}})^\top (A^{(m)})^\top \lambda = 0,\ \|(A^{(m)})^\top \lambda\| = 1,\ \lambda \geq 0,\ \mu \geq 0\}$.

[2]Instead of Hybrid A*, other methods such as RRT* could also be used.

[3]A Reeds-Shepp path is a valid path for the kinematic bicycle model [32].

[4]Finding a good warm-start τ^{ws} is challenging and highly problem dependent. In our numerical simulations, values between 0.5–1 s have provided good results.

(3) Solving OBCA: Finally, the OBCA problem (7) is solved with the initial guesses from step *(2)* using an off-the-shelf nonlinear solver. We point out that the warm-start is, in general, infeasible and does not satisfy the system dynamics.

Since the three steps above induce a hierarchical control structure, we call the resulting parking algorithm *Hierarchical Optimization-Based Collision Avoidance (H-OBCA)*. Its main steps are summarized in Algorithm 1.

Algorithm 1 H-OBCA for Autonomous Parking

Input: x_S, x_F, τ^{ws}, vehicle shape $\mathbb{B}$, obstacles $\{\mathbb{O}^{(i)}\}_i$
Output: Optimal trajectory $\mathbf{x}^\star$ and input sequence $\mathbf{u}^\star$

1: Compute $\{X_k^{\text{HA}}, Y_k^{\text{HA}}, \varphi_k^{\text{HA}}\}_k$ using Hybrid A*.
2: Compute initial guesses $\{v_k^{\text{ws}}, a_k^{\text{ws}}, \delta_k^{\text{ws}}, \lambda_k^{\text{m,ws}}, \mu_k^{\text{m,ws}}\}_{k,m}$.
3: Solve OBCA (7) with the initial guesses $\{X_k^{\text{HA}}, Y_k^{\text{HA}}, \varphi_k^{\text{HA}}, v_k^{\text{ws}}, a_k^{\text{ws}}, \delta_k^{\text{ws}}, \lambda_k^{\text{m,ws}}, \mu_k^{\text{m,ws}}, \tau^{\text{ws}}\}_{k,m}$.

IV. Simulation Results

In this section, we provide simulation results of two common parking scenarios, reverse parking and parallel parking, see Figure 1. Sample code is provided at `https://github.com/XiaojingGeorgeZhang/H-OBCA`.

A. Simulation Setup

We model the vehicle as a rectangle of size $4.7 \times 2\,\text{m}$ whose orientation is determined by the car's heading angle. The wheelbase is assumed to be $L = 2.7\,\text{m}$. Steering angle is limited between $\delta \in [-0.6, 0.6]\,\text{rad}$ (approximately ± 34 degrees), the acceleration between $a \in [-0.4, 0.4]\,\text{m/s}^2$. Furthermore, rate constraints on the steering angle are considered $\dot{\delta} \in [-0.6, 0.6]\,\text{rad/s}$, and the car's velocity is limited to lie between $v \in [-1, 2]\,\text{m/s}$. Finally, the sampling time for reverse and parallel parking is initialized with $\tau^{\text{ws}} = 0.6\,\text{s}$ and $\tau^{\text{ws}} = 0.9\,\text{s}$, respectively, and allowed to vary by $\pm 20\%$.

Our Hybrid A* implementation uses a coarse state and input grid to speed up computation time. The state grid size is $0.3\,\text{m}$ in both X and Y dimension, and 5° in the heading angle φ. The steering input is discretized using five points, and we use a motion resolution of $0.1\,\text{m}$. Collision avoidance is ensured by checking the ego car's shape $\mathbb{E}(x)$ with the obstacles' boundaries, that are discretized with a resolution of $0.1\,\text{m}$. As a cost function, we penalize the path length, the reverse movement, number of switchbacks, curve tightness and jerk of the path. As a heuristic, and to speed up computation, we use the collision-free shortest path from each grid point to the final state, computed using the standard A* algorithm (see [21] for details on heuristic functions).

The parking spot in the reverse parking scenario (Fig. 1, top) is assumed $2.6\,\text{m}$ wide and $5.2\,\text{m}$ long. The road where the car can maneuver in is $6\,\text{m}$ wide. The parking spot in the parallel parking scenario (Fig. 1, bottom) is $2.5\,\text{m}$ deep and $6\,\text{m}$ long. The space for maneuvering is $6\,\text{m}$ wide. Clearly, for both scenarios, the obstacles admit polytopic representations of the form (2). All subsequent simulations were performed on a 2013 MacBook Pro with an Intel i7 processor clocked at 2.4GHz. The simulations were implemented in the Julia programming language. The OBCA problem (7) is formulated with the modeling toolbox JuMP and solved with the nonlinear solver IPOPT [28].

B. Results

1) Computation Time: To evaluate the proposed H-OBCA algorithm, we consider 57 parallel parking scenarios and 57 reverse parking scenarios. In each scenario, the end position is fixed at $(X_F, Y_F, \varphi_F, v_F) = (0, 1.3, \pi/2, 0)$ (reverse parking) and $(X_F, Y_F, \varphi_F, v_F) = (-1.35, 4, 0, 0)$ (parallel parking). The starting positions, on the other hand, are obtained by gridding the maneuvering space equally between $X \in [-9, 9]$ and $Y \in [6.5, 9.5]$, with nineteen points in the X-direction and three points in the Y-direction. For all initial conditions, the vehicle is at standstill and oriented to the right, i.e., $v_S = \varphi_S = 0$.

The computation time of H-OBCA is reported in Fig. 1 for all scenarios. They are further broken down in Table I with respect to step 1 of Algorithm 1 (Hybrid A*) and step 3 of Algorithm 1 (OBCA). Step 2 is neglected since it takes roughly $0.1\,\text{s}$.

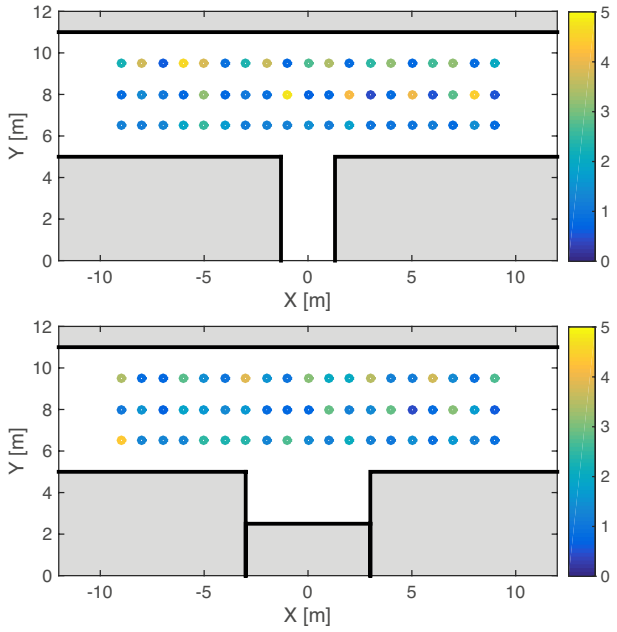

Fig. 1: Computation time of proposed H-OBCA algorithm for reverse (top) and parallel parking (bottom).

2) Driveability of Generated Path: In this section, we examine the quality of the paths generated by H-OBCA in terms of how much time it takes for a simple path follower to follow the computed path, and what the maximum tracking error is. These quantities are evaluated using a simple path follower that consists of a P-controller in the longitudinal direction, and an LQR-controller in the lateral direction. This is a common approach in vehicle path following [26]. Table II reports the maneuvering time and the maximum tracking error of the proposed H-OBCA algorithm.

TABLE I: Computation time of Hybrid A* (step 1), OBCA (step 2) and H-OBCA (Algorithm 1).

	min	max	mean
Reverse Parking			
Hybrid A*	0.1496 s	4.2263 s	0.9276 s
OBCA	0.2814 s	3.2961 s	0.8743 s
Total (H-OBCA)	0.4481 s	4.7011 s	1.8019 s
Parallel Parking			
Hybrid A*	0.1437 s	2.1362 s	0.4603 s
OBCA	0.3137 s	4.0192 s	1.2475 s
Total (H-OBCA)	0.4645 s	4.3262 s	1.7078 s

TABLE II: Maneuvering time and maximum tracking error of path generated by H-OBCA for reverse parking (top) and parallel parking (bottom).

	min	max	mean
Reverse Parking			
maneuver time H-OBCA	14.0500 s	34.8500 s	24.1930 s
max tracking error H-OBCA	0.0380 m	0.0880 m	0.0578 m
Parallel Parking			
maneuver time H-OBCA	17.4500 s	67.3500 s	39.5675 s
max tracking error H-OBCA	0.0500 m	0.1328 m	0.0742 m

C. Discussion and Comparison

1) Computation Time and Success Rate of H-OBCA: We see from Fig. 1 that the proposed H-OBCA algorithm is able to park the car from all 57 starting points. Moreover, we see from Table I that H-OBCA is computationally efficient, with an average computation time of around 1.8 s, and a maximum computation time of 4.7 s. Furthermore, we see from Table II that the path provided by H-OBCA can be easily tracked, with a maximum tracking error of 8.8 cm (reverse parking) and 13.3 cm (parallel parking). This information can be used, for example, to enforce a minimum distance between the obstacle and the car when solving OBCA in (7). Finally, we see from Table II that parallel parking generally requires longer execution time (average 39.6 s) than reverse parking (average 24.2 s). This is due to the fact that the paths in parallel parking are generally longer since the car needs to first drive to the right before it can back into the parking lot. We close this section by pointing out that the numerical results suggest that the proposed H-OBCA algorithm is *robust*, *real-time feasible*, and able to generate *easy-to-track* trajectories.

2) Comparison with OBCA: In this section, we examine how the proposed H-OBCA algorithm compares to a "naive" implementation of OBCA, where problem (7) is solved with a simple (or no) initial guess. In particular, we consider the following two scenarios: (A) No initial guess is provided to OBCA (7); (B) A simple path connecting the start and end position that neglects the obstacles is provided to OBCA (7) as an initial guess. Table III reports the success rate of finding a feasible solution to (7), as well as the computation time for solving (7). Clearly, a naive implementation of OBCA not only results in very low success rates, but also leads to much higher computation time. Indeed, comparing Table III with Table I, we see that H-OBCA has a maximum computation time of 4.7 s, while a naive OBCA has a computation time of 13.8 s. Table III suggests that the proposed hierarchical H-OBCA algorithm vastly outperforms a naive implementation of OBCA, while underscoring the importance of selecting a good warm-start when it comes to solving non-convex optimization problems.

TABLE III: Success rate and computation time of OBCA with no warm-start (A) and a linear interpolation as warm-start (B).

	succ. rate	max comp. time	mean comp. time
Reverse Parking			
(A): no warm-start	43.9%	13.4529 s	8.1932 s
(B): lin. interpolation	84.2%	13.8684 s	6.1525 s
H-OBCA	100%	4.7011 s	1.8019 s
Parallel Parking			
(A): no warm-start	45.6%	13.2161 s	5.3833 s
(B): lin. interpolation	29.8%	13.6204 s	5.8313 s
H-OBCA	100%	4.3262 s	1.7078 s

3) Comparison with Hybrid A*: In this section, we compare the hierarchical H-OBCA algorithm with the Hybrid A* algorithm. We begin by pointing out that the Hybrid A* algorithm has, obviously, a lower computation time than H-OBCA (Table I), since it is an integral part of H-OBCA.

To examine the quality of the paths generated by Hybrid A*, Table IV reports the maneuver time and maximum tracking error along the path when tracked with the aforementioned low-level path following controller. Comparing Table IV with Table II, we see that the paths computed by H-OBCA can be tracked more accurately, although not by much. However, when it comes to maneuver time, we see that paths generated by Hybrid A* require more time to track than the paths generated by H-OBCA. This is mainly due to two reasons: First, the paths generated by H-OBCA satisfy the kinodynamic constraints since the vehicle model is explicitly incorporated in (7). Second, the Hybrid A* algorithm does not explicitly take into account the rate constraints in the steering angle, allowing the car to take "aggressive" curves, see Fig. 2 for an example. To make these curves, the vehicle needs to drive slowly, which explains the longer maneuver times. However, it turns out that by applying a different velocity profile for the low-level controller, one would be able to track the Hybrid A* path quicker. Unfortunately, such a controller would result in a large tracking error which could possibly lead to collisions. Furthermore, we see from Fig. 2 that H-OBCA is able to "correct" paths obtained by the Hybrid A* algorithm, and generate smoother and more natural paths. We summarize the comparison section by pointing out that the hierarchical H-OBCA algorithm is

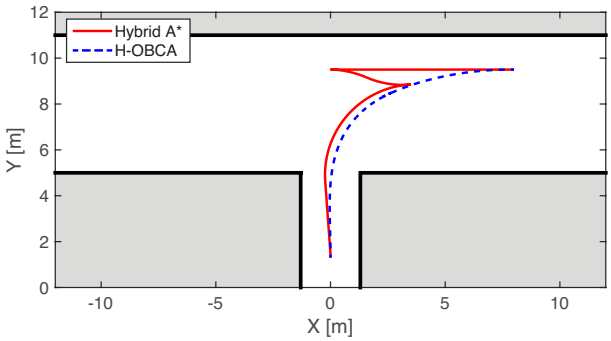

Fig. 2: Paths generated by Hybrid A* (red) and H-OBCA (dashed-blue).

able to generate paths that are smoother and easier to track than those of Hybrid A*.

TABLE IV: Maneuvering time and path following quality of Hybrid A* for reverse parking (top) and parallel parking (bottom).

	min	max	mean
Reverse Parking			
Maneuver time Hybrid A*	36.8500 s	75.9000 s	55.2404 s
Max tracking error Hybrid A*	0.0051	0.1195	0.0692
Parallel Parking			
Maneuver time Hybrid A*	86.4588 s	131.9000 s	51.1000 s
Max tracking error Hybrid A*	0.0367	0.1445	0.0860

V. Conclusion

In this paper, we presented a novel hierarchical controller called H-OBCA for autonomous parking that effectively combines Hybrid A*'s global path planning capability with OBCA's ability to generate smooth, collision-free, and dynamically feasible paths. Our studies suggest that the performance of Hybrid A* and OBCA combined is significantly greater than that of the individual parts. Extensive numerical simulations have demonstrated that the proposed H-OBCA controller is both robust and computationally efficient, with computation times of less than 5 s. Finally, we have shown that the paths generated by H-OBCA are smooth and can be tracked accurately by a low-level path following controller.

References

[1] D. C. Conner, H. Kress-Gazit, H. Choset, A. A. Rizzi, and G. J. Pappas, "Valet parking without a valet," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 572–577.

[2] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[3] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot motion planning and control*. Springer, 1998, pp. 1–53.

[4] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference*, May 2002, pp. 1936–1941.

[5] F. Borrelli, T. Keviczky, and G. J. Balas, "Collision-free UAV formation flight using decentralized optimization and invariant sets," in *IEEE Conference on Decision and Control*, Dec 2004, pp. 1099–1104.

[6] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 65–93.

[7] R. Patel and P. J. Goulart, "Trajectory generation for aircraft avoidance maneuvers using online optimization," *AIAA Journal of Guidance, Control and Dynamics*, vol. 34, no. 1, pp. 218–230, 2011.

[8] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, Dec 2011.

[9] S. Sutrisno, E. Joelianto, A. Budiyono, I. E. Wijayanti, and N. Y. Megawati, "Model predictive control for obstacle avoidance as hybrid systems of small scale helicopter," in *International Conference on Instrumentation Control and Automation*, Aug 2013, pp. 127–132.

[10] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[11] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *Transactions on Intelligent Transportation Systems*, 2016.

[12] J. Carrau, A. Liniger, X. Zhang, and J. Lygeros, "Efficient Implementation of Randomized MPC for Miniature Race Cars," in *European Control Conference*, Jun 2016, pp. 957–962.

[13] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *Transactions on Control Systems Technology*, 2017.

[14] A. Liniger, X. Zhang, P. Aeschbach, A. Georghiou, and J. Lygeros, "Racing Miniatuare Cars: Enhancing Performance using Stochastic MPC and Disturbance Feedback," in *American Control Conference*, Jun 2017.

[15] C. Liu and M. Tomizuka, "Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification," *System & Control Letters*, vol. 108, pp. 56 – 63, 2017.

[16] U. Rosolia, S. DeBruyne, and A. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2017.

[17] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *preprint arXiv:1711.03449*, 2017. [Online]. Available: https://arxiv.org/abs/1711.03449

[18] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "FaSTrack: A modular framework for fast and guaranteed safe motion planning," in *Conference on Decision and Control*, Dec 2017, pp. 1517–1522.

[19] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[20] I. E. Grossmann, "Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques," *Optimization and Engineering*, vol. 3, no. 3, pp. 227–252, 2002.

[21] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments," *International Journal of Robotics Research*, 2010.

[22] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[23] M. Campbell, M. Egerstedt, J. How, and R. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2010.

[24] B. Paden, M. Cáp, S. Yong, D. S. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Transactions on Intelligent Vehicles*, 2016.

[25] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[26] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[27] F. Fahroo and I. Ross, "Direct trajectory optimization by a Chebyshev pseudospectral method," in *American Control Conference*, 2000.

[28] A. Wächter and L. 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.

[29] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear model predictive control*. Springer, 2009, pp. 391–417.

[30] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[31] ——, "Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths"," *ACM SIGART Bulletin*, Dec. 1972.

[32] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.