Closed-Form Minkowski Sum Approximations for Efficient Optimization-Based Collision Avoidance

James Guthrie, Marin Kobilarov, Enrique Mallada

***Abstract*— Motion planning methods for autonomous systems based on nonlinear programming offer great flexibility in incorporating various dynamics, objectives, and constraints. One limitation of such tools is the difficulty of efficiently representing obstacle avoidance conditions for non-trivial shapes. For example, it is possible to define collision avoidance constraints suitable for nonlinear programming solvers in the canonical setting of a circular robot navigating around M convex polytopes over N time steps. However, it requires introducing $(2+L)MN$ additional constraints and LMN additional variables, with L being the number of halfplanes per polytope, leading to larger nonlinear programs with slower and less reliable solving time. In this paper, we overcome this issue by building closed-form representations of the collision avoidance conditions by outer-approximating the Minkowski sum conditions for collision. Our solution requires only MN constraints (and no additional variables), leading to a smaller nonlinear program. On motion planning problems for an autonomous car and quadcopter in cluttered environments, we achieve speedups of 4.8x and 8.7x respectively with significantly less variance in solve times and negligible impact on performance arising from the use of outer approximations.**

I. INTRODUCTION

Motion planning is a central task of most autonomous systems, including robots, drones, and autonomous vehicles. Of the many approaches to motion planning, techniques based on nonlinear programming (NLP) such as direct multiple shooting [1] and direct collocation [2] generally offer the most flexibility in regards to choice of objectives and constraints imposed. As high-quality NLP solvers and supporting automatic differentiation tools have become available, it has become feasible to utilize these optimization-based approaches for real-time motion planning or trajectory generation [3].

Despite the flexibility that NLP solvers provide, it can be difficult to efficiently represent obstacle avoidance constraints. Due to their reliance on gradient and Hessian information, most NLP solvers require the objective and constraints to be twice continuously differentiable expressions. This presents a challenge for collision avoidance constraints which often cannot be represented in smooth closed-forms. We briefly review two viable approaches and discuss their advantages and limitations.

Distance Formulation: Collision avoidance can be viewed as ensuring the minimum distance between an obstacle $\mathcal{O}$ and a vehicle $\mathcal{V}$ is greater than zero. In robotics this would be classified as performing collision checking directly in the workspace [4] as shown in the left subplot of Figure 1. When the obstacle and the vehicle have convex shapes, the distance between these sets can be computed through convex optimization [5]. Using this formulation as a constraint leads to a bi-level NLP for which we lack reliable solvers. By leveraging strong duality [6], it is possible to reformulate the collision avoidance conditions into expressions amenable to a NLP solver. This is done at the expense of introducing new variables and constraints. In practice, the solver performance can be highly sensitive to the initialization of these variables [7] and the resulting increase in problem complexity can be problematic for real-time motion planning in cluttered environments. Similar remarks hold for collision avoidance reformulations based on Farkas' Lemma [8] or polar set representations [9].

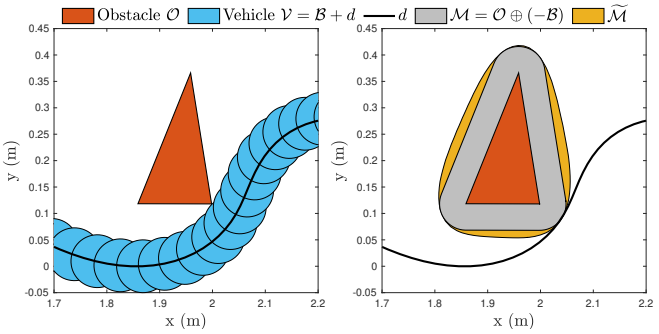

Fig. 1. Obstacle avoidance in workspace (left) and C-space (right)

Minkowski Sum Formulation: Collision avoidance can alternatively be viewed through the lens of computational geometry as shown in the right subplot of Figure 1. Given the vehicle position $d \in \mathbb{R}^n$, and shapes $\mathcal{B}, \mathcal{O} \subset \mathbb{R}^n$ of vehicle and obstacle, respectively, collision avoidance can be posed as ensuring $d \notin \mathcal{M}$ where $\mathcal{M} = \mathcal{O} \oplus (-\mathcal{B})$, with $\oplus$ being the Minkowski sum operation [4]. In robotics this is often referred to as the configuration-space (C-space) approach. Incorporating this as a constraint in an NLP solver would require a closed-form, smooth representation of the indicator function of this set. In general, this does not exist as the sets are semialgebraic, involving multiple polynomial (in)equalities. A notable exception is the case of bodies whose boundary surface are smooth and admit both implicit and parametric representations [10], which includes ellipsoids and convex superquadrics [11]. However, many implicit surfaces do not admit a parametric representation and for others obtaining one is an open problem [12]. Additionally, this approach cannot address the practical case

J. Guthrie and E. Mallada are with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218, USA. Email: {jguthri6, mallada}@jhu.edu.

M. Kobilarov is with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA. Email: marin@jhu.edu

of non-smooth boundaries such as convex polytopes.

A. Contributions

In this work we propose efficient collision avoidance conditions based on closed-form, outer approximations $\widetilde{\mathcal{M}} \supseteq \mathcal{M}$ of the Minkowski sum. We focus on the important case in which the obstacle is a bounded, convex polytope and the vehicle is represented by Euclidean balls (possibly multiple). Building upon recent successes of sum-of-squares (SOS) optimization for outer approximating semialgebraic sets [13]–[17], we develop SOS programs for finding $\widetilde{\mathcal{M}}$. Figure 1 shows an example of the resulting outer approximations. We then use $\widetilde{\mathcal{M}}$ to perform optimization-based motion planning of an autonomous car and quadcopter navigating cluttered environments. Compared to the exact method [6], our approximate method solves 4.8x (car) and 8.7x (quadcopter) faster while introducing minimal conservatism arising from the use of outer approximations.

The rest of the paper is organized as follows. Section II reviews relevant aspects of convex sets, Minkowski sums, and SOS optimization. Section III defines the motion planning problem. Section IV poses the obstacle avoidance constraints using Minkowski sums and provides methods for outer approximating the set. Section V applies our approach to motion planning for an autonomous car and quadcopter. Section VI concludes the paper with a discussion of future directions.

II. PRELIMINARIES

We briefly review some basic properties of convex sets, Minkowski sums, and sum-of-squares polynomials. This is mostly done to setup our notation. The reader is referred to [5], [18], [19] for proofs and further details.

A. Set Definitions

Definition 1 (Convex Hull)**.** The convex hull of a set $\mathcal{B}$ is defined as: $\mathbf{conv}\ \mathcal{B} = \{\theta_1 x_1 + \ldots + \theta_k x_k \mid x_i \in \mathcal{B}, \theta_i \geq 0, i = 1, \ldots, k, \sum_{i=1}^{k} \theta_i = 1\}$. Let $\mathcal{C}$ be any convex set that contains $\mathcal{B}$. The convex hull is the smallest convex set that contains $\mathcal{B}$:

$$\mathcal{B} \subseteq \mathcal{C} \Leftrightarrow \mathbf{conv}\ \mathcal{B} \subseteq \mathcal{C} \tag{1}$$

Definition 2 (α-sublevel Set)**.** The α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$ is: $B_\alpha = \{x \mid f(x) \leq \alpha\}$. We denote the boundary of the set as $\partial B_\alpha = \{x \mid f(x) = \alpha\}$.

Lemma 1. *Let $\mathcal{B}_\alpha$ be a convex set that is the α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$. Then $\mathcal{B}_\alpha = \mathbf{conv}\ \partial\mathcal{B}_\alpha$.*

We use the notation $-\mathcal{B} = \{-b \mid b \in \mathcal{B}\}$ to represent the set $\mathcal{B}$ reflected about the origin. Note that $-\mathcal{B}$ is convex if and only $\mathcal{B}$ is convex.

Definition 3 (Polytope)**.** A polytope $\mathcal{P}$ is defined as the solution set of j linear inequalities in $\mathbb{R}^n$. This set is convex by construction. We impose the additional requirement that the set is bounded. The linear inequalities give the halfspace representation

$$\mathcal{P} = \{x \mid Ax \leq b\} \tag{2}$$

where $A \in \mathbb{R}^{j \times n}, b \in \mathbb{R}^j$. Alternatively, the polytope can be represented by the convex hull of its k vertices

$$\mathcal{P} = \mathbf{conv}\ \{v_1, v_2, \ldots, v_k\} \tag{3}$$

where $v_i \in \mathbb{R}^n, i \in [k] := \{1, \ldots, k\}$.

B. Minkowski Sum Properties

Definition 4 (Minkowski Sum)**.** Given two sets $\mathcal{A}, \mathcal{B}$, their Minkowski sum is defined as follows:

$$\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\} \tag{4}$$

Lemma 2. *If A and B are convex sets then $A \oplus B$ is convex.*

Lemma 3. *For any sets $\mathcal{A}, \mathcal{B}$ the following equality holds:*

$$\mathbf{conv}\ (A \oplus B) = \mathbf{conv}\ (A) \oplus \mathbf{conv}\ (B) \tag{5}$$

C. Sum-of-Squares Optimization

For $x \in \mathbb{R}^n$, let $\mathbb{R}[x]$ denote the set of polynomials in x with real coefficients.

Definition 5 (Sum-of-Squares Polynomial)**.** A polynomial $p(x) \in \mathbb{R}[x]$ is a sum-of-squares (SOS) polynomial if there exists polynomials $q_i(x) \in \mathbb{R}[x], i \in [j]$ such that $p(x) = \sum_{i \in [j]} q_i^2(x)$. We use $\sum[x]$ to denote the set of SOS polynomials in x. A polynomial of degree $2d$ is a SOS polynomial if and only if there exists a positive semi-definite matrix P (the Gram matrix) such that $p(x) = z(x)^T P z(x)$ where $z(x)$ is the vector of all monomials of x up to degree d [19].

Note that a polynomial being SOS is a sufficient condition for the polynomial to be non-negative (i.e. $p(x) \geq 0 \,\forall\, x$).

Definition 6 (SOS-Convex)**.** A polynomial $p(x)$ is SOS-convex if the following holds

$$u^T \nabla^2 p(x) u \in \textstyle\sum[x, u] \tag{6}$$

where $u, x \in \mathbb{R}^n$. SOS-convexity is a sufficient condition for the Hessian of $p(x)$ to be positive semi-definite and therefore $p(x)$ to be convex.

In the development that follows, we will be interested in solving slight variations of the following problem.

$$\min_{P} \quad -\log\det P \tag{7a}$$

$$\text{s.t.}$$

$$P \succeq 0, \quad p(x) = z(x)^T P z(x), \tag{7b}$$

$$1 - p(x) \geq 0 \quad \forall\, x \in \mathcal{X}, \tag{7c}$$

Here $\mathcal{X}$ is a semialgebraic set defined by n_i polynomial inequalities and n_j polynomial equalities.

$$\mathcal{X} = \{x \mid g_i(x) \geq 0, i \in [n_i], h_j(x) = 0, j \in [n_j]\} \tag{8}$$

Equation (7b) constrains $p(x)$ to be a SOS polynomial. Equation (7c) is a set-containment condition. The generalized

$\mathcal{S}$-procedure provides a sufficient condition for the set-containment to hold [19]. For each polynomial equality $g_i(x)$ or inequality $h_j(x)$ describing the set $\mathcal{X}$, we introduce a non-negative polynomial $\lambda_i(x)$ or polynomial $\mu_j(x)$ respectively. The generalized $\mathcal{S}$-procedure involves replacing (7c) with the following:

$$1 - p(x) - \sum_i \lambda_i(x) g_i(x) - \sum_j \mu_j(x) h_j(x) \geq 0, \quad (9)$$

$$\lambda_i(x) \geq 0 \quad i \in [n_i] \quad (10)$$

By replacing the non-negativity constraints in (9), (10) with the more restrictive condition that the expressions be SOS polynomials, we obtain a semidefinite program which is readily solved.

$$\begin{aligned} &\min_{P, \lambda_{[1:n_i]}(x), \mu_{[1:n_j]}(x)} && -\log\det P \\ &\text{s.t.} \\ &P \succeq 0, \quad p(x) = z(x)^T P z(x), \\ &1 - p(x) - \sum_i \lambda_i(x) g_i(x) - \sum_j \mu_j(x) h_j(x) \in \sum[x], \\ &\lambda_i(x) \in \sum[x], \quad i \in [n_i]\,. \end{aligned} \quad (11)$$

Note when a polynomial is listed as a decision variable, e.g., $\lambda_{[1:n_i]}(x)$ and $\mu_{[1:n_j]}(x)$ underneath the $\min$, it is implied that the monomial basis is specified and the coefficients are decision variables.

Remark 1. Representing an equality constraint requires introducing a polynomial $\mu(x)$. In contrast, representing an inequality requires introducing a SOS polynomial $\lambda(x)$ which has a smaller feasible set and creates an additional semidefinite constraint. As such, it is generally advantageous to represent sets using equalities when applying the generalized $\mathcal{S}$-procedure.

In the development that follows we focus on transforming problems of interest into the form of (7). Once in this form, the subsequent application of the generalized $\mathcal{S}$-procedure is mechanical.

III. Problem Statement

We now setup the problem of optimization-based motion planning with collision avoidance constraints. For convenience, our notation closely follows that of [7].

A. Vehicle and Obstacle Models

Consider a vehicle with states $x_k \in \mathbb{R}^{n_x}$ and inputs $u_k \in \mathbb{R}^{n_u}$ at time step k. The dynamics evolve according to $x_{k+1} = f(x_k, u_k)$ where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. The vehicle occupies space in $\mathbb{R}^n$. The vehicle's shape is assumed to be represented by n_b Euclidean balls with radii $r^{(i)}$.

$$\mathcal{B}^{(i)} = \{y \in \mathbb{R}^n \mid \|y\|_2 \leq r^{(i)}\},\ i \in [n_b]. \quad (12)$$

The center of each ball is a function of the vehicle's state as given by $t^{(i)} : \mathbb{R}^{n_x} \to \mathbb{R}^n$. At time index k, the space occupied by ball i is given by:

$$\mathcal{V}^{(i)}(x_k) = \mathcal{B}^{(i)} + t^{(i)}(x_k)\,. \quad (13)$$

The union $\bigcup_i \mathcal{V}^{(i)}(x_k)$ gives the total space occupied by the vehicle at time index k. For ease of exposition, in what follows we focus w.l.o.g. on the case when the vehicle is represented by a single ball and drop the superscript (i).

We assume there are M obstacles present in the environment indexed by $m \in [M]$. Each obstacle $\mathcal{O}^{(m)}$ is a polytope (closed, convex) with $k^{(m)}$ vertices $\{v_1, \ldots, v_{k^{(m)}}\}$ defining the convex hull as in (3). Equivalently represented in halfspace form (2), the obstacle m is defined by $j^{(m)}$ constraints given by $A^{(m)} \in \mathbb{R}^{j^{(m)} \times n}, b^{(m)} \in \mathbb{R}^{j^{(m)}}$.

B. Optimal Control Problem

We consider an optimal control problem of controlling the vehicle over N steps. The vehicle starts at state x_S and must end at final state x_F. Let X, U denote the vector of all states and controls respectively, $X = [x_0^T, \ldots, x_N^T]^T, U = [u_0^T, \ldots, u_{N-1}]^T$. We seek to minimize an objective $l(X, U)$ where $l : X \times U \to \mathbb{R}$. Additionally, the vehicle is subject to n_h constraints given by $h(X, U) \leq 0$ where $h : X \times U \to \mathbb{R}^{n_h}$ and the inequality is interpreted element-wise. We assume that $l(X, U)$ and $h(X, U)$ are continuously differentiable and therefore suitable for nonlinear programming solvers which utilize gradient and Hessian information. Lastly, we enforce collision avoidance constraints between each obstacle and the vehicle. The resulting optimization problem takes the following form:

$$\min_{X, U} \quad l(X, U) \quad (14a)$$

$$\text{s.t.}$$

$$x_0 = x_S, \quad x_N = x_F, \quad (14b)$$

$$x_{k+1} = f(x_k, u_k), \quad k = 0, \ldots, N-1 \quad (14c)$$

$$h(X, U) \leq 0, \quad (14d)$$

$$\mathcal{V}(x_k) \cap \mathcal{O}^{(m)} = \emptyset, \quad k \in [N],\, m \in [M]. \quad (14e)$$

Equation (14e) represents the collision avoidance constraints which are non-convex and non-smooth in general. In [6], the authors provide an exact, smooth reformulation of these constraints. As the distance between two convex shapes can be computed using convex optimization, the authors leverage strong duality to develop necessary and sufficient conditions for a Euclidean ball of radius r to not intersect a given convex shape. This requires introducing dual variables associated with the halfspace constraints representing each obstacle $\lambda_k^{(m)} \in \mathbb{R}^{j^{(m)}}$, $k \in [N]$, $m \in [M]$ and replacing (14e) with the following constraints.

$$\begin{aligned} (A^{(m)} t(x_k) - b^{(m)})^T \lambda_k^{(m)} &> r, \\ \|A^{(m)^T} \lambda_k^{(m)}\|_2^2 &\leq 1, \\ \lambda_k^{(m)} &\geq 0, \\ k \in [N],\, m &\in [M]. \end{aligned} \quad (15)$$

If each obstacle has L halfspace constraints, this method introduces $(2 + L)MN$ constraints and LMN dual variables which can result in a large nonlinear program that

is computationally intensive. In the following, we present a method for approximating the collision avoidance constraints while introducing only MN constraints and no additional variables.

IV. Collision Avoidance via Minkowski Sums

We will utilize Minkowski sums to represent the collision avoidance constraints between a closed, convex polytope obstacle $\mathcal{O} = \{y \in \mathbb{R}^n \,|\, a_i^T y \leq b_i, i \in [L]\}$ and a vehicle with shape given by the Euclidean ball $\mathcal{B} = \{w \in \mathbb{R}^n \,|\, w^T w \leq r^2\}$. We first review a fundamental result from computational geometry.

Lemma 4. *Let $\mathcal{O}$ and $\mathcal{B}$ be sets in $\mathbb{R}^n$. Let $\mathcal{V} = \mathcal{B} + d$ be the set $\mathcal{B}$ translated by $d \in \mathbb{R}^n$. Then the following relation holds:*

$$\mathcal{O} \cap \mathcal{V} \neq \emptyset \Leftrightarrow d \in \mathcal{O} \oplus (-\mathcal{B}) \tag{16}$$

Proof. See, e.g. [4], [20] □

In words, when $\mathcal{B}$ is located at position d, it makes contact with $\mathcal{O}$ if and only if d is in the Minkowski sum $\mathcal{O} \oplus (-\mathcal{B})$. Thus collision avoidance with respect to obstacle $\mathcal{O}$ is equivalent to ensuring $d \notin \mathcal{O} \oplus (-\mathcal{B})$.

When $\mathcal{O}$ is a polytope and $\mathcal{B}$ is a Euclidean ball, the set $\mathcal{O} \oplus (-\mathcal{B})$ is semialgebraic. As such we cannot directly include the condition $d \notin \mathcal{O} \oplus (-\mathcal{B})$ as a constraint in a nonlinear optimization problem which requires closed-form, twice differentiable expressions.

Instead we propose to find an outer approximation $\mathcal{O} \oplus (-\mathcal{B}) \subseteq \widetilde{\mathcal{M}} \subset \mathbb{R}^n$ where $\widetilde{\mathcal{M}}$ is defined as the 1-sublevel set of a function $p : \mathbb{R}^n \to \mathbb{R}$. Recall in our setting the translation of the ball at time index k is a function of the vehicle's state x_k as given by $t : \mathbb{R}^{n_x} \to \mathbb{R}^n$. Collision avoidance with respect to obstacle $\mathcal{O}$ can then be ensured by imposing the constraint $p(t(x_k)) > 1 \Leftrightarrow t(x_k) \notin \widetilde{\mathcal{M}} \Rightarrow t(x_k) \notin \mathcal{O} \oplus (-\mathcal{B}) \Leftrightarrow \mathcal{O} \cap \mathcal{V} = \emptyset$.

If multiple obstacles $\mathcal{O}^{(m)}, m \in [M]$ are present, we repeat this process for each obstacle and denote the associated function as $p^{(m)}(x)$. In our trajectory optimization problem we replace (14e) with MN constraints.

$$p^{(m)}(t(x_k)) > 1, \quad k \in [N], \quad m \in [M]. \tag{17}$$

A. Outer Approximations of the Minkowski Sum

We would like our outer approximations to closely approximate the true set. To do so, we pose an optimization problem in which we minimize the volume of the outer approximation.

$$\begin{aligned} &\min_{p(x)} \quad \text{vol } \widetilde{\mathcal{M}} \\ &\text{s.t.} \\ &1 - p(x) \geq 0 \qquad \forall\, x \in \mathcal{O} \oplus (-\mathcal{B}), \\ &\widetilde{\mathcal{M}} = \{x \,|\, p(x) \leq 1\} \end{aligned} \tag{18}$$

In general we cannot solve this optimization problem. To arrive at a tractable formulation, we apply the generalized $\mathcal{S}$-procedure. We first parameterize the polynomial as $p(x) = z(x)^T P z(x)$ where $z(x)$ is a monomial basis chosen by the user and P is a positive semi-definite matrix of appropriate dimension. For arbitrary polynomials, we lack an expression for minimizing the volume of the 1-level set. Various heuristics have been proposed [13]–[16]. We have found maximizing the determinant of P, as proposed in [15], to work well for the problems herein. The resulting optimization problem is

$$\begin{aligned} &\min_{P} \quad -\log\det P \\ &\text{s.t.} \\ &p(x) = z(x)^T P z(x), \quad P \succeq 0, \\ &1 - p(x) \geq 0 \qquad \forall\, x \in \{y - w \,|\, a_i^T y \leq b_i, \\ &\qquad\qquad\qquad\qquad w^T w \leq r^2, i \in [L]\} \end{aligned} \tag{19}$$

where we have explicitly written the set resulting from the Minkowski sum in terms of y and w along with inequalities that ensure $y \in \mathcal{O}$ and $w \in \mathcal{B}$.

We apply the $\mathcal{S}$-procedure to replace the set-containment condition with a sufficient condition. This requires introducing multipliers $\lambda(y, w)$. We then replace the non-negativity conditions with the sufficient condition that the expression admits a SOS decomposition in terms of variables y and w.

Optimization Problem 1: Outer Approximation

$$\begin{aligned} &\min_{P,\, \lambda_{[0:L]}(y,w)} \quad -\log\det P \\ &\text{s.t.} \\ &p(x) = z(x)^T P z(x), \quad P \succeq 0, \\ &1 - p(y - w) - \lambda_0(y, w)(r^2 - w^T w) \\ &\qquad - \sum_{i=1}^{L} \lambda_i(y, w)(b_i - a_i^T y) \in \sum[y, w] \\ &\lambda_i(y, w) \in \sum[y, w] \quad i = 0, \ldots, L \end{aligned} \tag{OA}$$

The formulation given by (OA) is viable but computationally expensive because the SOS decompositions involve both w and y giving $2n$ free variables for $x \in \mathbb{R}^n$. As we seek higher-order approximations, the monomial basis grows rapidly in size leading to large semidefinite programs. We now develop a computationally cheaper program by leveraging convexity.

B. Convex Outer Approximations of the Minkowski Sum

In developing an efficient method for outer approximating the Minkowski sum, we will utilize the following Lemma.

Lemma 5. *Let $\mathcal{O} \subset \mathbb{R}^n$ be a polytope with K vertices $\{v_i\}, i \in [K]$. Let $\mathcal{B} \subset \mathbb{R}^n$ be a convex set that is the α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$. Let S be any convex set in $\mathbb{R}^n$. Then the following relation holds:*

$$\mathcal{O} \oplus (-\mathcal{B}) \subseteq S \Leftrightarrow \{v_i\} \oplus (-\partial\mathcal{B}) \subseteq S \tag{20}$$

Proof. First represent $\mathcal{O} \oplus (-\mathcal{B})$ in terms of its convex hull.

$$\begin{aligned} \mathcal{O} \oplus (-\mathcal{B}) &= \mathbf{conv}\{v_i\} \oplus \mathbf{conv}(-\partial\mathcal{B}), && \textbf{Lemma 1} \\ &= \mathbf{conv}[\{v_i\} \oplus (-\partial\mathcal{B})], && \textbf{Lemma 3} \end{aligned}$$

Next apply the property of the convex hull (1).

$$\mathbf{conv}[\{v_i\} \oplus (-\partial\mathcal{B})] \subseteq S \Leftrightarrow \{v_i\} \oplus (-\partial\mathcal{B}) \subseteq S$$

□

Lemma 5 provides a more efficient condition for outer approximating the Minkowski sum with a set $\widetilde{\mathcal{M}} = \{x \,|\, p(x) \leq 1\}$ as we only have to consider the vertices of $\mathcal{O}$ and the boundary of $\mathcal{B}$ in our set-containment constraint as follows.

$$1 - p(x) \geq 0 \quad \forall\, x \in v_i \oplus (-\partial\mathcal{B}) \quad i \in [K]. \tag{21}$$

However, it requires the condition that $\widetilde{\mathcal{M}}$ be a convex set. We argue that this is a reasonable constraint as $\mathcal{O} \oplus (-\mathcal{B})$ is itself convex. It is difficult to impose the condition that the 1-sublevel set of $p(x)$, i.e., $\widetilde{\mathcal{M}}$, is convex. Recalling Definition 6, we will instead impose the sufficient condition that the function $p(x)$ be sos-convex.

As done previously, we rewrite the set-containment conditions using the generalized $\mathcal{S}$-procedure. We then replace the non-negativity conditions with SOS conditions.

Optimization Problem 2: Convex Outer Approximation

$$\begin{aligned}
&\min_{P,\, \mu_{[1:K]}(w)} && -\log\det P \\
&\text{s.t.} && \\
&P \succeq 0,\ p(x) = z(x)^T P z(x) && \text{(COA)} \\
&1 - p(v_i - w) - \mu_i(w)(r^2 - w^T w) \in \textstyle\sum[w], i \in [K] && \\
&u^T \nabla^2 p(x) u \in \textstyle\sum[x, u] &&
\end{aligned}$$

Remark 2. The formulation of (COA) is advantageous in that the multipliers $\mu(w)$ do not have to be SOS and they only depend on n free variables ($w \in \mathbb{R}^n$). In contrast, (OA) requires SOS multipliers $\lambda(w, y)$ which depend on $2n$ free variables ($w, y \in \mathbb{R}^n$). The former leads to smaller semidefinite programs which scale better with respect to the dimension n or the complexity of $\mathcal{O}$. This is numerically illustrated in the following example.

C. 2D Example

We generate 1000 random test cases in $\mathbb{R}^2$. For each case we generate a polytope $\mathcal{O}$ with $n \in \{3, 4, \ldots, 12\}$ vertices $v_i \in [-1, 1]^2, i \in [n]$ along with a disk $\mathcal{B}$ with radius $r \in [0, 1]$. We form outer approximations $\widetilde{\mathcal{M}} = \{x \,|\, p(x) \leq 1\}$ of the set $\mathcal{M} = \mathcal{O} \oplus (-\mathcal{B})$ using both (OA) and (COA). For each we consider polynomials $p(x)$ of degree 2, 4 and 6. To assess the accuracy of our outer approximations, we compute the approximation error as $100 \times \frac{\text{Area}(\widetilde{\mathcal{M}}) - \text{Area}(\mathcal{M})}{\text{Area}(\mathcal{M})}$. Table I lists the mean approximation error of the 1000 test cases. Empirically, as we increase the polynomial order, the approximation error is reduced, indicating we are getting better outer approximations. Table II lists the mean solve times. As expected, (COA) has significantly faster solve times than (OA) due to the smaller semidefinite program. Figure 2 provides an example of the results.

Remark. For the case when $p(x)$ is a quadratic, the resulting minimum volume $\widetilde{\mathcal{M}}$ can be found exactly using the semidefinite program for finding the minimum volume outer ellipsoid (MVOE) covering a union of ellipsoids [5]. In this case each ellipsoid is a ball of radius r centered at vertex v_i. Our convex formulation (COA) can be seen as a generalized form of this result. The non-convex case (OA) has a smaller feasible set due to the reliance on SOS multipliers and does not return the minimum volume outer ellipsoid in general. Thus (OA) is only advantageous when seeking non-ellipsoidal approximations.

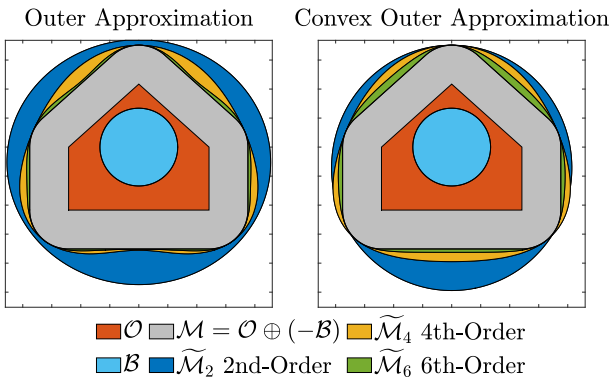

Fig. 2. Outer Approximation of Minkowski Sum

TABLE I

MEAN APPROXIMATION ERROR OF MINKOWSKI SUMS

Polynomial Degree	2	4	6
Outer Approximation	40%	9%	3%
Convex Outer Approximation	25%	9%	5%

TABLE II

MEAN SOLVE TIMES (S) OF OPTIMIZATION PROBLEMS 1 & 2

Polynomial Degree	2	4	6
Outer Approximation	0.020	0.174	0.925
Convex Outer Approximation	0.004	0.014	0.049

V. MOTION PLANNING EXAMPLES

We demonstrate our proposed obstacle avoidance conditions on an autonomous car and quadcopter example. We solve (14) using both the exact representation (15) and the approximate representation (17) of the collision avoidance constraints. We compute the sub-optimality of the approximate method relative to the exact method as $100 \times \frac{J_{approx} - J_{exact}}{J_{exact}}$ where J_{approx}, J_{exact} are the value of the objective function $l(X, U)$ for the respective solutions.

In each example, the dynamic constraints (14c) are implemented using a 4th-order Runge-Kutta integrator with a time-step of $0.02s$ over $N = 150$ steps giving a 3s time horizon. We use $A^\star$ [21] to find a minimum-distance collision free path on a discretized representation of the environment.[1] This path does not consider the dynamics

[1]The $A^\star$ computation time takes an average of 7ms in the case of 10 obstacles for the car example and 41ms for the quadcopter example. In both cases the $A^\star$ step is less than 1/10th the NLP runtime. Bypassing this guess generation and using a naive initialization by linearly interpolating from the initial state x_S to the final state x_F resulted in poor solver reliability.

and is generally not kinematically feasible. We use this to initialize our guess for the vehicle's states over time. The approximate representation utilizes 4th-order, convex polynomials to represent the Minkowski sums. For the exact method, similar to [6], we initialize the dual variables λ to 0.05.

A. Autonomous Car

We adopt the autonomous racing car model from [3]. The model has 6 states, $x = \begin{bmatrix} p_x & p_y & \psi & v_x & v_y & \omega \end{bmatrix}^T$ consisting of position (p_x, p_y), orientation (ψ), body velocities (v_x, v_y) and yaw rate (ω). The inputs are motor duty cycle (d) and steering angle (δ). We represent the vehicle's shape as a single disk $\mathcal{B}$ of radius $r = 0.05$m. The center of the disk at time step k is the $(p_{x,k}, p_{y,k})$ position of the vehicle: $t(x_k) = \begin{bmatrix} p_{x,k} & p_{y,k} \end{bmatrix}^T$.

We consider a situation in which the vehicle is making forward progress along a straight track while navigating obstacles. The objective is to minimize the 2-norm of the input, $l(X, U) = \|U\|_2^2$. The vehicle starts at state $x_S = \begin{bmatrix} 0 & p_{y,S} & 0 & 1 & 0 & 0 \end{bmatrix}^T$ and must end at position $(3, p_{y,F})$. At each step $k = 0, \ldots, N-1$, the vehicle is subject to box constraints on the position $p_{x,k} \in [0, 3], p_{y,k} \in [0, 0.3]$ and inputs $d_k \in [-0.1, 1], \delta_k \in [-1, 1]$.

We consider scenarios consisting of $M \in \{1, 2, \ldots, 10\}$ obstacles. For each scenario, we generate 100 random test cases in which we vary the start and final y position, $p_{y,S}, p_{y,F} \in [0, 0.3]$ along with the placement and shapes of the M obstacles. Figure 3 shows a scenario in which the vehicle navigates ten obstacles. We plot the obstacles along with the exact and approximate Minkowski sums of each obstacle and the vehicle $\mathcal{B}$. As the exact method is equivalent to ensuring the vehicle's position (p_x, p_y) remains outside the exact Minkowski sums $\mathcal{M}$, this helps to visualize the conservatism of our outer approximations. The 4th-order approximate representations $\widetilde{\mathcal{M}}_4$ are quite tight and are only visible as thin yellow borders around the exact Minkowski sums in gray. For reference, we also plot 2nd-order, ellipsoidal approximations $\widetilde{\mathcal{M}}_2$ which are unacceptably conservative as the vehicle cannot progress beyond $p_x = 1.9$ without violating constraints. As the objective penalizes large steering and acceleration commands, the vehicle naturally makes tight maneuvers around the obstacles. The exact method returns a slightly better trajectory because the configuration-space obstacles it must avoid are smaller, requiring less maneuvering. The approximate method makes slightly wider turns, resulting in a 4% sub-optimal trajectory. However the difference is minor and the resulting trajectories are nearly identical.

Figure 4 shows the solve time statistics of the approximate and exact methods as we vary the number of obstacles present. Comparing median solve times, the approximate method solves 1.6x faster than the exact method when just one obstacle is present. With ten obstacles present, the approximate method solves 4.8x faster. The approximate method shows less variability in the solve times, with a maximum solve time of 0.84s and no failed instances.[2] For the exact method, 240 of the 1000 cases either did not converge or exceeded the maximum allowed solve time of 5s. For 746 of the 760 cases in which the exact method was successfully solved, the approximate method returned a trajectory less than 5% sub-optimal. The worst-case sub-optimality was 18%.

TABLE III
SOLVE TIMES STATISTICS FOR QUADCOPTER EXAMPLE

Collision Avoidance	Min. (s)	Median (s)	Max. (s)
Approximate (4th-Order)	0.47	0.74	2.76
Exact	2.37	6.48	18.89

B. Quadcopter

We consider the quadcopter model from [22]. The model has 12 states consisting of position (p_x, p_y, p_z), velocity (v_x, v_y, v_z), Euler angles (ϕ, θ, ψ), and body rates (p, q, r). The inputs are the four rotor speeds $\omega_i, i \in [4]$ in scaled values. We represent the quadcopter's shape as a single ball $\mathcal{B}$ of radius $r = 0.25$m. The center of the ball at time step k is the position of the quadcopter: $t(x_k) = \begin{bmatrix} p_{x,k} & p_{y,k} & p_{z,k} \end{bmatrix}^T$.

We consider a situation in which the quadcopter is navigating a cluttered room with dimensions $10 \times 10 \times 5$. The quadcopter starts at the origin with state $x_S = \mathbf{0}_{12}$ and must end at state $x_F = \begin{bmatrix} p_{x,F} & p_{y,F} & p_{z,F} & \mathbf{0}_9 \end{bmatrix}^T$ while avoiding any obstacles. Here $\mathbf{0}_i$ denotes the zero vector in $\mathbb{R}^i$. The objective is to minimize the 2-norm of the rotor speed deviation from a trim condition $l(X, U) = \|U - 4.5\|_2^2$ where $\omega_i = 4.5, i \in [4]$ achieves a steady-state, hover condition. At each step $k = 0, \ldots, N-1$, the vehicle is subject to box constraints on the position $p_{x,k} \in [0, 10], p_{y,k} \in [0, 10], p_{z,k} \in [0, 5]$ and inputs $\omega_{i,k} \in [1.2, 7.8], i \in [4]$.

The environment contains 30 obstacles. We consider 174 different final positions. Table III lists the resulting solve times of the nonlinear program. The approximate method solved 8.7x faster than the exact method with respect to median solve times. In 9 instances the exact method failed or exceeded the maximum solve time of 20s. The approximate method was less than 5% sub-optimal for 155 of the remaining 165 test cases. The worst-case was 16% sub-optimal. The upper plot of Figure 5 shows the quadcopter navigating the cluttered environment. The lower plot gives the configuration-space view with the Minkowksi sum approximations shown in yellow.

C. Autonomous Car with Multiple-Disc Geometry

The previous examples used vehicle geometries consisting of a single Euclidean ball. We briefly demonstrate how our

[2]The solver times reported for the approximate method only reflect the time spent solving the nonlinear program. We do not include the time required to compute the outer approximations. In a real-time motion planning problem, these approximations would only be performed once per obstacle, either offline or online. We note that based on Table II, approximating an obstacle with a 4th-order convex polynomial takes 0.014s. If this computation time were included in Figure 4, the approximate method would still be consistently faster than the exact method.

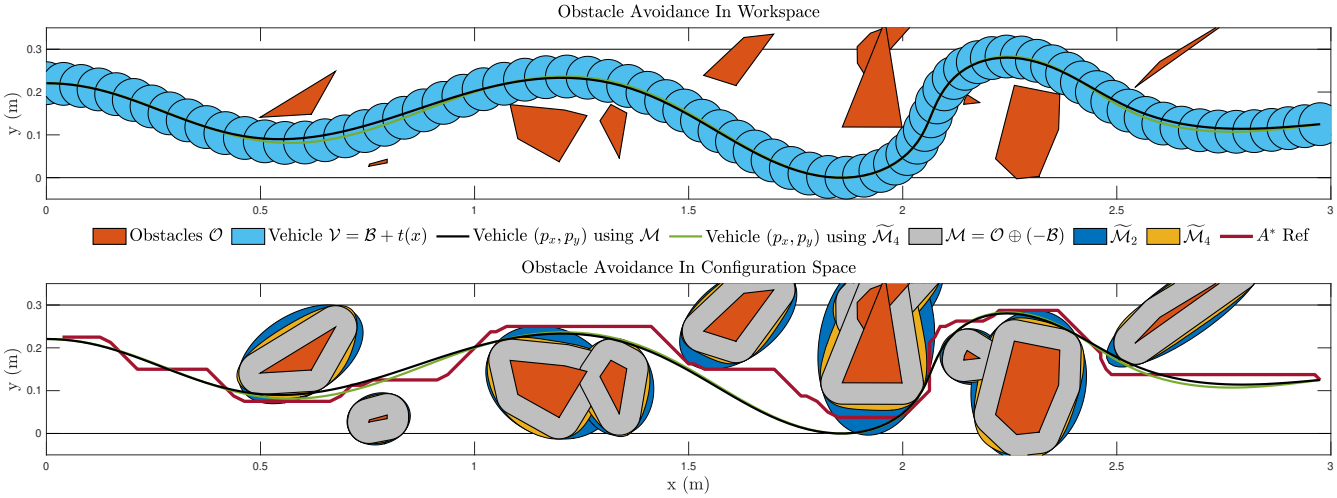

Fig. 3. Autonomous car navigating obstacles in workspace (upper) and configuration space (lower)

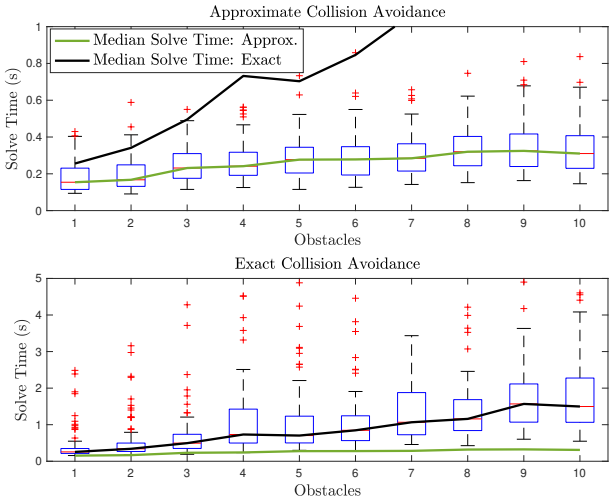

Fig. 4. Solve time statistics for autonomous car example.

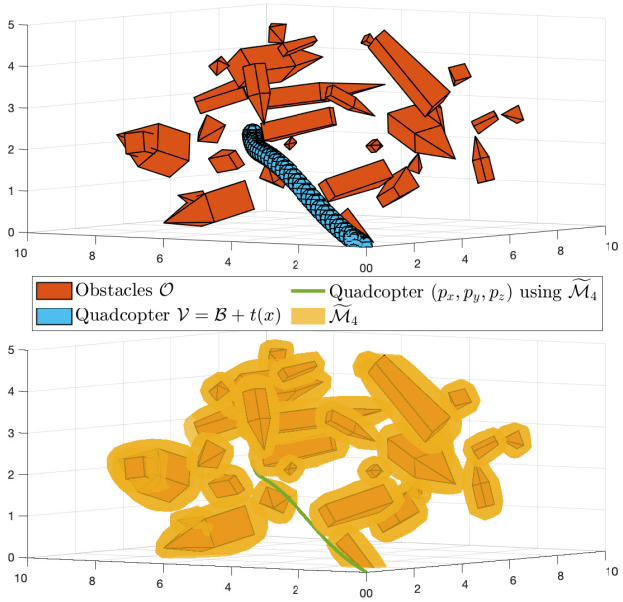

Fig. 5. Quadcopter navigating in workspace (upper) and C-space (lower)

method can handle vehicle geometries consisting of multiple Euclidean balls. Returning to the autonomous car example, we introduce two additional discs, each with radius 0.05m, to form an "L" shape. The center of each disc is a function of the vehicle's position (p_x, p_y) and orientation ψ:

$$t^{(i)}(x) = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} l_x^{(i)} \\ l_y^{(i)} \end{bmatrix} \tag{22}$$

where

$$l_{x,y}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; l_{x,y}^{(2)} = \begin{bmatrix} 0.05 \\ 0 \end{bmatrix}, \; l_{x,y}^{(3)} = \begin{bmatrix} 0 \\ 0.05 \end{bmatrix}, \tag{23}$$

Because the radius of each disc is identical, we can reuse the same approximation $\widetilde{\mathcal{M}} = \{y \in \mathbb{R}^2 \,|\, p^{(m)}(y) \leq 1\}$ for the Minkowski sum of a given obstacle m and a disc of radius 0.05m.[3] We simply change the argument $t^{(i)}(x)$ supplied to p, representing the center of the ball i as a function of the vehicle's state x. In this setting (17) is replaced with the following:

$$p^{(m)}(t^{(i)}(x_k)) > 1, \; k \in [N], \; m \in [M], \; i \in [3]. \tag{24}$$

Figure 6 shows the L-shaped vehicle navigating obstacles.

D. Scaling

One caveat of using SOS polynomials to approximate indicator functions of sets is that the polynomial may return large values for points far outside of the set. This may be problematic for NLP solvers which are often sensitive to the scaling of the problem. To improve the scaling, we can apply any smooth function $q(z) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ to (17) which is strictly-increasing for $z \geq 0$. In the examples shown we have utilized $q(z) = -\exp(-z)$, replacing (17) with:

$$-\exp(-p^{(m)}(t(x_k))) > -\exp(-1), \quad k \in [N], \quad m \in [M]. \tag{25}$$

As p is a SOS polynomial, the left-hand side of this equivalent formulation takes on values in the range $[-1, 0]$.

[3] If this were not the case we would have to compute separate approximations for each unique radius value.

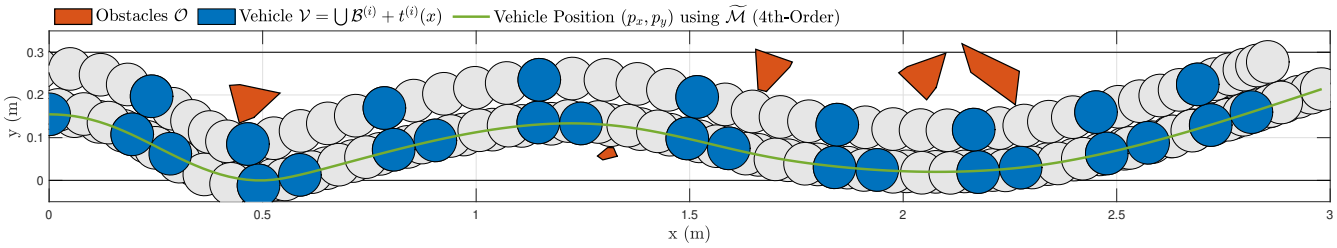

Fig. 6. L-shaped autonomous car navigating obstacles in workspace

E. Implementation Details

All examples were solved on a MacBook Pro with a 2.6 GHz 6-Core Intel Core i7 CPU. YALMIP [23] was used in conjunction with MOSEK [24] to solve the SOS optimization problems. IPOPT [25] with the MA27 linear solver was used to solve the nonlinear optimization problems with exact gradients and Hessians supplied by CasADi [26]. Supporting code is available at https://github.com/guthriejd1/mink-obca.

VI. CONCLUSIONS

This work presented novel obstacle avoidance conditions based on outer approximations of Minkowski sums. This method is advantageous in that it yields a much smaller nonlinear program compared to exactly representing the collision avoidance conditions. On motion planning problems for an autonomous vehicle and quadcopter, the approximate method solved 4.8x and 8.7x faster respectively when navigating cluttered environments. The resulting trajectories exhibited minimal sub-optimality compared to using exact collision avoidance conditions. Currently our method is limited to cases in which the vehicle is represented by a union of Euclidean balls and the obstacle is a bounded, convex polytope. In future work we plan to consider representations of non-convex obstacles.

REFERENCES

[1] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984. 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[2] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers & Chemical Engineering*, vol. 8, pp. 243–247, 1984.

[3] 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.

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

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

[6] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

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

[8] M. Gerdts, R. Henrion, D. Hömberg, and C. Landry, "Path planning and collision avoidance for robots," *Numerical Algebra, Control and Optimization*, vol. 2, no. 3, pp. 437–463, 2012.

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

[10] S. Ruan and G. S. Chirikjian, "Closed-form minkowski sums of convex bodies with smooth positively curved boundaries," *Computer-Aided Design*, vol. 143, p. 103133, 2022.

[11] S. Ruan, K. L. Poblete, Y. Li, Q. Lin, Q. Ma, and G. S. Chirikjian, "Efficient exact collision detection between ellipsoids and superquadrics via closed-form minkowski sums," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1765–1771, 2019.

[12] C. M. Hoffmann, "Conversion methods between parametric and implicit curves and surfaces," 1990.

[13] A. Magnani, S. Lall, and S. Boyd, "Tractable fitting with convex polynomials via sum-of-squares," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 1672–1677, 2005.

[14] F. Dabbene, D. Henrion, and C. M. Lagoa, "Simple approximations of semialgebraic sets and their applications to control," *Automatica*, vol. 78, pp. 110–118, 2017.

[15] A. A. Ahmadi, G. Hall, A. Makadia, and V. Sindhwani, "Geometry of 3d environments and sum of squares polynomials," in *Robotics: Science and Systems*, 2017.

[16] J. Guthrie, "Inner and outer approximations of star-convex semialgebraic sets," *arXiv preprint*, 2022. arXiv:2203.13071.

[17] J. Guthrie and E. Mallada, "Outer approximations of minkowski operations on complex sets via sum-of-squares optimization," in *2021 American Control Conference (ACC)*, pp. 2367–2373, 2021.

[18] R. Schneider, *Convex Bodies: The Brunn–Minkowski Theory*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2 ed., 2013.

[19] P. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[20] M. Berg, de, O. Cheong, M. Kreveld, van, and M. Overmars, *Computational geometry : algorithms and applications*. Germany: Springer, 3rd ed., 2008.

[21] 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.

[22] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[23] J. Lofberg, "Yalmip : a toolbox for modeling and optimization in matlab," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, pp. 284–289, Sep. 2004.

[24] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

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

[26] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.