# Efficient Collision-Avoidance for Multi-Robot System with Superquadric Models and Sum-of-Squares Approximation

Siyi Lu[1], Sipu Ruan*[1]

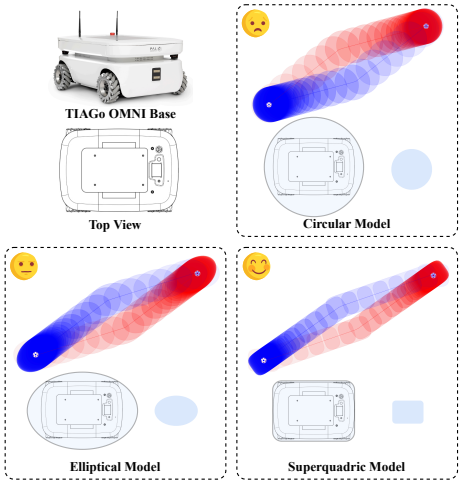


Fig. 1. Illustrative diagram of the path planning, approximating the top view boundary of the TIAGo OMNI Base using the circular model, elliptical model, and superquadric model.

## I. Introduction

The emergence of multi-robot collision-avoidance algorithms has fundamentally transformed the capability of multi-robot systems in safe navigation, thereby facilitating their integration into various real-world applications [1], [2], [3]. Simultaneously, these algorithms are frequently employed as crowd simulation tools in social navigation, significantly influencing the development of social navigation techniques [4], [5], [6], [7]. Traditional multi-agent collision avoidance methods frequently utilize simplified circular models to define both robot and pedestrian boundaries [1], [8]. However, these simplified models may suffer from underestimates of the collision-avoidance space, which makes collision avoidance algorithms conservative.

Recent advancements have introduced sophisticated boundary definitions such as polytypic [9], elliptical [10], [11], or superquadric models [12]. Among them, the superquadric model stands out due to its superior expressiveness, enabling precise representation of a wide range of shapes through adjustable parameters like semi-major and semi-minor axes along with exponent powers. However, the use of superquadric presents significant challenges, particularly concerning the calculation of the velocity obstacle domain.

[1]Siyi Lu, Sipu Ruan are with the Robotics Institute, School of Mechanical Engineering and Automation, Beihang University, China. siyilu@buaa.edu.cn, ruansp@buaa.edu.cn.
*Sipu Ruan is the corresponding author.

Current state-of-the-art methods, proposed in [12], rely on OpenCV's `convexHull` function, which is implemented based on Sklansky's algorithm [13] to compute the velocity obstacle domain. However, Sklansky's algorithm is fraught with inaccuracies and potential errors, as highlighted by Toussaint [14]. These limitations undermine the reliability of existing techniques when applied to real-world scenarios.

To address these challenges, we propose a novel collision-avoidance multi-robot motion planning algorithm **SSCA** (**C**ollision-**A**voidance for Multi-Robot System with **S**uperquadric Models and **S**um-of-Squares Approximation) based on superquadric models and optimization techniques as shown in Fig. 1. Our approach leverages advanced optimization methods to enhance computational efficiency and scalability while maintaining high accuracy.

## II. Research Contents

### A. Problem Formulation

We consider a system of $n$ robots with irregular shapes sharing a 2-D space, where each robot is required to complete its own navigation task in the plane $\mathbb{R}^2$. The robots are modeled using superquadric envelopes, as in [12] and [15]. Let $\mathbf{D}_{A|B} = \mathcal{D}_{r_A} \oplus -\mathcal{D}_{r_B}$ denote the contact space between robot A and robot B, defined as the Minkowski sum of $\mathcal{D}_{r_A}$ and the reflected set $-\mathcal{D}_{r_B}$. Based on $\mathbf{D}_{A|B}$, the VO $\mathcal{VO}^{\tau}_{A|B}$ for robot A induced by robot B within a time horizon $\tau$ is:

$$\mathcal{VO}^{\tau}_{A|B} = \left\{ \mathbf{v} \mid \exists t \in [0, \tau] :: t\mathbf{v} \in T(\mathbf{D}_{AB},\ \mathbf{p}_{r_{B|A}}) \right\}, \quad (1)$$

where, $\mathbf{p}_{r_{B|A}} = \mathbf{p}_{r_B} - \mathbf{p}_{r_A}$ denotes the relative position between robot B and robot A, $T(\mathbf{D}, \mathbf{p})$ denotes the translation of the contact space $\mathbf{D}$ by the vector $\mathbf{p}$. The joint contact space is given by $\mathbf{D}_{AB} = \mathbf{D}_{A|B} \cup \mathbf{D}_{B|A}$, implying that $\mathcal{VO}^{\tau}_{A|B}$ and $\mathcal{VO}^{\tau}_{B|A}$ are symmetric with respect to the origin.

**Theorem 1** (Closed-form Minkowski Sum [15])**.** *The close-form Minkowski sums $\mathbf{D}_{A|B}$ of $\mathcal{D}_{r_A}$ and $\mathcal{D}_{r_B}$ can be expressed as:*

$$\begin{aligned} \mathbf{D}_{A|B} &= \mathcal{D}_{r_A} \oplus -\mathcal{D}_{r_B} \\ &= \mathbf{R}_{r_A} \tilde{\tilde{f}}_{r_A}(\mathbf{m}_{r_A}) \\ &\quad - \mathbf{R}_{r_B} \tilde{\tilde{f}}_{r_B} \left( -\frac{\Psi\left(\mathbf{R}_{r_B}^{\top} \mathbf{R}_{r_A}^{-\top} \mathbf{m}_{r_A}\right)}{\left\| \mathbf{R}_{r_B}^{\top} \mathbf{R}_{r_A}^{-\top} \mathbf{m}_{r_A} \right\|} \mathbf{R}_{r_B}^{T} \mathbf{R}_{r_A}^{-\top} \mathbf{m}_{r_A} \right), \end{aligned} \quad (2)$$

*where $\mathbf{m}_{r_A}$ is un-normalized gradient, $\tilde{\tilde{f}}_{r_A}$ and $\tilde{\tilde{f}}_{r_B}$ map $\mathbf{m}$ to the boundary $\mathcal{D}_{r_A}$ and $\mathcal{D}_{r_B}$. $||\mathbf{m}_{r_B}|| = \Psi(\mathbf{m}_{r_A})$ can be written in closed form, more details can be seen in [15] .*

### B. Sum-of-Square Polynomials for Minkowski Sum Approximation

To construct the VO, we first need to calculate the tangent on $T(\mathbf{D}_{AB},\ \mathbf{p}_{r_{B|A}})$ that goes through the origin $\mathbf{0}$. Differentiating Eq. (2) in its current form is nontrivial. Hence, we seek a differentiable surrogate that approximates it accurately and is easier to handle. Inspired by [16], we use sum-of-squares (SOS) polynomials for this purpose. Let the monomial basis be defined as $[\mathbf{x}]_d = \left[1, x_1, \cdots, x_n, \cdots, x_1^d, \cdots, x_n^d\right]$. Based on the properties of SOS polynomials, we have the following theorem.

**Theorem 2** (SOS polynomial with semidefinite matrix [16])**.** *If $f(\mathbf{x})$ is an SOS polynomial, then there exists a semidefinite matrix $\mathbf{G}$ such that $f(\mathbf{x}) = [\mathbf{x}]_d \mathbf{G} [\mathbf{x}]_d^T$.*

This property enables us to reformulate the polynomial optimization problem as a semidefinite programming (SDP) problem, allowing us to compute an approximate Minkowski sum via SDP as follows.

$$\mathbf{P1:}\ \min_{\mathbf{G}}\ -\log\det\mathbf{G} \tag{3}$$

$$\text{s.t. } 1 - f(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \partial\mathbf{S}_{ms}. \tag{4}$$

$$f(\mathbf{x}) = [\mathbf{x}]_d \mathbf{G} [\mathbf{x}]_d^T. \tag{5}$$

$$\mathbf{G} \succeq 0. \tag{6}$$

where $\partial\mathbf{S}_{ms}$ represents the set of points on the boundary of the closed Minkowski sum $\mathbf{S}_{ms}$, which can be obtained by sampling. Constraint (4) ensures the SOS polynomial bounds the Minkowski sum. Constraints (5) and (6) enforce that $f(\mathbf{x})$ is an SOS polynomial. Since the area enclosed by $f(\mathbf{x})$ is positively correlated with $\det\mathbf{G}$ [17], the objective minimizes $-\log\det\mathbf{G}$. However, the resulting $f(\mathbf{x})$ from problem **P1** may not be convex, complicating subsequent tangent calculations. To address this, we first define convex SOS polynomials and then present the corresponding optimization formulation.

**Definition 1** (Convex SOS polynomial)**.** *A polynomial $f(\mathbf{x})$ is a convex SOS polynomial if the following holds $\mathbf{u}\nabla^2 f(\mathbf{x})\mathbf{u}^T \in \mathcal{P}_{\mathbf{x},\mathbf{u}}$, where $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$. $\mathcal{P}_{\mathbf{x},\mathbf{u}}$ is the set of SOS polynomials whose independent variables are $\mathbf{x}, \mathbf{u}$.*

Definition 2 ensures that $\mathbf{u}\nabla^2 f(\mathbf{x})\mathbf{u}^T$ is an SOS polynomial, such that the Hessian matrix $\nabla^2 f(\mathbf{x})$ of $f(\mathbf{x})$ is positive semi-definite, thereby making $f(\mathbf{x})$ convex. Since $\mathbf{u}\nabla^2 f(\mathbf{x})\mathbf{u}^T$ can be expressed as

$$\mathbf{u}\nabla^2 f(\mathbf{x})\mathbf{u}^T = [\mathbf{x}, \mathbf{u}]_d \mathbf{M} [\mathbf{x}, \mathbf{u}]_d^T, \tag{7}$$

where $[\mathbf{x}, \mathbf{u}]_d = [1, x_1, \cdots, x_n, \cdots, x_1^d, \cdots, x_n^d, u_1, \cdots, u_n, \cdots, u_1^d, \cdots, u_n^d]$. If $\mathbf{u}\nabla^2 f(\mathbf{x})\mathbf{u}^T \in \mathcal{P}_{\mathbf{x},\mathbf{u}}$, we have $[\mathbf{x}, \mathbf{u}]_d \mathbf{M} [\mathbf{x}, \mathbf{u}]_d^T \in \mathcal{P}_{\mathbf{x},\mathbf{u}}$, that is, $\mathbf{M} \succeq 0$. So we transform **P1** into **P2** and get the SDP problem of convex SOS polynomial as follows.

$$\mathbf{P2:}\ \min_{\mathbf{G},\mathbf{M}}\ -\log\det\mathbf{G} \tag{8}$$

$$\text{s.t. } (4), (5), (6), (7), \mathbf{M} \succeq 0.$$

Since the constraints and objective function of the problem **P2** can be expressed using a solver such as Mosek [18], they can be effectively solved. After solving, we obtain an approximate equation $f(\mathbf{x})$ for the Minkowski sum $\mathbf{S}_{ro}$. For convenience, we will use $f(\mathbf{x})$ to represent the approximate Minkowski sum in the following text.

### C. Penalty Function-Based Tangent Lines Computation

For $\mathbf{x} = [x, y] \in \mathbb{R}^2$, given a point $\mathbf{x}_0$ and the approximate equation $f(\mathbf{x})$, to find the tangent line of $f(\mathbf{x})$ at $\mathbf{x}_0$, we first need to compute the gradient $\nabla f(\mathbf{x})$. For the line passing through $\mathbf{x}_0$ to be tangent to $f(\mathbf{x})$, it must satisfy $F_1(x, y) := \frac{\partial f}{\partial x}(x, y)(x - x_0) + \frac{\partial f}{\partial y}(x, y)(y - y_0) = 0$. Moreover, the tangent point $\mathbf{x}$ must also lie on $f(\mathbf{x})$, which is expressed as $F_2(x, y) := f(\mathbf{x}) - 1 = 0$. So, solving the following system of nonlinear equations will yield a point $\mathbf{x}$ on $f(\mathbf{x})$ such that the line connecting the points $\mathbf{x}$ and $\mathbf{x}_0$ is tangent to $f(\mathbf{x})$

$$\begin{cases} \frac{\partial f}{\partial x}(x, y)(x - x_0) + \frac{\partial f}{\partial y}(x, y)(y - y_0) = 0, \\ f(\mathbf{x}) - 1 = 0. \end{cases} \tag{9}$$

We propose using the Newton-Raphson method [19] to solve these higher-degree nonlinear equations. Using gradient information of a function can speed up the search for feasible solutions compared to directly searching on the Minkowski sum. To ensure finding two different tangents, we transform the problem of solving nonlinear equation system (9) into optimization problems **P3** and **P4**.

$$\mathbf{P3/P4:}\ \min_{(x,y)} \sum_{i=1}^{2} ||F_i(x, y)||^2 \tag{10}$$

$$\text{s.t. } y - y_0 - \frac{y_0 - y_c}{x_0 - x_c}(x - x_0) > 0 \text{ (of } \mathbf{P3}) \tag{11}$$

$$y - y_0 - \frac{y_0 - y_c}{x_0 - x_c}(x - x_0) < 0 \text{ (of } \mathbf{P4}) \tag{12}$$

where $(x_c, y_c)$ is the center of $f(\mathbf{x})$. Problems **P3** and **P4** aim to compute the points $(x, y)$ on either side of the line connecting the center of the approximate Minkowski sum and a given point that minimize $\sum_{i=1}^{2} ||F_i(x, y)||^2$. Since we know in advance that these two tangent points must lie on opposite sides of this line, solving problems **P3** and **P4** will yield the corresponding two tangents.

The solutions to problems **P3** and **P4** can be obtained using the penalty function method. By introducing a constant $\beta$, these constrained optimization problems are transformed into unconstrained optimization problems, namely problems **P5** and **P6**.

$$\mathbf{P5:}\ \min_{(x,y)} \sum_{i=1}^{2} ||F_i(x, y)||^2 + \frac{1}{\beta}\log[-(y - y_0 - \frac{y_0 - y_c}{x_0 - x_c}(x - x_0))]. \tag{13}$$

$$\mathbf{P6:}\ \min_{(x,y)} \sum_{i=1}^{2} ||F_i(x, y)||^2 + \frac{1}{\beta}\log[y - y_0 - \frac{y_0 - y_c}{x_0 - x_c}(x - x_0)]. \tag{14}$$

Thus, by solving problems **P5** and **P6**, we can effectively find the two distinct tangent points required.

### D. Solving for Minimum Velocity Change

After obtaining the two contact points $\mathbf{p}_a = (x_a, y_a)$ and $\mathbf{p}_b = (x_b, y_b)$ of the velocity collision domain, we draw a straight line $l_{ab}$ passing through these points. Let $\mathcal{P}(\mathbf{p}, \mathbf{p}_a, \mathbf{p}_b) = [(\mathbf{p} - \mathbf{p}_a) \times (\mathbf{p}_b - \mathbf{p}_a)] \cdot [(\mathbf{p}_0 - \mathbf{p}_a) \times (\mathbf{p}_b - \mathbf{p}_a)]$. The boundary of the velocity collision domain $f_{vo}(x, y) \leq 1$ is composed of two segments: the Minkowski sum boundary on the origin side of the straight line $l_{ab}$ and the tangent lines on the other side. If the relative velocity $(x_p, y_p)$ falls within the velocity collision domain, finding the position on the curve $f_0(x, y) = 1$ of the Minkowski sum boundary on the origin side of the straight line $l_{ab}$ that is closest to $(x_p, y_p)$ is equivalent to solving **P7**.

$$\textbf{P7:} \min_{\mathbf{p}=(x,y)} \ (x - x_p)^2 + (y - y_p)^2 \tag{15}$$

$$\text{s.t. } f_0(x, y) = 1 \tag{16}$$

$$\mathcal{P}(\mathbf{p}, \mathbf{p}_a, \mathbf{p}_b) \geq 0 \tag{17}$$

where $\mathbf{p} = (x, y)$, $\mathbf{p}_0$ is the origin.

Next, solve another problem: finding the position on the two tangent lines $f_1(x, y) = 1$ and $f_2(x, y) = 1$ on the other side of the straight line $l_{ab}$ that is closest to $(x_p, y_p)$ is equivalent to solving **P8** and **P9**.

$$\textbf{P8/P9:} \min_{\mathbf{p}=(x,y)} \ (x - x_p)^2 + (y - y_p)^2 \tag{18}$$

$$\text{s.t. } f_1(x, y) = 1 \text{ (of } \mathbf{P8}) \tag{19}$$

$$f_2(x, y) = 1 \text{ (of } \mathbf{P9}) \tag{20}$$

$$\mathcal{P}(\mathbf{p}, \mathbf{p}_a, \mathbf{p}_b) \leq 0 \text{ (of } \mathbf{P8}) \tag{21}$$

$$\mathcal{P}(\mathbf{p}, \mathbf{p}_a, \mathbf{p}_b) \leq 0 \text{ (of } \mathbf{P9}) \tag{22}$$

Note that the objective function $(x-x_p)^2+(y-y_p)^2$ in **P7** is convex, while the constraint $f_0(x, y) = 1$ is non-convex. Therefore, **P7** is an optimization problem with non-convex equality constraints. However, the constraint $f_0(x, y) = 1$ can be modified to $f_0(x, y) \geq 1$. Since $f_0(x, y) \geq 1$ has strong convexity (as we added a convex constraint in **P2**), the modified problem becomes a convex problem. It is known that the optimal solution must lie on the curve, so the optimal solution remains unchanged. The problem is thus transformed into

$$\textbf{P10:} \min_{\mathbf{p}=(x,y)} \ (x - x_p)^2 + (y - y_p)^2 + \frac{1}{\beta}\log[\mathcal{P}(\mathbf{p}, \mathbf{p}_a, \mathbf{p}_b)]$$

$$\text{s.t. } f_0(x, y) \geq 1, \tag{23}$$

**P10** can be solved using a general solver or gradient descent method. Then, compare the minimum velocity changes obtained from solving **P8**, **P9**, and **P10** separately, and select the smallest velocity change as the final velocity change.

## III. Simulation Results

### A. Navigation Travel Time and Average Speed

The detailed results are reported in Table I, where the best-performing values in each scenario are highlighted in bold and underlined. The results show that when the number of robots was two, all three algorithms achieved comparable travel times. As the number of robots increased, however, SSCA consistently outperformed the baselines. In the four-robot scenarios, SSCA reduced travel time by up to 6.67% relative to ORCA and 15.75% relative to RL-RVO. In the six-robot scenarios, the reductions were 9.27% and 27.45%, respectively. The improvement was most pronounced in the eight-robot cases, where SSCA achieved up to 11.97% lower travel time compared with ORCA and 33.32% lower compared with RL-RVO. These results indicate that the efficiency advantage of SSCA grows with increasing robot density.

TABLE I

Numerical Comparison of Trajectory Length and Average Speed between SSCA and Baseline Algorithms (ORCA and RL-RVO) under Different Number of Robots ($N$) and Aspect Ratio ($a_{r,1}/a_{r,2}$)

| Algorithms | $N = 6, a_{r,1}/a_{r,2} = 7$ | | $N = 6, a_{r,1}/a_{r,2} = 10$ | |
|---|---|---|---|---|
| | Travel Time | Average Speed | Travel Time | Average Speed |
| ORCA | 111.16±10.67 | 0.62±**0.06** | 114.5±17.05 | 0.62±**0.07** |
| RL-RVO | 140.85±22.49 | 0.51±0.14 | 131.54±28.84 | 0.51±0.18 |
| SSCA (ours) | **108.89**±**6.49** | **0.65**±0.07 | **112.5**±**10.85** | **0.69**±0.14 |
| Algorithms | $N = 8, a_{r,1}/a_{r,2} = 7$ | | $N = 8, a_{r,1}/a_{r,2} = 10$ | |
| | Travel Time | Average Speed | Travel Time | Average Speed |
| ORCA | 124.69±24.05 | 0.62±**0.13** | 120.74±15.24 | 0.61±**0.08** |
| RL-RVO | 155.96±30.18 | 0.43±0.16 | 172.75±47.65 | 0.36±0.13 |
| SSCA (ours) | **117.33**±**14.93** | **0.72**±0.14 | **115.20**±**10.32** | **0.76**±0.17 |

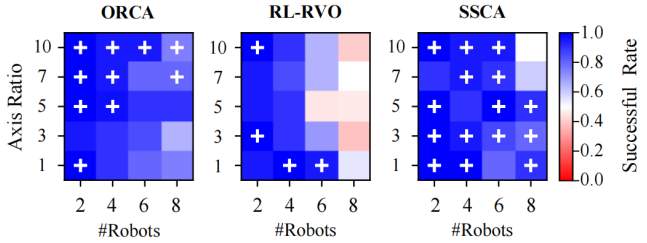


Fig. 2. The success rates of three multi-robot navigation algorithms.

### B. Navigation Success Rate

The results are summarized in Fig. 2, where a color gradient from red to blue indicates success rates from 0 to 1. The simulation varied both the number of robots and the aspect ratios of their superquadric models. RL-RVO consistently yielded the lowest success rates, dropping below 50% in some cases. This poor performance is due to a fundamental mismatch between its circular agent model and the precise goal-reaching requirements of the superquadric-based environment. In contrast, SSCA demonstrated a clear advantage over ORCA, achieving equal or higher success rates in 16 out of 20 scenarios. This improvement is attributed to SSCA's more accurate shape representation, which effectively mitigates the deadlocks commonly induced by circular approximations.

## References


[1] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*, 2011, pp. 3–19.

[2] B. J. Nelson, "Advancing intelligent robotics for a smarter future: Bridging technology and human potential," *SmartBot*, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:277539653

[3] J. Snape, J. v. d. Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[4] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 007–10 013.

[5] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022.

[6] D. M. Baselga, L. Riazuelo, and L. Montano, "Improving robot navigation in crowded environments using intrinsic rewards," in *IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2023, pp. 9428–9434.

[7] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 3517–3524.

[8] Z. Liu, Z. Jiang, T. Xu, H. Cheng, Z. Xie, and L. Lin, "Avoidance of high-speed obstacles based on velocity obstacles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7624–7630.

[9] J. Huang, J. Zeng, X. Chi, K. Sreenath, Z. Liu, and H. Su, "Velocity obstacle for polytopic collision avoidance for distributed multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3502–3509, 2023.

[10] A. Best, S. Narang, and D. Manocha, "Real-time reciprocal collision avoidance with elliptical agents," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 298–305.

[11] B. H. Lee, J. D. Jeon, and J. H. Oh, "Velocity obstacle based local collision avoidance for a holonomic elliptic robot," *Auton. Robots*, vol. 41, no. 6, p. 1347–1363, Aug. 2017.

[12] L. Zhao, J. Zhao, Z. Liu, D. Yang, and H. Liu, "Solving the real-time motion planning problem for non-holonomic robots with collision avoidance in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 510–10 517, 2022.

[13] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79–83, 1982.

[14] G. T. Toussaint and H. El Gindy, "A counterexample to an algorithm for computing monotone hulls of simple polygons," *Pattern Recognition Letters*, vol. 1, no. 4, pp. 219–222, 1983.

[15] 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.

[16] J. Guthrie, M. Kobilarov, and E. Mallada, "Closed-form minkowski sum approximations for efficient optimization-based collision avoidance," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3857–3864.

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

[18] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.*, 2024. [Online]. Available: http://docs.mosek.com/latest/toolbox/index.html

[19] T. J. Ypma, "Historical development of the newton–raphson method," *SIAM review*, vol. 37, no. 4, pp. 531–551, 1995.