Optimization-Based Collision Avoidance

Xiaojing Zhang[a], Alexander Liniger[b], Francesco Borrelli[a]

[a] *Model Predictive Control Laboratory, Department of Mechanical Engineering, University of California, Berkeley, USA*
[b] *Automatic Control Laboratory, Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland*

arXiv:1711.03449v3 [math.OC] 10 Jun 2018

Abstract

This paper presents a novel method for reformulating non-differentiable collision avoidance constraints into *smooth* nonlinear constraints using strong duality of convex optimization. We focus on a controlled object whose goal is to avoid obstacles while moving in an n-dimensional space. The proposed reformulation does not introduce approximations, and applies to general obstacles and controlled objects that can be represented as the union of convex sets. We connect our results with the notion of signed distance, which is widely used in traditional trajectory generation algorithms. Our method can be applied to generic navigation and trajectory planning tasks, and the smoothness property allows the use of general-purpose gradient- and Hessian-based optimization algorithms. Finally, in case a collision cannot be avoided, our framework allows us to find "least-intrusive" trajectories, measured in terms of penetration. We demonstrate the efficacy of our framework on a quadcopter navigation and automated parking problem, and our numerical experiments suggest that the proposed methods enable real-time optimization-based trajectory planning problems in tight environments. Source code of our implementation is provided at `https://github.com/XiaojingGeorgeZhang/OBCA`.

Keywords: obstacle avoidance, collision avoidance, path planning, navigation in tight environments, autonomous parking

1. Introduction

Maneuvering autonomous systems in an environment with obstacles is a challenging problem that arises in a number of practical applications including robotic manipulators and trajectory planning for autonomous systems such as self-driving cars and quadcopters. In almost all of those applications, a fundamental feature is the system's ability to avoid collision with obstacles which are, for example, humans operating in the same area, other autonomous systems, or static objects such as walls.

Optimization-based trajectory planning algorithms such as Model Predictive Control (MPC) have received significant attention recently, ranging from (unmanned) aircraft to robots to autonomous cars [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. This can be attributed to the increase in computational resources, the availability

[☆] E-mail addresses: {xiaojing.zhang, fborrelli}@berkeley.edu, liniger@control.ee.ethz.ch

of robust numerical algorithms for solving optimization problems, as well as MPC's ability to systematically encode system dynamics and constraints inside its formulation.

One fundamental challenge in optimization-based trajectory planning is the appropriate formulation of collision avoidance constraints, which are known to be non-convex and computationally difficult to handle in general. While a number of formulations have been proposed in the literature for dealing with collision avoidance constraints, they are typically limited by one of the following features: (*i*) The collision avoidance constraints are approximated through linear constraint, and it is difficult to establish the approximation error [9]; (*ii*) Existing formulations focus on point-mass controlled objects, and are not applicable to full-dimensional objects; (*iii*) When the obstacles are polyhedral, then the collision avoidance constraints are often reformulated using integer variables [13]. While this reformulation is attractive for linear systems with convex constraints since in this case a mixed-integer convex optimization problem can be solved, integer variables should generally be avoided when dealing with nonlinear systems when designing real-time controllers for robotic systems.

In this paper, we focus on a *controlled object* that moves in a general n-dimensional space while avoiding obstacles, and propose a novel approach for modeling obstacle avoidance constraints that overcomes the aforementioned limitations. Specifically, the contributions of this paper can be summarized as follows:

- We show that if the controlled object and the obstacles are described by convex sets such as polytopes or ellipsoids (or can be decomposed into a finite union of such convex sets), then the collision avoidance constraints can be exactly and non-conservatively reformulated as a set of smooth non-convex constraints. This is achieved by appropriately reformulating the *distance*-function between two convex sets using strong duality of convex optimization.

- We provide a second formulation for collision avoidance based on the notion of *signed distance*, which characterizes not only the distance between two objects but also their penetration. This reformulation allows us to compute "least-intrusive" trajectories in case collisions cannot be avoided.

- We demonstrate the efficacy of the proposed obstacle avoidance reformulations on a quadcopter trajectory planning problem and autonomous parking application, where the controlled vehicles must navigate in tight environments. We show that both the distance reformulation and the signed distance reformulation enable real-time path planning and find trajectories even in challenging circumstances.

Furthermore, since both our formulations allow the incorporation of system dynamics and input constraints, the generated trajectories are *kinodynamically feasible*, and hence can be tracked by simple low-level controllers.

This paper is organized as follows: Section 2 introduces the problem setup. Section 3 presents the collision avoidance and minimum-penetration formulations for the case when the controlled object is a point mass.

These results are then extended to full-dimensional controlled objects in Section 4. Numerical experiments demonstrating the efficacy of the proposed method are given in Sections 5 and 6, and conclusions are drawn in Section 7. The Appendix contains auxiliary results needed to prove the main results of the paper. The source code of a quadcopter navigation example and autonomous parking example described in Sections 5 and 6 is provided at `https://github.com/XiaojingGeorgeZhang/OBCA`.

Related Work

A large body of work exists on the topic of obstacle avoidance. In this paper, we do not review, or compare, optimization-based collision avoidance methods with alternative approaches such as those based on dynamic programming [14], reachability analysis [15, 16, 17], graph search [18, 19, 20], (random) sampling [21, 22, 23, 24], or interpolating curves [25]. Indeed, collision avoidance problems are known to be NP-hard in general [26], and all practical methods constitute some sort of "heuristics", whose performance depends on the specific problem and configuration at hand. In the following, we briefly review optimization-based approaches, and refer the interested reader to [27, 28, 29, 30, 31] for a comprehensive review on existing trajectory planning and obstacle avoidance algorithms.

The basic idea in optimization-based methods is to express the collision avoidance problem as an optimal control problem, and then solve it using numerical optimization techniques. One way of dealing with obstacle avoidance is to use unconstrained optimization, in which case the objective function is augmented with "artificial potential fields" that represent the obstacles [32, 33, 34, 35, 36]. More recently, methods based on constrained optimization has attracted attention in the control community, due to its ability to explicitly formulate collision avoidance through constraints [1, 4, 6, 9]. Broadly speaking, constrained optimization-based collision-avoidance algorithms can be divided into two cases based on the modeling of the controlled object: point-mass models and full-dimensional objects. Due to its conceptual simplicity, the vast majority of literature focuses on collision avoidance for point-mass models, and consider the shape of the controlled object by inflating the obstacles. The obstacles are generally assumed to be either polytopes or ellipsoids. For polyhedral obstacles, disjunctive programming can be used to ensure collision avoidance, which is often reformulated as a mixed-integer optimization problem [1, 4, 13]. In case of ellipsoidal obstacles, the collision avoidance constraints can be formulated as a smooth non-convex constraint [37, 38], and the resulting optimization problem can be solved using generic non-linear programming solvers.

The case of full-dimensional controlled objects has, to be best of the authors' knowledge, not been widely studied in the context of optimization-based methods, with the exception of [9, 39]. In [39], the authors model the controlled object through its vertices and, under the assumption that all involved object are rectangles, ensure collision avoidance by keeping all vertices of the controlled object outside the obstacle. A more general way of handling collision avoidance for full-dimensional controlled object has been proposed in [9] using the notion of signed distance, where the authors also propose a sequential linearization technique

to deal with the non-convexity of the signed distance function.

The approach most closely related to our formulation is probably the work of [6], where the authors propose a smooth reformulation of the collision avoidance constraint for point-mass controlled objects and polyhedral obstacles. However, our approach differs from [6] as (i) our approach generalizes to full-dimensional controlled objects, and (ii) we are, based on the notion of penetration, also able to compute least-intrusive trajectories in case collisions cannot be avoided.

Notation

Given a proper cone $\mathcal{K} \subset \mathbb{R}^l$ and two vectors $a, b \in \mathbb{R}^l$, then $a \preceq_{\mathcal{K}} b$ is equivalent to $(b-a) \in \mathcal{K}$. If $\mathcal{K} = \mathbb{R}^l_+$ is the standard cone, then $\preceq_{\mathbb{R}^l_+}$ is equivalent to the standard (element-wise) inequality $\leq$. Moreover, $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$, and $\mathcal{K}^* \subset \mathbb{R}^l$ is the dual cone of $\mathcal{K}$. The "space" occupied by the controlled object (e.g., a drone, vehicle, or robot in general) is denoted as $\mathbb{E} \subset \mathbb{R}^n$; similarly, the space occupied by the obstacles is denoted as $\mathbb{O} \subset \mathbb{R}^n$.

2. Problem Description

2.1. Dynamics, Objective and Constraints

We assume that the dynamics of the controlled object takes the form

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ is the state of the controlled object at time step k given an initial state $x_0 = x_S$, $u_k \in \mathbb{R}^{n_u}$ is the control input, and $f\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ describes the dynamics of the system. In most cases, the state x_k contains information such as the position $p_k \in \mathbb{R}^n$ and angles $\theta_k \in \mathbb{R}^n$ of the controlled object, as well the velocities $\dot{p}_k$ and angular rates $\dot{\theta}_k$. In this paper, we assume that no disturbance is present, and that the system is subject to input and state constraints of the form

$$h(x_k, u_k) \leq 0, \tag{2}$$

where $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_h}$, n_h is the number of constraints, and the inequality in (2) is interpreted element-wise. Our goal is to find a control sequence, over a horizon N, which allows the controlled object to navigate from the initial state x_S to its final state $x_F \in \mathbb{R}^{n_x}$, while optimizing some objective function $J = \sum_{k=0}^{N} \ell(x_k, u_k)$, where $\ell\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is a stage cost, and avoiding $M \geq 1$ obstacles $\mathbb{O}^{(1)}, \mathbb{O}^{(2)}, \ldots, \mathbb{O}^{(M)} \subset \mathbb{R}^n$. Throughout this paper, we assume that the functions $f(\cdot,\cdot)$, $h(\cdot,\cdot)$ and $\ell(\cdot,\cdot)$ are smooth. Smoothness is assumed for simplicity, although all forthcoming statements apply equally to cases when those functions are twice continuously differentiable.

2.2. Obstacle and Controlled Object Modeling

Given the state x_k, we denote by $\mathbb{E}(x_k) \subset \mathbb{R}^n$ the "space" occupied by the controlled object at time k, which we assume is a subset of $\mathbb{R}^n$. The collision avoidance constraint at time k is now given by[1]

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

Constraint (3) is non-differentiable in general, e.g., when the obstacles are polytopic [6, 9]. In this paper, we will remodel (3) in such a way that both continuity and differentiability are preserved. To this end, we assume that the obstacles $\mathbb{O}^{(m)}$ are convex compact sets with non-empty relative interior[2], and can be represented as

$$\mathbb{O}^{(m)} = \{y \in \mathbb{R}^n \colon A^{(m)} y \preceq_{\mathcal{K}} b^{(m)}\}, \tag{4}$$

where $A^{(m)} \in \mathbb{R}^{l \times n}$, $b^{(m)} \in \mathbb{R}^l$, and $\mathcal{K} \subset \mathbb{R}^l$ is a closed convex pointed cone with non-empty interior. Representation (4) is entirely generic since any compact convex set admits a conic representation of the form (4) [40, p.15]. In particular, polyhedral obstacles can be represented as (4) by choosing $\mathcal{K} = \mathbb{R}^l_+$; in this case $\preceq_{\mathcal{K}}$ corresponds to the well-known element-wise inequality $\leq$. Likewise, ellipsoidal obstacles can be represented by letting $\mathcal{K}$ be the second-order cone, see [41] for details. To simplify the upcoming exposition, the same cone $\mathcal{K}$ is assumed for all obstacles; the extension to obstacle-specific cones $\mathcal{K}^{(m)}$ is straight-forward.

In this paper, we will consider controlled objects $\mathbb{E}(x_k)$ that are modeled as *point-masses* as well as *full-dimensional* objects. In the former case, $\mathbb{E}(x_k)$ simply extracts the position p_k from the state x_k, i.e.,

$$\mathbb{E}(x_k) = p_k. \tag{5a}$$

In the latter case, we will model the controlled object $\mathbb{E}(x_k)$ as the rotation and translation of an "initial" convex set $\mathbb{B} \subset \mathbb{R}^n$, i.e.,

$$\mathbb{E}(x_k) = R(x_k)\mathbb{B} + t(x_k), \quad \mathbb{B} := \{y \colon Gy \preceq_{\bar{\mathcal{K}}} g\}, \tag{5b}$$

where $R \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n \times n}$ is an (orthogonal) rotation matrix and $t \colon \mathbb{R}^{n_x} \to \mathbb{R}^n$ is the translation vector. The matrices $(G, g) \in \mathbb{R}^{h \times n} \times \mathbb{R}^h$ and the cone $\bar{\mathcal{K}} \subset \mathbb{R}^h$, which we assume is closed, convex and pointed, define the shape of our initial (compact) set $\mathbb{B}$ and are assumed known. Often, the rotation matrix $R(\cdot)$ depends on the angles θ_k of the controlled object, while the translation vector $t(\cdot)$ depends on the position p_k of the controlled object. We assume throughout that the functions $R(\cdot)$ and $t(\cdot)$ are smooth.

[1] In this paper, we only consider collision avoidance constraints that are associated with the position and geometric shape of the controlled object, which are typically defined by its position p_k and angles θ_k. This is not a restriction of the theory as the forthcoming approaches can be easily generalized to collision avoidance involving other states, but done to simplify exposition of the material.

[2] Non-convex obstacles can often be approximated/decomposed as the union of convex obstacles

2.3. Optimal Control Problem with Collision Avoidance

By combining (1)–(3), the constrained finite-horizon optimal control problem with collision avoidance constraint is given by

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N} \ell(x_k, u_k) \\ \text{s.t.} \quad & x_0 = x_S,\ x_{N+1} = x_F, \\ & \left.\begin{aligned} & x_{k+1} = f(x_k, u_k), \\ & h(x_k, u_k) \leq 0, \\ & \mathbb{E}(x_k) \cap \mathbb{O}^{(m)} = \emptyset, \end{aligned}\right\} \begin{aligned} & k = 0, \dots, N, \\ & m = 1, \dots, M, \end{aligned} \end{aligned} \tag{6}$$

where $\mathbb{E}(x_k)$ is either given by (5a) (point-mass model) or (5b) (full-dimensional set), $\mathbf{x} := [x_0, x_1, \dots, x_{N+1}]$ is the collection of all states, and $\mathbf{u} := [u_0, u_1, \dots, u_N]$ is the collection of all inputs. A key difficulty in solving problem (6), even for linear systems with convex objective function and convex state/input constraints, is the presence of the collision-avoidance constraints $\mathbb{E}(x_k) \cap \mathbb{O}^{(m)} = \emptyset$, which in general are non-convex and non-differentiable [6, 9]. In the following, we present two novel approaches for modeling collision avoidance constraints that preserve continuity and differentiability, and are amendable for use with existing off-the-shelf gradient- and Hessian-based optimization algorithms.

2.4. Collision Avoidance

A popular way of formulating collision avoidance is based on the notion of *signed distance* [9]

$$\text{sd}(\mathbb{E}(x), \mathbb{O}) := \text{dist}(\mathbb{E}(x), \mathbb{O}) - \text{pen}(\mathbb{E}(x), \mathbb{O}), \tag{7}$$

where $\text{dist}(\cdot,\cdot)$ and $\text{pen}(\cdot,\cdot)$ are the distance and penetration function, and are defined as

$$\text{dist}(\mathbb{E}(x), \mathbb{O}) := \min_t \{\|t\| \colon\ (\mathbb{E}(x) + t) \cap \mathbb{O} \neq \emptyset\}, \tag{8a}$$

$$\text{pen}(\mathbb{E}(x), \mathbb{O}) := \min_t \{\|t\| \colon\ (\mathbb{E}(x) + t) \cap \mathbb{O} = \emptyset\}. \tag{8b}$$

Roughly speaking, the signed distance is positive if $\mathbb{E}(x)$ and $\mathbb{O}$ do not intersect, and negative if they overlap. Therefore, collision avoidance can be ensured by requiring $\text{sd}(\mathbb{E}(x), \mathbb{O}) > 0$. Unfortunately, directly enforcing $\text{sd}(\mathbb{E}(x), \mathbb{O}) > 0$ inside the optimization problem (6) is generally difficult since it is non-convex and non-differentiable in general [9]. Furthermore, for optimization algorithms to be numerically efficient, they require an explicit representation of the functions they are dealing with, in this case $\text{sd}(\cdot,\cdot)$. This, however, is difficult to obtain in practice since $\text{sd}(\cdot,\cdot)$ itself is the solution of the optimization problems (8a) and (8b). As a result, existing algorithms approximate (7) through local linearization [9], for which it is difficult to establish bounds on approximation errors.

In the following, we propose two reformulation techniques for obstacles avoidance that overcome the issues of non-differentiability and that do not require an explicit representation of the signed distance. We

begin with point-mass models in Section 3, and treat the general case of full-dimensional controlled objects in Section 4.

3. Collision Avoidance for Point-Mass Models

In this section, we first present a smooth reformulation of (6) when $\mathbb{E}(x_k) = p_k$ in Section 3.1, and then extend the approach in Section 3.2 to generate minimum-penetration trajectories in case collisions cannot be avoided. To simplify notation, the time indices k are omitted in the remainder of this section.

3.1. Collision-Free Trajectory Generation

Proposition 1. *Assume that the obstacle $\mathbb{O}$ and the controlled object are given as in* (4) *and* (5a)*, respectively, and let $\mathsf{d}_{\min} \geq 0$ be a desired safety margin between the controlled object and the obstacle. Then we have:*

$$\begin{aligned} &\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}_{\min} \\ &\quad \iff \exists \lambda \succeq_{\mathcal{K}^*} 0 \colon (A\,p - b)^\top \lambda > \mathsf{d}_{\min},\ \|A^\top \lambda\|_* \leq 1. \end{aligned} \tag{9}$$

Proof. It follows from (4) and (8a) that $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) = \min_t \{\|t\| \colon A(\mathbb{E}(x) + t) \preceq_{\mathcal{K}} b\}$. Following [41, p.401], its dual problem is given by $\max_\lambda \left\{(A\mathbb{E}(x) - b)^\top \lambda \colon \|A^\top \lambda\|_* \leq 1,\ \lambda \succeq_{\mathcal{K}^*} 0\right\}$, where $\|\cdot\|_*$ is the dual norm associated to $\|\cdot\|$ and $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$. Since $\mathbb{O}$ is assumed to have non-empty relative interior, strong duality holds, and $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) = \max_\lambda \left\{(A\,\mathbb{E}(x) - b)^\top \lambda \colon \|A^\top \lambda\|_* \leq 1,\ \lambda \succeq_{\mathcal{K}^*} 0\right\}$. Hence, for any non-negative scalar $\mathsf{d}_{\min}$, $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}_{\min}$ is satisfied if, and only if, there exists $\lambda \succeq_{\mathcal{K}^*} 0 \colon (A\,\mathbb{E}(x) - b)^\top \lambda > \mathsf{d}_{\min},\ \|A^\top \lambda\|_* \leq 1$. The desired result follows from identity (5a). □

Intuitively speaking, any variable λ satisfying the right-hand-side of (9) is a certificate verifying the condition $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}_{\min}$. Since $\mathbb{E}(x) \cap \mathbb{O} = \emptyset$ is equivalent to $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) > 0$, the optimal control problem (6) for the point-mass model (5a) is given by

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u},\boldsymbol{\lambda}} \quad & \sum_{k=0}^{N} \ell(x_k, u_k) \\ \text{s.t.} \quad & x_0 = x_S,\ x_{N+1} = x_F, \\ & x_{k+1} = f(x_k, u_k),\ h(x_k, u_k) \leq 0, \\ & (A^{(m)}\,p_k - b^{(m)})^\top \lambda_k^{(m)} > 0, \\ & \|A^{(m)^\top} \lambda_k^{(m)}\|_* \leq 1,\ \lambda_k^{(m)} \succeq_{\mathcal{K}^*} 0, \\ & \text{for } k = 0, \ldots, N,\ m = 1, \ldots, M, \end{aligned} \tag{10}$$

where p_k is the position of the controlled object at time k, $\lambda_k^{(m)}$ is the dual variable associated with obstacle $\mathbb{O}^{(m)}$ at time step k, and the optimization is performed over the states $\mathbf{x}$, the inputs $\mathbf{u}$ and the dual variables

$\boldsymbol{\lambda} = [\lambda_0^{(1)}, \ldots, \lambda_0^{(m)}, \lambda_1^{(1)}, \ldots, \lambda_N^{(m)}]$. We emphasize that (10) is an *exact* reformulation of (6) and that the optimal trajectory $\mathbf{x}^* = [x_0^*, x_1^*, \ldots, x_N^*]$ obtained by solving (6) is *kinodynamically feasible*.

Remark 1. Without further assumptions on the norm $\|\cdot\|$ and the cone $\mathcal{K}$, the last two constraints in (10) are not guaranteed to be smooth, a property that many general-purpose non-linear optimization algorithms require[3]. Fortunately, it turns out that these constraints are smooth for the practically relevant cases of $\|\cdot\|$ being the Euclidean distance and $\mathcal{K}$ either the standard cone or the second-order cone, which allows us to model polyhedral and ellipsoidal obstacles. In these cases, and under the assumption that the functions $f(\cdot,\cdot)$, $h(\cdot,\cdot)$ and $\ell(\cdot,\cdot)$ are smooth, (10) is a smooth nonlinear optimization problem that is amendable to general-purpose non-linear optimization algorithms such as IPOPT [42]. Without going into details, we point out that smoothness is retained when $\|\cdot\| = \|\cdot\|_p$ is a general p-norm, with $p \in (1, \infty)$, and $\mathcal{K}$ is the cartesian product of p-order cones $\mathcal{K}_p := \{(s, z) \colon \|z\|_p \leq s\}$, with $p \in (1, \infty)$. In this case, the dual norm is given by $\|\cdot\|_* = \|\cdot\|_q$ and the dual cone is $(\mathcal{K}_p)^* = \mathcal{K}_q$, where q satisfies $1/p + 1/q = 1$, see [41] for details on dual norms and dual cones.

While reformulation (10) can be used for obstacle avoidance, it is limited to finding collision-free trajectories. Indeed, in case collisions cannot be avoided, the above formulation is not able to find "least-intrusive" trajectories by softening the constraints. Intuitively speaking, this is because (10) is based on the notion of distance, and the distance between two overlapping objects (as is in the case of collision), is always zero, regardless of the penetration. From a practical point of view, this implies that slack variables cannot be included in the constraints of (9), because the optimal control problem is not able to distinguish between "severe" and "less severe" colliding trajectories. Furthermore, in practice, it is often desirable to soften constraints and include slack variables to ensure feasibility of the (non-convex) optimization problem, since (local) infeasibilities in non-convex optimization problem are known to cause numerical difficulties. In the following, we show how the above limitations can be overcome by considering the notion of penetration and softening the collision avoidance constraints.

3.2. Minimum-Penetration Trajectory Generation

In this section, we consider the design of *minimum-penetration* trajectories for cases when collision cannot be avoided and the goal is to find a "least-intrusive" trajectory. Following the literature [43, 44], we measure "intrusion" in terms of *penetration* as defined in (8b).

Proposition 2. *Assume that the obstacle* $\mathbb{O}$ *and controlled object are given as in* (4) *and* (5a)*, respectively,*

[3]Strictly speaking, these solvers often require the cost function and constraints to be twice continuously differentiable only. Smoothness is assumed in this paper for the sake of simplicity.

and let $\mathsf{p}_{\max} \geq 0$ *be a desired maximum penetration of the controlled object and the obstacle. Then we have:*

$$\begin{aligned} &\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) < \mathsf{p}_{\max} \\ &\quad \iff \exists \lambda \succeq_{\mathcal{K}^*} 0 \colon (b - A\,p)^\top \lambda < \mathsf{p}_{\max},\ \|A^\top \lambda\|_* = 1. \end{aligned} \tag{11}$$

Proof. The proof, along with auxiliary lemmas, is given in the Appendix. □

Proposition 2 resembles Proposition 1 with the difference that the convex inequality constraint $\|A^\top \lambda\|_* \leq 1$ is replaced with the non-convex equality constraint $\|A^\top \lambda\|_* = 1$. In the following, we will see that Propositions 1 and 2 can be combined to represent the *signed distance* as defined in (7).

Theorem 1. *Assume that the obstacle* $\mathbb{O}$ *and the controlled object are given as in* (4) *and* (5a)*, respectively. Then, for any* $\mathsf{d} \in \mathbb{R}$*, we have:*

$$\begin{aligned} &\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d} \\ &\quad \iff \exists \lambda \succeq_{\mathcal{K}^*} 0 \colon (A\,p - b)^\top \lambda > \mathsf{d},\ \|A^\top \lambda\|_* = 1. \end{aligned} \tag{12}$$

Proof. By definition, $\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) = \operatorname{dist}(\mathbb{E}(x), \mathbb{O})$ if $\mathbb{E}(x) \cap \mathbb{O} = \emptyset$, and $\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) = -\operatorname{pen}(\mathbb{E}(x), \mathbb{O})$ if $\mathbb{E}(x) \cap \mathbb{O} \neq \emptyset$. Let $\mathbb{E}(x) \cap \mathbb{O} \neq \emptyset$, in which case (12) follows directly from (11). If $\mathbb{E}(x) \cap \mathbb{O} = \emptyset$, then we have from (9) that $\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}$ is equivalent to $\exists \lambda \succeq_{\mathcal{K}^*} 0 \colon (A\,p - b)^\top \lambda > \mathsf{d},\ \|A^\top \lambda\|_* \leq 1$. Due to homogeneity with respect to λ, if the previous condition is satisfied, then there always exists a (scaled) dual multiplier $\lambda' \succeq_{\mathcal{K}^*} 0$ such that $(A\,p - b)^\top \lambda' > \mathsf{d},\ \|A^\top \lambda'\|_* = 1$. This concludes the proof. □

Reformulation (12) is similar to reformulation (9), with the difference that (12) holds for all $\mathsf{d} \in \mathbb{R}$, while (9) only holds for $\mathsf{d} \geq 0$. The "price" we pay for this generalization is that the convex constraint $\|A^\top \lambda\|_* \leq 1$ is turned into the non-convex equality constraint $\|A^\top \lambda\|_* = 1$ which, as we will see later on, generally results in longer computation times. Nevertheless, Theorem 1 allows us to compute trajectories of least penetration whenever collision cannot be avoided by solving the following soft-constrained *minimum-penetration* problem:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u},\mathbf{s},\boldsymbol{\lambda}} \quad & \sum_{k=0}^{N} \left[\ell(x_k, u_k) + \kappa \cdot \sum_{m=1}^{M} s_k^{(m)} \right] \\ \text{s.t.} \quad & x_0 = x(0),\ x_{N+1} = x_F, \\ & x_{k+1} = f(x_k, u_k),\ h(x_k, u_k) \leq 0, \\ & (A^{(m)}\,p_k - b^{(m)})^\top \lambda_k^{(m)} > -s_k^{(m)}, \\ & \|{A^{(m)}}^\top \lambda_k^{(m)}\|_* = 1, \\ & s_k^{(m)} \geq 0,\ \lambda_k^{(m)} \succeq_{\mathcal{K}^*} 0, \\ & \text{for } k = 0, \ldots, N,\ m = 1, \ldots, M, \end{aligned} \tag{13}$$

where p_k is the position of the controlled object at time k, $s_k^{(m)} \in \mathbb{R}_+$ is the slack variable associated to the object $\mathbb{O}^{(m)}$ at time step k, and $\kappa \geq 0$ is a weight factor that keeps the slack variable as close to zero as

possible. Without going into details, we point out that the weight κ should be chosen "big enough" such that the slack variables only become active when the original problem is infeasible, i.e., when obstacle avoidance is not possible [45]. Notice that a positive slack variable implies a colliding trajectory, where the penetration depth is given by $s_k^{(m)}$. We close this section by pointing out that if, a priori, it is known that a collision-free trajectory can be generated, then formulation (10) should be given preference over formulation (13) because the former has fewer decision variables, and because the constraint $\|{A^{(m)}}^\top \lambda_k^{(m)}\|_* \leq 1$ is convex, which generally leads to improved solution times. Smoothness of (13) is ensured if $\|\cdot\|$ is the Euclidean distance, and $\mathcal{K}$ is either the standard cone or the second-order cone, see Remark 1 for details.

4. Collision Avoidance for Full-Dimension Controlled Objects

The previous section provided a framework for computing collision-free and minimum-penetration trajectories for controlled objects that are described by point-mass models. While such models can be used to generate trajectories for "ball-shaped" controlled objects, done by setting the minimum distance $d_{\min}$ equal to the radius of the controlled object (see Section 5 for such an example), it can be restrictive in other cases. For example, modeling a car in a parking lot as a Euclidean ball can be very conservative, and prevent the car from finding a parking spot. To alleviate this issue, we show in this section how the results of Section 3 can be extended to full-dimensional controlled objects.

4.1. Collision-Free Trajectory Generation

Similar to Section 3, we begin by first reformulating the distance function, which will allow us to generate collision-free trajectories:

Proposition 3. *Assume that the controlled object and the obstacle are given as in* (5b) *and* (4)*, respectively, and let* $\mathsf{d}_{\min} \geq 0$ *be a desired safety margin. Then we have:*

$$\begin{aligned} &\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}_{\min} \qquad (14)\\ &\quad \Leftrightarrow\ \exists \lambda \succeq_{\mathcal{K}^*} 0, \mu \succeq_{\bar{\mathcal{K}}^*} 0 \colon\ -g^\top \mu + (At(x) - b)^\top \lambda > \mathsf{d}_{\min},\ G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* \leq 1. \end{aligned}$$

Proof. Recall that $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) = \min_{e,o}\{\|e-o\| \colon Ao \preceq_{\mathcal{K}} b, e \in \mathbb{E}(x)\} = \min_{e',o}\{\|R(x)e' + t(x) - o\| \colon Ao \preceq_{\mathcal{K}} b, Ge' \preceq_{\bar{\mathcal{K}}} g\}$, where the last equality follows from (5b). The dual of this minimization problem is given by $\max_{\lambda,\mu}\{-g^\top \mu + (At(x) - b)^\top \lambda \colon G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* \leq 1,\ \lambda \succeq_{\mathcal{K}^*} 0,\ \mu \succeq_{\bar{\mathcal{K}}^\star} 0\}$, see e.g., [41, Section 8.2] for the derivation, where $\|\cdot\|_*$ is the dual norm, and $\mathcal{K}^*$ and $\bar{\mathcal{K}}^\star$ are the dual cones of $\mathcal{K}$ and $\bar{\mathcal{K}}$, respectively. Since $\mathbb{O}$ and $\mathbb{B}$ are assumed to have non-empty relative interior, strong duality holds, and $\operatorname{dist}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}_{\min} \ \Leftrightarrow\ \max_{\lambda,\mu}\{-g^\top \mu + (At(x) - b)^\top \lambda \colon G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* \leq 1,\ \lambda \succeq_{\mathcal{K}^*} 0,\ \mu \succeq_{\bar{\mathcal{K}}^\star} 0\} > \mathsf{d}_{\min} \ \Leftrightarrow \exists \lambda \succeq_{\mathcal{K}^*} 0, \mu \succeq_{\bar{\mathcal{K}}^*} 0 \colon\ -g^\top \mu + (At(x) - b)^\top \lambda > \mathsf{d}_{\min},\ G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* \leq 1.$ □

Compared to Proposition 1, we see that full-dimensional controlled objects require the introduction of the additional dual variables $\mu^{(m)}$, one for each obstacle $\mathbb{O}^{(m)}$. By setting $\mathsf{d}_{\min} = 0$, we obtain now the following reformulation of (6) for full-dimensional objects:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u},\boldsymbol{\lambda},\boldsymbol{\mu}} \quad & \sum_{k=0}^{N} \ell(x_k, u_k) \\
\text{s.t.} \quad & x_0 = x(0),\ x_{N+1} = x_F, \\
& x_{k+1} = f(x_k, u_k),\ h(x_k, u_k) \leq 0, \\
& -g^\top \mu_k^{(m)} + (A^{(m)}\, t(x_k) - b^{(m)})^\top \lambda_k^{(m)} > 0, \\
& G^\top \mu_k^{(m)} + R(x_k)^\top {A^{(m)}}^\top \lambda_k^{(m)} = 0, \\
& \|{A^{(m)}}^\top \lambda_k^{(m)}\|_* \leq 1,\ \lambda_k^{(m)} \succeq_{\mathcal{K}^*} 0,\ \mu_k^{(m)} \succeq_{\bar{\mathcal{K}}^*} 0, \\
& \text{for } k = 0, \ldots, N,\ m = 1, \ldots, M,
\end{aligned} \tag{15}$$

where $\lambda_k^{(m)}$ and $\mu_k^{(m)}$ are the dual variables associated with the obstacle $\mathbb{O}^{(m)}$ at step k, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the collection of all $\lambda_k^{(m)}$ and $\mu_k^{(m)}$, respectively, and the optimization is performed over $(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu})$. Notice that (15) is an *exact* reformulation of (6). Smoothness of (15) is ensured if $\|\cdot\|$ is the Euclidean distance, and $\mathcal{K}$ and $\bar{\mathcal{K}}$ are either the standard cone or the second-order cone, see also Remark 1 for details.

Similar to the point-mass case in Section 3.1, the optimal control problem (15) is able to generate collision-free trajectories, but unable to find "least-intrusive" trajectories in case collision-free trajectories do not exist. This limitation is addressed next.

4.2. Minimum-Penetration Trajectory Generation

We overcome the above limitation by considering again the notion of penetration. We begin with the following result:

Proposition 4. *Assume that the obstacles and controlled object are given as in* (4) *and* (5b)*, respectively, and let* $\mathsf{p}_{\max} \geq 0$ *be a maximal penetration depth. Then we have:*

$$\begin{aligned}
&\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) < \mathsf{p}_{\max} \\
&\quad \iff \exists \lambda \succeq_{\mathcal{K}^*} 0, \mu \succeq_{\bar{\mathcal{K}}^*} 0 \colon g^\top \mu + (b - A\, t(x))^\top \lambda < \mathsf{p}_{\max},\ G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* = 1.
\end{aligned} \tag{16}$$

Proof. It follows from [43] that $\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) = \operatorname{pen}(0, \mathbb{O} - \mathbb{E}(x))$, where $\mathbb{O} - \mathbb{E}(x) := \{o - e \colon o \in \mathbb{O},\ e \in \mathbb{E}(x)\}$ is the Minkowski difference. Furthermore, we have from the proof of Proposition 2 that $\operatorname{pen}(0, \mathbb{O} - \mathbb{E}(x)) = \inf_{\{z \colon \|z\|_* = 1\}} \{\max_{y \in \mathbb{O} - \mathbb{E}(x)} \{y^\top z\}\}$. Using strong duality of convex optimization, we can dualize the inner maximization problem as $\max_{o \in \mathbb{O}, e \in \mathbb{E}(x)} \{z^\top (o - e)\} = \max_{o \in \mathbb{O}, e' \in \mathbb{B}} \{z^\top (o - R(x)e' - t(x))\} = \min_{\lambda,\mu} \{b^\top \lambda + g^\top \mu - z^\top t(x) \colon A^\top \lambda = z,\ G^\top \mu = -R^\top z \colon \lambda \succeq_{K^*} 0,\ \mu \succeq_{\bar{\mathcal{K}}^*} 0\}$. Hence, $\operatorname{pen}(0, \mathbb{O} - \mathbb{E}(x)) = \inf_{z,\lambda,\mu} \{b^\top \lambda + g^\top \mu - z^\top t(x) \colon A^\top \lambda = z,\ G^\top \mu = -R^\top z,\ \|z\|_* = 1\}$. Eliminating the z-variable using the first equality constraint and following the steps of the proof of Proposition 2 gives the desired result. □

The following theorem shows that Propositions 3 and 4 can be combined to represent the *signed distance* function.

Theorem 2. *Assume that the obstacles and controlled object are given as in* (4) *and* (5b)*, respectively. Then, for any* $\mathsf{d} \in \mathbb{R}$*, we have:*

$$\begin{aligned} &\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d} \\ &\quad \iff \exists \lambda \succeq_{\mathcal{K}^*} 0, \mu \succeq_{\bar{\mathcal{K}}^*} 0 \colon -g^\top \mu + (At(x) - b)^\top \lambda > \mathsf{d},\ G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* = 1. \end{aligned} \tag{17}$$

Proof. By definition, $\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) = \operatorname{dist}(\mathbb{E}(x), \mathbb{O})$ if $\mathbb{E}(x) \cap \mathbb{O} = \emptyset$, and $\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) = -\operatorname{pen}(\mathbb{E}(x), \mathbb{O})$ if $\mathbb{E}(x) \cap \mathbb{O} \neq \emptyset$. Consider now $\mathbb{E}(x) \cap \mathbb{O} \neq \emptyset$, in which case (17) follows directly from (16). Assume now that $\mathbb{E}(x) \cap \mathbb{O} = \emptyset$; then we have from (14) that $\operatorname{sd}(\mathbb{E}(x), \mathbb{O}) > \mathsf{d}$ is equivalent to $\exists \lambda \succeq_{\mathcal{K}^*} 0, \mu \succeq_{\bar{\mathcal{K}}^*} 0 \colon -g^\top \mu + (At(x) - b)^\top \lambda > \mathsf{d},\ G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* \leq 1$. Due to homogeneity with respect to λ and μ, if the previous condition is satisfied, then there also exists a $\lambda' \succeq_{\mathcal{K}^*} 0$ and $\mu' \succeq_{\bar{K}^*} 0$ such that $-g^\top \mu + (At(x) - b)^\top \lambda > \mathsf{d},\ G^\top \mu + R(x)^\top A^\top \lambda = 0,\ \|A^\top \lambda\|_* = 1$. This concludes the proof. □

Theorem 2 allows us to formulate the following soft-constrained *minimum-penetration* optimal control problem

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u},\mathbf{s},\boldsymbol{\lambda},\boldsymbol{\mu}} \quad & \sum_{k=0}^{N} \left[\ell(x_k, u_k) + \kappa \cdot \sum_{m=1}^{M} s_k^{(m)} \right] \\ \text{s.t.} \quad & x_0 = x_S,\ x_{N+1} = x_F, \\ & x_{k+1} = f(x_k, u_k),\ h(x_k, u_k) \leq 0, \\ & -g^\top \mu_k^{(m)} + (A^{(m)} t(x_k) - b^{(m)})^\top \lambda_k^{(m)} > -s_k^{(m)}, \\ & G^\top \mu_k^{(m)} + R(x_k)^\top {A^{(m)}}^\top \lambda_k^{(m)} = 0, \\ & \|{A^{(m)}}^\top \lambda_k^{(m)}\|_* = 1, \\ & s_k^{(m)} \geq 0,\ \lambda_k^{(m)} \succeq_{\mathcal{K}^*} 0,\ \mu_k^{(m)} \succeq_{\bar{\mathcal{K}}^*} 0, \\ & \text{for } k = 0, \ldots, N,\ m = 1, \ldots, M. \end{aligned} \tag{18}$$

where $s_k^{(m)} \in \mathbb{R}_+$ is the slack variable associated to obstacle $\mathbb{O}^{(m)}$ at time step k, and $\kappa \geq 0$ is a weight factor that keeps the slack variable as small as possible. Smoothness of (18) is ensured if $\|\cdot\|$ is the Euclidean distance, and $\mathcal{K}$ and $\bar{\mathcal{K}}$ are either the standard cone or the second-order cone, see Remark 1 for details.

In the following sections, we illustrate our obstacle avoidance formulation on two applications: a quadcopter path planning problem where the point-mass formulation is used (Section 5), and an automated parking problem, where the full-dimensional obstacle avoidance problem formulation is used (Section 6).

5. Example 1: Quadcopter Path Planning

In this section, we illustrate reformulations (10) and (13) on a quadcopter navigation problem, where the quadcopter must find a path from one end of the room to the other end, while avoiding a low-hanging

wall and passing through a small window hole, see Fig. 1.

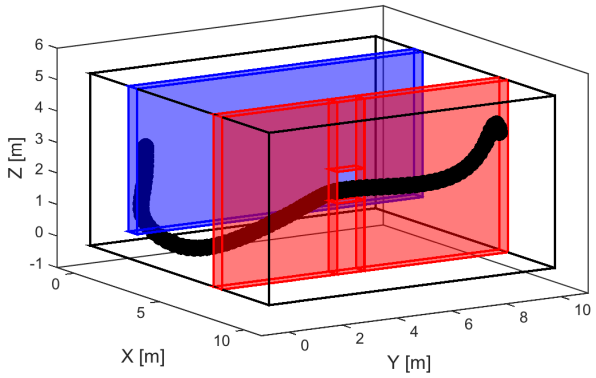

Figure 1: Setup of quadcopter example together with a (locally optimal) point-to-point trajectory. The start position is behind the blue wall at $X = 1$, and the end position in front of red wall at $X = 9$. The quadcopter needs to fly below the blue wall and pass through a window in the red wall, see `https://youtu.be/7WLNJhaHcoQ` for an animation. The black circles illustrate the safety distance $\mathtt{d}_{\min} = 0.25\,\mathrm{m}$ which models the shape of the quadcopter.

5.1. Environment and Obstacle Modeling

The size of the room is $10.5 \times 10.5 \times 5.5\,\mathrm{m}$, and we see from Fig. 1 that the direct path between the start and end position is blocked by two obstacles. The first obstacle, a low-hanging wall, blocks the entire upper part of the room, and can only be passed from below. The second obstacle, another wall, blocks the entire room, but has a small window through which the quadcopter must pass to reach its target position. We approximate the shape of the quadcopter by a (Euclidean) sphere of radius 0.25 m. In the framework of (10) and (13), the shape of the quadcopter can be taken into account by requiring a safety distance of $\mathtt{d}_{\min} = 0.25\,\mathrm{m}$.

The first wall, which can only be passed from below, is placed at $X = 2\,\mathrm{m}$, and the passage below is 0.85 m high. The second wall is placed at $X = 7\,\mathrm{m}$, and the window (size $1 \times 1\mathrm{m}$) is placed in the middle of the second wall at a height of $Z = 2.5\,\mathrm{m}$. Finally the depth of both walls is 0.5 m. This obstacle formation can be formally formulated using five axis-aligned rectangles, where the first obstacle is represented by one such rectangle and the window can be modeled as the union of four rectangles, see Fig. 1. The collision avoidance constraints with respect to the four outer walls of the room are achieved by appropriately upper- and lower-bounding the $(X, Y, Z)-$coordinates of the quadcopter.

5.2. Quadcopter Model

We consider the standard quadcopter model as used in [46], which is derived by finding the equation of motion of the center of gravity (CoG) and summarized next. In this model, X, Y, Z denote the position of the CoG in the world frame, and we use the Z-X-Y Euler angles do describe the rotation of the quadcopter, where ϕ is the pitch angle, θ is the roll angle, and ψ is the yaw angle. The rotation matrix that translates from the world to the body frame, which is defined with respect to the CoG, is hence given by

$$R_{WB} = \begin{bmatrix} c_\psi c_\theta - s_\phi s_\psi s_\theta & -c_\phi s_\psi & c_\psi s_\theta + c_\phi s_\psi s_\theta \\ s_\psi c_\theta + s_\phi c_\psi s_\theta & c_\phi c_\psi & s_\psi s_\theta - s_\phi c_\psi c_\theta \\ -c_\phi s_\theta & s_\phi & c_\phi c_\theta \end{bmatrix},$$

where $s_\phi := \sin(\phi)$ and $c_\phi := \cos(\phi)$. The accelerations of the CoG can be derived by considering the sum of the forces produced by the four rotors F_i which point in positive z-direction in the body frame, and the gravity force which acts on the negative z-direction in the world frame, resulting in the following equation of motion,

$$m \begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R_{WB} \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{bmatrix}, \tag{19a}$$

where m is the mass of the quadcopter and $\ddot{X}$, $\ddot{Y}$ and $\ddot{Z}$ are the second time derivatives of X, Y and Z, respectively. The attitude dynamics of the quadcopter is derived in the body rates p, q, and r which are related to the Euler angles through the following rotation matrix,

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & s_\theta \\ s_\theta t_\phi & 1 & -c_\theta t_\phi \\ -s_\theta / c_\phi & 0 & c_\theta / c_\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{19b}$$

The body rates are given by the following equation of motion, which is driven by the four rotor forces F_i, as well as the corresponding moments M_i and has the following form,

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{19c}$$

where I is the inertia matrix which in our case is diagonal and L is the distance from the CoG to the rotor. The rotor forces and moments depend quadratically on the motor speed ω_i, which are the control inputs and are defined as follows,

$$F_i = k_F \omega_i^2, \qquad M_i = k_M \omega_i^2, \tag{19d}$$

where, k_F and k_M are constants depending on the rotor blades. Hence, the state of the quadcopter is $x = [X, Y, Z, \phi, \theta, \psi, \dot{X}, \dot{Y}, \dot{Z}, p, q, r]$ and the inputs are the four rotor speeds $u = [\omega_1, \omega_2, \omega_3, \omega_4]$.

The parameters of the model, as well as the bounds on the inputs, are taken from [47], which corresponds to a quadcopter which weighs 0.5 kg and has a diameter of half a meter. Bounds on the angles, velocities and body rates are considered, and the dynamics can be brought into the form (1) using a (forward) Euler discretization, such that $x_{k+1} = x_k + T_{\text{opt}}\tilde{f}(x_k, u_k)$, where T_{opt} is the sampling time, and $\tilde{f}(\cdot, \cdot)$ is the continuous-time dynamics that can be obtained from (19a)–(19d).

5.3. Cost function

Our control objective is to navigate the quadcopter as fast as possible, while avoiding excessive control inputs. We combine these competing goals as a weighted sum of the form $J = q\tau_F + \sum_{k=0}^{N-1} u_k^T R u_k$, where τ_F is the final time and $R = R^\top \succeq 0$, and $q \geq 0$ are weighting factors. Motivated by [48], we do not directly minimize τ_F; instead, observing that $\tau_F = NT_{\text{opt}}$, we will treat the discretization time T_{opt} as a decision variable. This allows the use of the slightly modified cost function

$$J(\mathbf{u}, T_{\text{opt}}) = qNT_{\text{opt}} + \sum_{k=0}^{N-1} u_k^T R u_k \tag{20}$$

which will be used in the numerical simulations later on.

Treating T_{opt} as an optimization variable has the additional benefit that the duration of the maneuver does not need to be fixed a priori, allowing us to avoid feasibility issues caused by a too short maneuver lengths. We point out that having T_{opt} as a decision variable comes at the cost of introducing an additional decision variable T_{opt}, which renders the dynamics "more non-linear", which can be seen when looking at the Euler discretization in the previous section.

5.4. Choice of Initial Guess

Recall that (10) and (13) are non-convex optimization problems, and hence computationally challenging to solve in general. In practice, one has to content oneself with a locally optimal solution that, for instance, satisfied the Karush-Kuhn-Tucker (KKT) conditions [42], since most numerical solvers operate locally. Furthermore, it is well-known that the solution quality critically depends on the initial guess ("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 and highly problem dependent; ideally, the initial guess should be obstacle-free and approximately satisfy the system dynamics.

For the quadcopter example, we have observed that the well-known A* algorithm is able to provide good initial guesses. A* is a graph search algorithm that is able to find obstacle-free paths by gridding the position space. It is similar to Dijkstra's algorithm, but uses a so-called heuristic function to perform a "best-first" search, see [49, 50] for details. In our quadcopter example, we use the A* algorithm to find an obstacle-free

path in the position space, which we use to initialize the states that correspond to the quadcopter's position. The remaining states are initialized with zero, while inputs are initialized with the steady state input that keeps the quadcopter in a hoovering position. The dual variables $\lambda_k^{(m)}$ are initialized with 0.05, and the discretization time T_{opt} with 0.25. Fig. 2 depicts the initial guess used to generate the trajectory shown in Fig. 1. Notice that, due to gridding, the path in Fig. 2 exhibits a zigzag pattern.

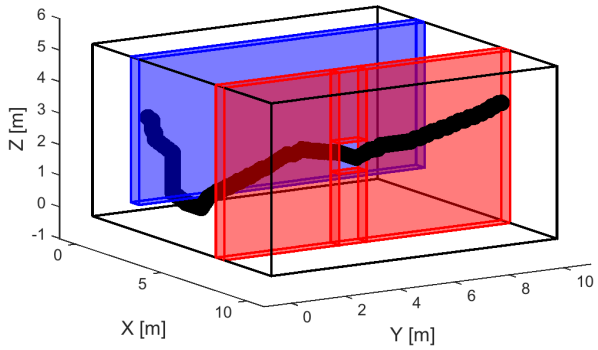

Figure 2: A sample warm start trajectory, obtained from the A* algorithm, is shown. Notice that this trajectory avoids obstacles but does not satisfy the dynamic constraints of the quadcopter, which will be "corrected" by solving (10) and (13). The black tube is the safety distance d_{min} which is used to represent the shape of the quadcopter.

5.5. Simulation Results

To verify the performance and robustness of our approach, we considered 36 path planning scenarios, each starting and ending in a hovering position. The starting point is always located at $(X, Y, Z) = (1, 1, 3)\,\text{m}$, and the finishing point is always located behind the wall with the window at $X = 9\,\text{m}$, but with varying Y and Z coordinates. The final positions are generated by gridding the (Y, Z) space with nine points in the Y direction and four points in the Z direction as shown in Fig. 3. We tested both the distance formulation (10) as well as the signed distance formulation (13). The horizon N equals the number of steps performed by the $A^\star$ algorithm, and takes values between 100 and 129 for the given setup and a grid size of 0.1 m. The sampling time T_{opt} is restricted to lie between 0.125 s and 0.375 s. The optimization problems are implemented with the modeling toolbox JuMP in the programming language Julia [51], and solved using the general purpose nonlinear solver IPOPT [42]. The problems are solved on a 2013 MacBook Pro with an i7 processor clocked at 2.6 GHz.

Table 1 lists the minimum, maximum and average computation time of the A* algorithm, and the time required to solve problems (10) and (13). Fig. 3 reports the solution time as a function of the finishing position, where a circle indicates that IPOPT has successfully found a solution. We see from Fig. 3 that both the distance and signed distance formulation are able to compute all paths successfully. Interestingly, however, the computation time pattern of these two approaches are not correlated; in other words, a "difficult" scenario for the distance formulation might be "easy" for the signed distance formulation, and vice versa. In practice, this implies that to obtain feasible trajectories as fast as possible, the navigation problem should be solved with both obstacle avoidance formulations, and the first solution should be taken. In our setup, such an approach would result in a worst case computation time of 28.9 s, as opposed to 48.0 s and 59.1 s if the distance and signed distance reformulation are considered individually.

We close this section by pointing out that, with a maximum computation time of 2.8 s, the time for A* to find an initial guess is considerably lower than that for solving the optimization problems, see Table 1. This is not surprising since A* only plans a path in the (X, Y, Z)-space and ignores the system dynamics which leads to zigzag behavior, see Fig. 2. A dynamically feasible path is only obtained after solving the optimal control problems (10) and (13) which, by explicitly taking into account system dynamics, smoothen and locally optimize the path provided by the A* algorithm.

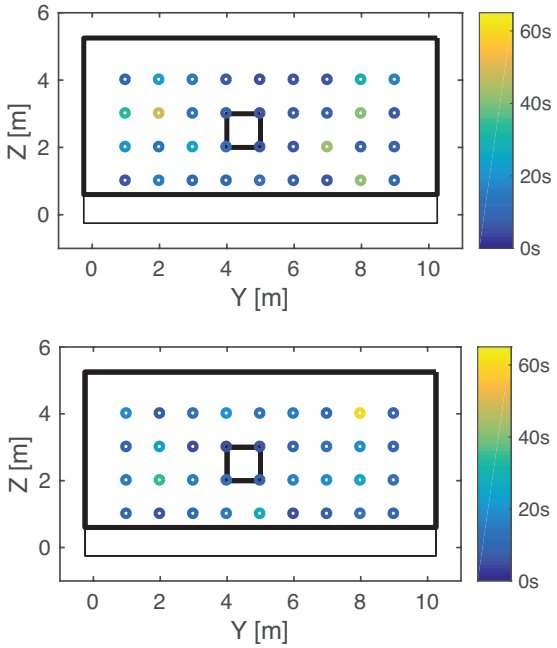

Figure 3: Solution time for quadcopter trajectory planning with distance formulation (10) (top) and signed-distance formulation (13) (right).

Table 1: Computation time of A*, distance formulation (10) and signed distance formulation (13).

Quadcopter navigation	min	max	mean
warm start (A*)	0.5724 s	2.8157 s	1.6207 s
distance formulation (10)	4.6806 s	47.9762 s	14.9716 s
signed distance formulation (13)	4.7638 s	59.1031 s	14.3962 s

6. Example 2: Autonomous Parking

As a second application for our collision avoidance formulation, we consider the autonomous parking problem for self-driving cars. In contrast to the quadcopter case, modeling a car as a point-mass and then approximating its shape with a ball can be very conservative and prevent the car from finding a feasible parking trajectory, especially when the environment is tight. In this section, we model the car as a rectangle, and then employ the full-dimensional formulation described in Section 4. We show that our modelling framework allows us to find obstacle-free parking trajectories even in tight environments. Two scenarios are considered: reverse parking (Fig. 4) and parallel parking (Fig. 5).

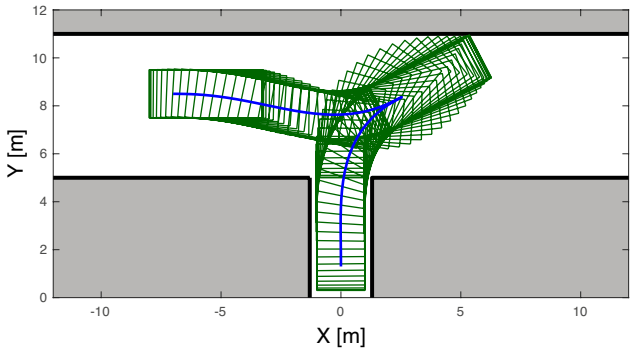

Figure 4: Reverse parking maneuver. The controlled vehicle is shown in green at every time step. Vehicle starts on the left facing to the right, and ends facing upwards, see `https://youtu.be/V7IUPW2qDFc` for an animation.

6.1. Environment and Obstacle Modeling

For the reverse parking scenario, the parking spot is assumed 2.6 m wide and 5.2 m long. The width of the road, where the car can maneuver in, is 6 m, see Fig. 4 for an illustration. For the parallel parking scenario, the parking spot is 2.5 m deep and 6 m long, and the space to maneuver is 6 m wide (Fig. 5). Note that the obstacles in the reverse parking scenario can be described by three axis aligned rectangles, while the obstacles in the parallel parking scenario can be described by four axis aligned rectangles. In both cases,

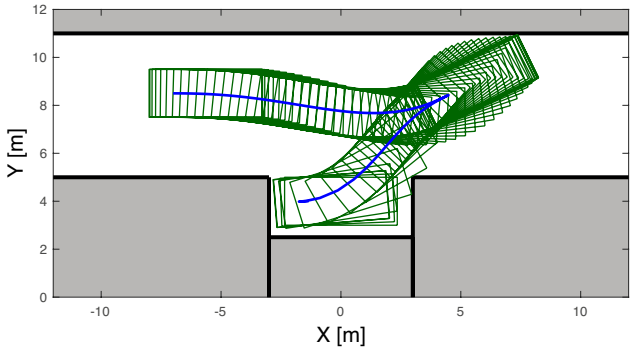

Figure 5: Parallel parking maneuver. The controlled vehicle is shown in green at every time step. Vehicle starts facing to the right, and ends facing to the right, see `https://youtu.be/FST7li4M6lU` for an animation.

the controlled vehicle is modeled as a rectangle of size $4.7 \times 2\,\mathrm{m}$, whose orientation is determined by the car's yaw angle.

6.2. System Dynamics and Cost Function

The car is described by the classical kinematic bicycle model, which is well-suited for velocities used in typical parking scenarios. The states (X, Y) correspond to the center of the rear axes, while φ is the yaw angle with respect to the X-axis, and v is the velocity with respect to the rear axes. The inputs are the steering angle δ and the acceleration a. Hence, the continuous-time dynamics of the car is given by

$$\begin{aligned} \dot{X} &= v\cos(\varphi)\,, \\ \dot{Y} &= v\sin(\varphi)\,, \\ \dot{\varphi} &= \frac{v\tan(\delta)}{L}\,, \\ \dot{v} &= a\,, \end{aligned}$$

where $L = 2.7\,\mathrm{m}$ is the wheel base of the car. The steering angle is limited between $\pm 0.6\,\mathrm{rad}$ (approximately 34 deg), with rate constraints $\dot{\delta} \in [-0.6, 0.6]\,\mathrm{rad/s}$; acceleration is limited to be between $\pm 1\,\mathrm{m/s^2}$. We limit the car's velocity to lie between -1 and $2\,\mathrm{m/s}$. Similar as in the quadcopter case, the continuous-time dynamics are discretized using a forward Euler scheme. Finally, the same cost function as in the quadcopter is used, i.e., a weighted sum between the discretization-time and control effort is considered.

6.3. Initial Guess

Similar to the previous example, the solution quality of the non-convex optimization problems heavily depends on the initial guess provided to the numerical solvers. Unfortunately, it turns out that the A* algorithm used in the quadcopter example generally provides a poor warm start as it is unable to take into

account the vehicle's non-holonomic dynamics[4]. To address this issue, we resort to a modified version of A*, called Hybrid A* [19]. The main idea behind Hybrid A* is to use a simplified vehicle model with states (X, Y, φ), and a finite number of steering inputs to generate a coarse parking trajectory. Like A*, Hybrid A* grids the state space and performs a tree search, where the nodes are expanded using the simplified vehicle model. We refer the interested reader to [19] for details on Hybrid A*. Fig. 6 and Fig. 7 depict two trajectories obtained from the Hybrid A* algorithm. Notice that, due to discretization of state and input, the paths generated by Hybrid A* seems more "bang-bang" and less "smooth" than those shown in Fig. 4 and Fig. 3.

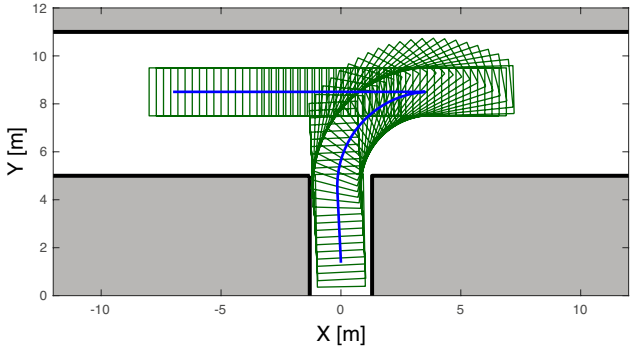

Figure 6: Initial guess provided by Hybrid A* for reverse parking.

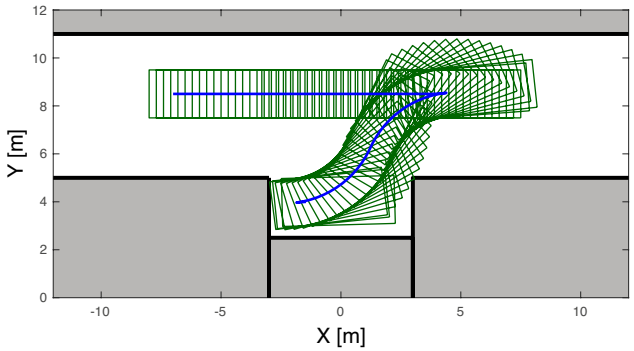

Figure 7: Initial guess provided by Hybrid A* for parallel parking.

6.4. Simulation Results

To evaluate the performance of formulations (15) and (18), we study the reverse and parallel trajectory planning problem. For both cases, we consider different starting positions but one fixed end position at

[4]Roughly speaking, A* will return trajectories that would require the vehicle to move sideways. Simulations indicate that the numerical solvers are typically not able to "correct" such a behavior and unable to recover a feasible solution when initialized with A*.

$X = 0\,\text{m}$, and investigate the computation time of each method. The starting positions are generated by gridding the maneuvering space within $X \in [-10, 10]\,\text{m}$ and $Y \in [6.5, 9.5]\,\text{m}$, with 21 grid points in the X direction and 4 grid points in Y direction, see Fig. 8. The orientation for all the starting points is $\varphi = 0$, resulting in a total of 84 starting points. The horizon length N is given by the Hybrid A* algorithm. The optimization problems are again implemented with the modeling toolbox JuMP in the programming Julia [51], and IPOPT [42] is used as the numerical solver. The problems are solved on a 2013 MacBook Pro with a i7 processor clocked at 2.6 GHz. A Julia-based example code can be found at `https://github.com/XiaojingGeorgeZhang/OBCA`.

We begin by considering the reverse parking case, where one specific maneuver is illustrated in Fig. 4. The computation times for the distance and the signed distance formulation are listed in Table 2 (upper half) and shown in Fig. 8, for all 84 initial conditions. Table 2 indicates that the distance formulation is generally faster than the signed-distance formulation, with a mean computation time of 0.60 s compared to 1.03 s. This is not surprising since the signed distance formulation has more decision variables due to the presence of the slack variables $s_k^{(m)}$, see (18). Furthermore, we see from Fig. 8 that both approaches are able to find feasible parking trajectories, for all 84 considered initial conditions. Interestingly, we see that there are no obvious relations between starting positions and solution times.

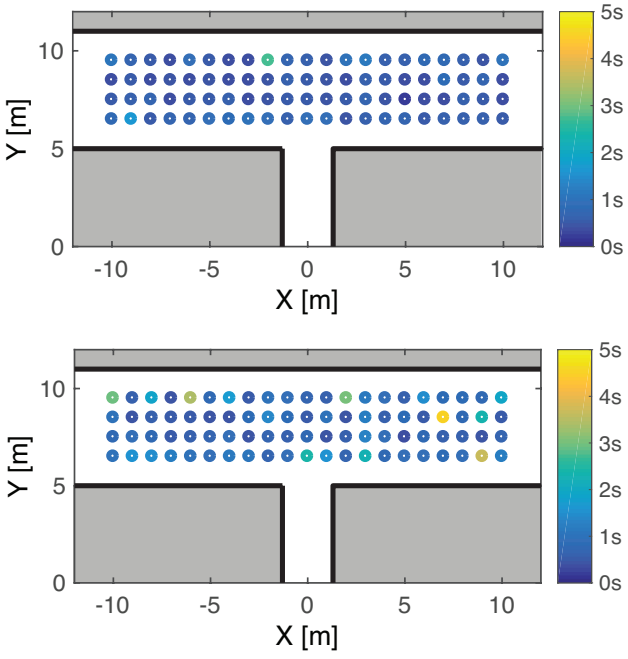

Figure 8: Solution time for reverse parking with distance formulation (15) (top) and signed-distance formulation (18) (bottom).

The computation times of the parallel parking case is shown in Fig. 9 and Table 2 (lower half). Similar

as in the reverse parking case, we see that both approaches have a 100% success rate, and that, again due to the presence of the slack variables, the signed distance formulation requires longer computation time (1.67 s on average) than the distance formulation (0.87 s on average). Compared to reverse parking we see that parallel parking is computationally more demanding. We believe that this is due to the fact that the paths in parallel parking are generally longer than in reverse parking, since the car first needs to drive to the right before it can back into the parking lot, see also Fig. 5.

Table 2: Computation time of Hybrid A*, distance formulation (15) and signed distance formulation (18).

	min	max	mean
Reverse Parking			
warm start (Hybrid A*)	0.0315 s	3.2230 s	0.5491 s
distance formulation (15)	0.2111 s	2.7166 s	0.6046 s
signed distance formulation (18)	0.3200 s	4.4840 s	1.0344 s
Parallel Parking			
warm start (Hybrid A*)	0.0421 s	2.4766 s	0.3012 s
distance formulation (15)	0.2561 s	3.9885 s	0.8682 s
signed distance reformulation (18)	0.3850 s	6.7266 s	1.6703 s

We close this section with the following two remarks: First, we point out that, while the paths generated by the Hybrid A* are collision-free and kinodynamically feasible, they are challenging to track with low-level path following controllers because they do not incorporate information on the velocity and do not take into account the rate constraints in both steering and acceleration, allowing the car to take "aggressive" maneuveures. As demonstrated in [52], this leads, in general, to significantly longer maneuvering times. Second, we notice from Table 2 that the computation time of Hybrid A* is comparable to those of (signed) distance. Furthermore, the maximum overall computation time of Hybrid A* and signed distance reformulation is 7.7 s (reserve parking), and 9.2 s (parallel parking). This implies that, when initialized with Hybrid A*, the proposed collision avoidance framework enables real-time autonomous parking in tight environments.

7. Conclusion

In this paper, we presented smooth reformulations for collision avoidance constraints for problems where the controlled object and the obstacle can be represented as the finite union of convex sets. We have shown that non-differentiable polytopic obstacle constraints can be dealt with via dualization techniques to preserve differentiability, allowing the use of gradient- and Hessian-based optimization methods. The presented reformulation techniques are exact and non-conservative, and apply equally to point-mass and

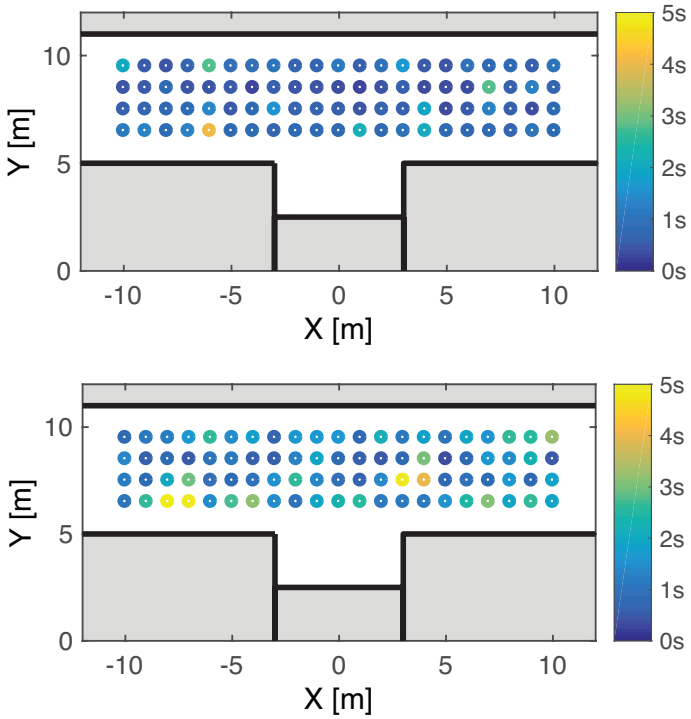

Figure 9: Solution time for parallel parking with distance formulation (15) (top) and signed-distance formulation (18) (bottom).

full-dimensional controlled vehicles. Furthermore, in case collision-free trajectories cannot be generated, our framework allows us to find least-intrusive trajectories, measured in terms of penetration.

Our numerical studies, performed on a quadcopter trajectory planning and autonomous car parking example, indicate that, when appropriately initialized, the proposed framework is robust, real-time feasible, and able to generate dynamically feasible trajectories. Furthermore, we have seen that the initialization method is problem-dependent, and should be chosen depending on the system at hand. Current research focuses on appropriately warm starting the discretization time T_{opt}, as well as on methods for further speeding up computation times.

Appendix: Proof of Proposition 2

We need the following standard results from convex analysis:

Lemma 1. *Let $\mathbb{C} \subset \mathbb{R}^n$ be a compact convex set.*

(i) Then, $\mathcal{H}_{\mathbb{C}}(z) := \{x \in \mathbb{R}^n : z^\top x \leq \max_{y \in \mathbb{C}} y^\top z\}$ is a supporting half space with normal vector z, and

$$\mathbb{C} = \bigcap_{z\,:\,\|z\|=1} \mathcal{H}_{\mathbb{C}}(z), \tag{21}$$

for any norm $\|\cdot\|$.

(*ii*) *Let* $\partial\mathcal{H}_{\mathbb{C}}(z) := \{x \in \mathbb{R}^n \colon z^\top x = \max_{y\in\mathbb{C}} y^\top z\}$ *be the supporting hyperplane with normal vector* z. *Then, for any* $\bar{x} \in \mathbb{C}$, *it holds that*

$$\operatorname{dist}(\bar{x}, \partial\mathcal{H}_{\mathbb{C}}(z)) = \frac{\max_{y\in\mathbb{C}}\{y^\top z\} - z^\top \bar{x}}{\|z\|_*}, \tag{22}$$

where dist$(\cdot,\cdot)$ *is defined as in* (8a).

Proof of Proposition 2

First observe that $\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) = \operatorname{dist}(\mathbb{E}(x), \mathbb{O}^{\complement})$, where $\mathbb{O}^{\complement} \subset \mathbb{R}^n$ denotes the complement of the set $\mathbb{O}$. Using this relationship and recalling the definition of $\operatorname{dist}(\cdot,\cdot)$, it follows from (21) that $\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) = \inf_{\{z\colon \|z\|_*=1\}} \operatorname{dist}(\mathbb{E}(x), \partial\mathcal{H}_{\mathbb{O}}(z))$, where we exploited the fact that (21) holds for any norm[5], and hence also the dual norm $\|\cdot\|_*$. Furthermore, it follows from (22) that, since $\|z\|_* = 1$, $\operatorname{dist}(\mathbb{E}(x), \partial\mathcal{H}_{\mathbb{O}}(z)) = \max_{y\in\mathbb{O}}\{y^\top z\} - z^\top \mathbb{E}(x)$, which allows us to rewrite the penetration function as

$$\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) = \inf_{\{z\colon \|z\|_*=1\}} \{\max_{y\in\mathbb{O}}\{y^\top z\} - z^\top \mathbb{E}(x)\}.$$

To see that the min-max problem is equivalent to (11), we use strong duality of convex optimization to reformulate the inner maximization problem as $\max_{y\in\mathbb{O}}\{y^\top z\} = \min_\lambda\{b^\top\lambda \colon A^\top\lambda = z,\ \lambda \succeq_{\mathcal{K}^*} 0\}$. Hence, $\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) = \inf_{z,\lambda}\{b^\top\lambda - z^\top\mathbb{E}(x) \colon \|z\|_* = 1,\ A^\top\lambda = z,\ \lambda \succeq_{\mathcal{K}^*} 0\} = \inf_\lambda\{(b - A\mathbb{E}(x))^\top\lambda \colon \|A^\top\lambda\|_* = 1,\ \lambda \succeq_{\mathcal{K}^*} 0\}$. Finally, we have $\operatorname{pen}(\mathbb{E}(x), \mathbb{O}) < \mathsf{p}_{\max} \Leftrightarrow \inf_\lambda\{(b - A\mathbb{E}(x))^\top\lambda \colon \|A^\top\lambda\|_* = 1,\ \lambda \succeq_{\mathcal{K}^*} 0\} < \mathsf{p}_{\max} \Leftrightarrow \lambda \succeq_{\mathcal{K}^*} 0 \colon (b - A\,\mathbb{E}(x))^\top\lambda < \mathsf{p}_{\max}$, which concludes the proof.

References

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

[2] 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*, pages 1099–1104 Vol.1, Dec 2004.

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

[4] L. Blackmore and B. Williams. Optimal manipulator path planning with obstacles using disjunctive programming. In *American Control Conference*, June 2006.

[5] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *ASME Dynamic Systems and Control Conference*, 2010.

[6] R. Patel and P. J. Goulart. Trajectory generation for aircraft avoidance maneuvers using online optimization. *AIAA Journal of Guidance, Control and Dynamics*, 34(1):218–230, 2011.

[7] L. Blackmore, M. Ono, and B. C. Williams. Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27(6):1080–1094, Dec 2011.

[5] Geometrically speaking, $\operatorname{pen}(\mathbb{E}(x), \mathbb{O})$ is the minimum distance between $\mathbb{E}(x)$ and any supporting hyperplane $\partial\mathcal{H}_{\mathbb{O}}(z)$ of $\mathbb{O}$.

[8] 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*, pages 127–132, Aug 2013.

[9] 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*, 33(9):1251–1270, 2014.

[10] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[11] U. Rosolia, A. Carvalho, and F. Borrelli. Autonomous racing using learning model predictive control. In *American Control Conference (ACC), 2017*, 2017.

[12] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. *IEEE Transactions on Control Systems Technology*, 25(4):1204–1216, July 2017.

[13] I. E. Grossmann. Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. *Optimization and Engineering*, 3(3):227–252, 2002.

[14] G. Schildbach and F. Borrelli. A dynamic programming approach for nonholonomic vehicle maneuvering in tight environments. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 151–156, June 2016.

[15] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

[16] K. Margellos and J. Lygeros. Hamilton-Jacobi Formulation for Reach-Avoid Differential Games. *IEEE Transactions on Automatic Control*, 56(8):1849–1861, Aug 2011.

[17] K. Margellos and J. Lygeros. Toward 4-D Trajectory Management in Air Traffic Control: A Study Based on Monte Carlo Simulation and Reachability Analysis. *IEEE Transactions on Control Systems Technology*, 21(5):1820–1833, Sept 2013.

[18] M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[19] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[20] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev. Multi-Heuristic A*. *The International Journal of Robotics Research*, 35(1-3):224–243, 2016.

[21] S. M. LaValle and J. J. Kuffner. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[22] J. Ziegler and M. Werling. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *IEEE Intelligent Vehicles Symposium*, pages 787–791, June 2008.

[23] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[24] H. Banzhaf, L. Palmieri, D. Nienhuser, T. Schamm, S. Knopp, and J.M. Zollner. Hybrid curvature steer: A novel extend function for sampling-based nonholonomic motion planning in tight environments. In *IEEE International Conference on Intelligent Transportation Systems*, pages 2239–2246, 2017.

[25] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar. Automatic parallel parking with geometric continuous-curvature path planning. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 465–471, June 2014.

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

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

[28] M. Campbell, M. Egerstedt, How J, 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, 368(1928):4649–4672, 2010.

[29] C. Katrakazas, M. Quddus, W. Chen, and L0 Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416 – 442, 2015.

[30] B. Paden, M. Cáp, S. Yong, D. S. Yershov, and E0 Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles*, 1:33–55, 2016.

[31] D. Gonzlez, J. Prez, V. Milans, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[32] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[33] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pages 4569–4574, May 2011.

[34] Chonhyon Park, Jia Pan, and Dinesh Manocha. ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, pages 207–215, 2012.

[35] Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[36] L. Li, X. Long, and M. A. Gennert. BiRRTOpt: A combined sampling and optimizing motion planner for humanoid robots. In *16th International Conference on Humanoid Robots*, pages 469–476, Nov 2016.

[37] U. Rosolia, S. DeBruyne, and A. Alleyne. Autonomous vehicle control: A nonconvex approach for obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 25(2):469–484, 2017.

[38] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges. Real-Time Motion Planning for Aerial Videography With Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters*, 2(3):1696–1703, 2017.

[39] 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*, 86:11 – 20, 2015.

[40] T.R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[41] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[42] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.

[43] S. Cameron and R. Culley. Determining the minimum translational distance between two convex polyhedra. In *International Conference on Robotics and Automation*, volume 3, pages 591–596, Apr 1986.

[44] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9(6):518–533, Jun 1993.

[45] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[46] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[47] D. Mellinger. *Trajectory Generation and Control for Quadrotors*. PhD thesis, 2012.

[48] F. Fahroo and I. Ross. Direct trajectory optimization by a Chebyshev pseudospectral method. In *American Control Conference*, volume 6, pages 3860–3864, 2000.

[49] 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, 4(2):100–107, July 1968.

[50] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *ACM SIGART Bulletin*, (37):28–29, December 1972.

[51] I. Dunning, J. Joey, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[52] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Autonomous Parking using Optimization-based Collision Avoidance. available online at: `https://drive.google.com/file/d/1LmUUz-lmppCyCVJKbw3A8UqLuZ0ZDByV/view?usp=sharing`, UC Berkeley, 2018.