

A Linear and Exact Algorithm for Whole-Body Collision Evaluation via Scale Optimization

Qianhao Wang [†], Zhepei Wang [†], Liua Pei, Chao Xu, and Fei Gao

Abstract—Collision evaluation is of essential importance in various applications. However, existing methods are either cumbersome to calculate or not exact. Therefore, considering the cost of implementation, most whole-body planning works, which require evaluating collision between robots and environments, struggle to tradeoff between accuracy and computationally efficiency. In this paper, we propose a zero-gap whole-body collision evaluation that can be formulated as a low-dimensional linear programming. This evaluation can be solved analytically in linear complexity. Moreover, the method provides gradient efficiently, making it accessible to optimization-based applications. Additionally, this method provides support for obstacles represented by either points or hyperplanes. Experiments on the widely used aerial and car-like robots validate the versatility and practicality of our method.

I. INTRODUCTION

Collision Evaluation is critical in a variety of fields, such as physics engines, computer graphics and robot navigation. In recent years, with the development of autonomy, an increasingly large number of robots are deployed in complex real-world scenarios, where they may be requested to navigate through dense and highly dynamic environments, as illustrated in Fig.7, or even to cross narrow gaps of similar size to themselves as shown in Fig.1. These planning problems where the robot’s shape has to be taken into account, namely whole-body planning, urgently require exact and efficient collision evaluation of robots and environments.

Generally, there are several expectations for a collision evaluation method. (1) Exactitude: no gap or approximation with the true value is expected. (2) Efficiency: low computational overhead is required. (3) Locality: instead of a rigorous enumeration of each obstacle [1, 2], the method only focus on a small number of obstacles around the robot.

Vast different approaches [1]–[5] have been introduced in response to the above mentioned requirements. However, when applied to whole-body robot motion planning, they may suffer from difficulty in gradient calculation and struggle to deal with different map representations. In detail, there are some intractable challenges for collision evaluation in whole-body planning. (1) The method should be capable of providing precise gradients efficiently, which allows it to be applied in optimization-based frameworks. (2) With the progress of environment reconstruction and SLAM technology, various map representations such as point clouds [6], surfaces [7], grid maps [8] and semantic information [9], are active in

[†] **Equal contribution.**

All authors are with the College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027, China, and also with the Huzhou Institute of Zhejiang University, Huzhou, 313000, China. Email:{qhwaagaa, wangzhepei, fgaoaa}@zju.edu.cn

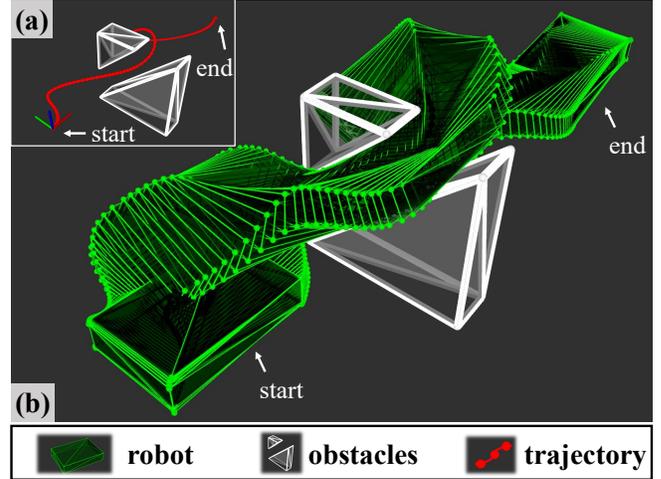


Fig. 1. The application of a whole-body SE(3) planning problem which requires the multicopter to fly from the start to the end. The green rectangles are the snapshots of the multicopter which flies along the red trajectory to cross the narrow gap built by the two white obstacles.

different planning works. The ability to deal with different map representations that may contain lots of redundant information without overly complex pre-processing is required eagerly. (3) It is unacceptable to make the dimension of the problem increase dramatically for collision avoidance, such as Zhang’s work [10] that introduces numerous dual variables in trajectory optimization. Because this can lead to the optimization becoming hard to solve.

In this paper, we propose a novel method that analytically evaluates the collisions of two convex objects, addressing the above requirements and challenges. This method evaluates collision by calculating a minimum scale. One of the objects enlarged or reduced by this scale can have a collision with another object, as shown in Fig.2. This method works seamlessly with objects represented by either points or hyperplanes, which we then abbreviate with **V**(ertex)-representation and **H**(yperplane)-representation (detailed in Sec. III). Another highlight of our method is that it works directly without pre-processing even for redundant points or hyperplanes. Then we formulate the scale calculation into a low dimensional linear programming (LP) whose computational time is $O(m)$ [11], where m is the sum of the number of points or hyperplanes that make up objects, which is also the total number of the linear inequalities in the LP. Regardless of the number of obstacles, this method uses only one variable, the minimum scale $\beta \in \mathbb{R}_{\geq 0}$ defined in Sec.III, to evaluate the collision with the environment.

When applied to whole-body trajectory optimization, with the active constraints of LP, this method provides gradient of the scale β w.r.t. the ego-motion of the scaled object analytically (detailed in Sec. IV). Even when using a naive trajectory that does not take into account the robot's shape as initial values, a safe whole-body trajectory can be obtained after optimization with our method. Finally, to verify the generality and practicality, we apply the proposed method to generate SE(3) aerial robot trajectories based on the differential flatness of multicopter, and plan 2-d car-like robot trajectories considering nonholonomic constraints. The major contributions of this paper are summarized as:

- We propose an exact and rapid collision evaluation method that supports locality, via low-dimensional LP.
- We implement the method in V-representation and H-representation and derive the analytic gradient for whole-body trajectory optimization.
- We validate our method on several challenging cases, SE(3) aerial robots and 2-d car-like robots, testing in both static and dynamic environments.

II. RELATED WORK

A. Collision Evaluation

The simplex-based iterative approaches Gilbert-Johnson-Keerthi (GJK) [3] and enhancing GJK [4] have been widely utilized to calculate the distance between two convex objects. However, these algorithms demand pre-processing to compute support functions. When two objects intersect, the computational cost of GJK has to consider the expanding polytope algorithm (EPA) [5]. Gilbert et al. [1] propose a growth distance to measure collision. Similarly, Tracy et al. [2] calculate the minimum scale that both objects enlarge or reduce simultaneously for an intersection to exist. They formulate the scale computation as a conic problem solved by primal-dual interior-point approaches, which are complex to solve and get the gradient. Additionally, both methods do not support locality. It means that when we demand to evaluate the collision of a robot with many obstacles, both methods require us to perform calculations with all the obstacles. This leads to an increase in computational overhead as obstacles increases. Moreover, the methods encounter problems with unstable values when one of the objects is much larger than the other. Recently, Lutz et al. [12] propose a constructive solid geometry method based on two-layer LogSumExp functions. But it has a gap with the true value.

B. Whole-Body Trajectory Optimization

As stated in Sec. I, to achieve low-cost collision avoidance, there are several brilliant efforts in the field of robot trajectory optimization considering the whole-body shape.

Extensive works [8, 13] model aerial robots as spheres. For car-like robots, Li [14] and Ziegler [15] treat with several circles intuitively. Simply inflating the obstacles according to the robot's radius, they bound the centers of the circles or spheres in the free space of the inflate map for safety. To generate an SE(3) trajectory that enables the drone to perch on a moving platform, Ji et al. [16] model it as a disc to

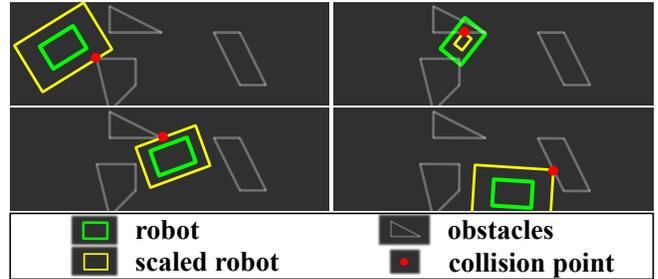


Fig. 2. This figure illustrates how the robot's corresponding scaled robot changes when it moves through a map made up of some obstacles.

evaluate collision. However, neither the conservative methods [8, 13]–[15] nor the task-specific method [16] can be applied in narrow environments where the physical robot shape should be considered. Liu et al. [17] propose an SE(3) planner which formulates the quadrotor as an ellipsoid to cross narrow gaps. As the distance between obstacle and ellipsoid is hard to obtain, they adopt to generate motion primitives and check safety along every primitive. Nonetheless, this search-based method has to raise the primitive resolution to improve the success rate in complex environments, which leads to an explosion in computational overhead. Zhang et al. [10, 18] use differentiable dual variables to formulate the distance between objects to achieve optimization-based collision avoidance (OBCA). Whereas, since the number of dual variables is related to the number of obstacles, the dimension of the problem rises considerably when the obstacles increases. Wang [7] and Han [19] generate safe SE(3) trajectories through optimization with explicit spatial constraints. They generate a series of convex polyhedrons based on the free space as flight corridors and model the robot as a convex polyhedron according to its shape, ensuring trajectory safety by constraining the robot convex polyhedron in the flight corridor. Similarly, for vehicles, Ding [20] and Manzinger [21] construct safe corridor by rectangles. Both the corridor-based methods require the intersection of two adjacent polyhedrons of corridor to contain at least one robot polyhedron. However, when this demanding request is not satisfied, there is no more feasible solution.

III. PROBLEM DEFINITION

We will present the problem definition of calculating the minimum scale in V-representation and H-representation respectively. For ease of presentation, we will refer to the scaled object as the **body** and to the other object as the **obstacle** below.

A. V-representation

In V-representation, as shown in Fig.3(a), both the yellow and blue objects are defined by a convex hull that can contain the redundant points. However, instead of processing points into a convex hull, the proposed method is capable of working directly on the redundant points. For the simplicity of visualization, we do not visualize points that are redundant for representing object in Fig.3(b).

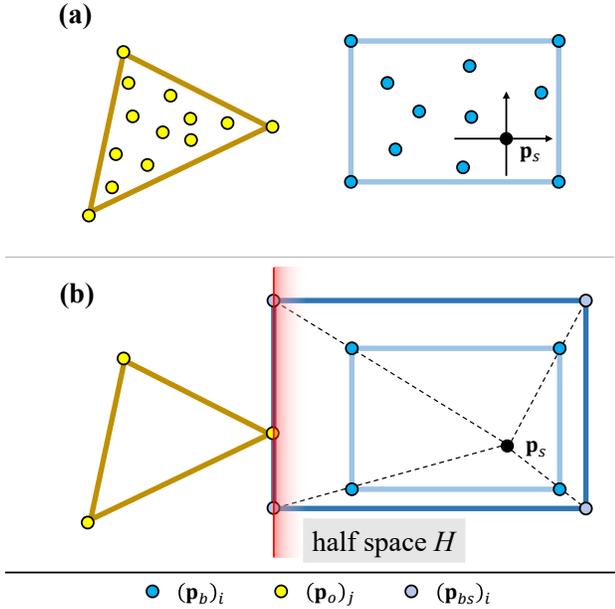


Fig. 3. This figure illustrates the problem definition in V-representation. (a): The obstacle and the body are defined by the yellow and blue convex hull respectively. We use the yellow and blue redundant points to represent the body and the obstacle respectively. (b): To prevent confusion in the visualization, the redundant points are not illustrated. The light blue points represent the scaled body and the red line denotes the half space containing the scaled body but not the obstacle.

As illustrated in Fig.3(a), we use point sets $P_{body}^b = \{(\mathbf{p}_b^b)_i | i = 1, 2, \dots, n_b\}$ and $P_{obs}^b = \{(\mathbf{p}_o^b)_j | j = 1, 2, \dots, n_o\}$ to represent body and obstacle in the body frame. We define a point \mathbf{p}_s in the body frame as scale seed point, which the body point sets scale about. We define $\beta \in \mathbb{R}_{\geq 0}$ as the scale. Then we get obstacle and scaled body point sets in the coordinate system with the point \mathbf{p}_s as the origin as

$$\begin{aligned} P_{body}^s(\beta) &= \{(\mathbf{p}_{bs}^s)_i = \beta((\mathbf{p}_b^b)_i - \mathbf{p}_s^b) | i = 1, 2, \dots, n_b\}, \\ P_{obs}^s &= \{(\mathbf{p}_o^s)_j = (\mathbf{p}_o^b)_j - \mathbf{p}_s^b | j = 1, 2, \dots, n_o\}, \end{aligned} \quad (1)$$

Then we define the problem of calculating the minimum scale in V-representation to maximize the scale β with the constraints of

$$\begin{cases} \alpha_o^T \beta ((\mathbf{p}_b^b)_i - \mathbf{p}_s^b) \leq 1, & i = 1, 2, \dots, n_b \\ \alpha_o^T ((\mathbf{p}_o^b)_j - \mathbf{p}_s^b) \geq 1, & j = 1, 2, \dots, n_o \end{cases}, \quad (2)$$

which means a half space $H = \{x | \alpha_o^T x \leq 1\}$ is required so that the scaled body $P_{body}^s(\beta)$ is inside the half space and the obstacle P_{obs}^s is outside, as shown in Fig.3(b).

We define $\alpha^T = \beta \alpha_o^T$. Since $\beta \in \mathbb{R}_{\geq 0}$, we can formulate the scale calculation as a low dimension LP problem:

$$\begin{aligned} \max \beta \\ \text{s.t.} \quad \begin{cases} \alpha^T ((\mathbf{p}_b^b)_i - \mathbf{p}_s^b) \leq 1, & i = 1, 2, \dots, n_b \\ \alpha^T ((\mathbf{p}_o^b)_j - \mathbf{p}_s^b) \geq \beta, & j = 1, 2, \dots, n_o \end{cases} \end{aligned} \quad (3)$$

B. H-representation

In H-representation, as illustrated in Fig.4(a), we use the intersection of several redundant half spaces to represent a

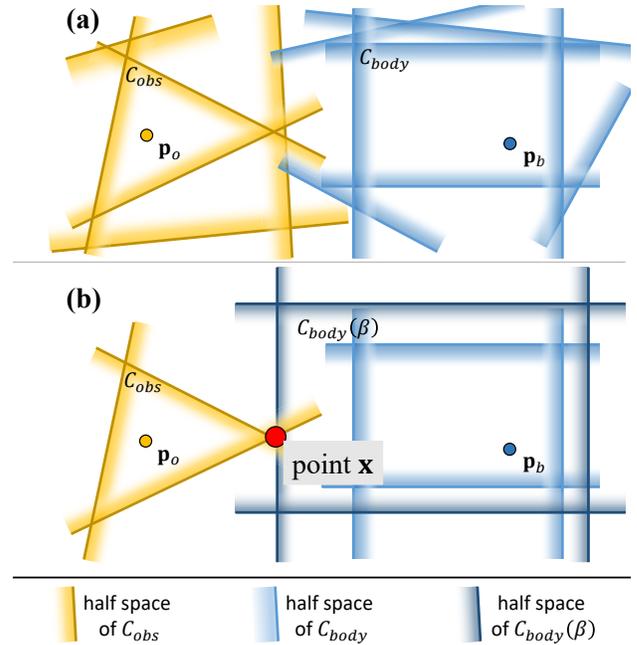


Fig. 4. This figure illustrates the problem definition in H-representation. (a): The obstacle and the body are represented by intersections of several redundant yellow and blue half spaces respectively. (b): For better visualization, we do not show the half spaces that are redundant for the representation. The dark blue points represent the scaled body and the red point denotes the point that belongs to both the scaled body and the obstacle.

convex polyhedron. For a clear visualization, the redundant half spaces are not shown in Fig.4(b).

As indicated in Fig.4(a), we use $C_{body} = \{x | (\alpha_b)_i^T (x - \mathbf{p}_b) \leq 1, i = 1, 2, \dots, n_b\}$ and $C_{obs} = \{x | (\alpha_o)_j^T (x - \mathbf{p}_o) \leq 1, j = 1, 2, \dots, n_o\}$ to represent body and obstacle in the world frame. \mathbf{p}_b and \mathbf{p}_o are points inside body and obstacle respectively. n_b and n_o are the numbers of half spaces whose intersections define body and obstacle. We make the body scale about \mathbf{p}_b , then the scaled body can be written as

$$C_{body}(\beta) = \{x | (\alpha_b)_i^T (x - \mathbf{p}_b) \leq \beta, i = 1, 2, \dots, n_b\}, \quad (4)$$

where $\beta \in \mathbb{R}_{\geq 0}$ is the scale.

Referring to Eq.2 and Eq.3, we define the problem in H-representation as

$$\begin{aligned} \min \beta \\ \text{s.t.} \quad \begin{cases} (\alpha_b)_i^T (x - \mathbf{p}_b) \leq \beta, & i = 1, 2, \dots, n_b \\ (\alpha_o)_j^T (x - \mathbf{p}_o) \leq 1, & j = 1, 2, \dots, n_o \end{cases} \end{aligned} \quad (5)$$

which is obviously a low dimension LP problem. And the constraints of Eq.5 means that a point x which satisfies $x \in C_{body}(\beta) \cap C_{obs}$ is required as illustrated in Fig.4(b).

IV. GRADIENT COMPUTATION

Since both problems we defined in V-representation and H-representation are in the form of LP, we only derive the sub-gradient in V-representation in this paper and the idea of gradient computation in H-representation is similar.

We use the active constraints of the low dimension LP problem to get the gradient of the scale β . In n -dimensional environment, based on the LP problem definition in Eq.3,

there are two different kinds of active constraints which can be written in the form of linear equations:

$$\begin{aligned} \left[\left((\mathbf{p}_{b.ac}^b)_i - \mathbf{p}_s^b \right)^T \quad 0 \right] \begin{bmatrix} \boldsymbol{\alpha}_{n \times 1} \\ \beta \end{bmatrix} &= 1, \quad i = 1, 2, \dots, n_{b.ac} \\ \left[\left((\mathbf{p}_{o.ac}^b)_j - \mathbf{p}_s^b \right)^T \quad -1 \right] \begin{bmatrix} \boldsymbol{\alpha}_{n \times 1} \\ \beta \end{bmatrix} &= 0, \quad j = 1, 2, \dots, n_{o.ac} \end{aligned} \quad (6)$$

where $(\mathbf{p}_{b.ac}^b)_i \in P_{body}^b$, $(\mathbf{p}_{o.ac}^b)_j \in P_{obs}^b$ and $n_{b.ac} + n_{o.ac} = n + 1$, which means we should have $n + 1$ active constraints to solve the $(n + 1)$ -dimensional LP problem.

We refer to $(\mathbf{p}_{b.ac}^b)_i$ and $(\mathbf{p}_{o.ac}^b)_j$ as the active constraint point on the body and the obstacle respectively. In Fig.5, We show the result of the minimum scale calculation defined in V-representation. As indicated in Fig.5(b), the red points in the green and white convex hull are the $(\mathbf{p}_{b.ac}^b)_i$ and $(\mathbf{p}_{o.ac}^b)_j$ respectively. Additionally, the points in the scaled body obtained from the $(\mathbf{p}_{b.ac}^b)_i$ according to Eq.1 and the points $(\mathbf{p}_{o.ac}^b)_j$ are on the hyperplane $P = \{x | \boldsymbol{\alpha}^T x = \beta\}$ which is represented in Fig.5(b) by the blue plane.

We combine all the linear equations in Eq.6 and write them in matrix form as

$$\begin{bmatrix} \left((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b \right)^T & 0 \\ \vdots & \vdots \\ \left((\mathbf{p}_{b.ac}^b)_{n_{b.ac}} - \mathbf{p}_s^b \right)^T & 0 \\ \left((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b \right)^T & -1 \\ \vdots & \vdots \\ \left((\mathbf{p}_{o.ac}^b)_{n_{o.ac}} - \mathbf{p}_s^b \right)^T & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{n \times 1} \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (7)$$

Then we block this matrix equation in Eq.7 into

$$\begin{bmatrix} \mathbf{A}_{n \times n} & \mathbf{B}_{n \times 1} \\ \mathbf{C}_{1 \times n} & \mathbf{D}_{1 \times 1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{n \times 1} \\ \beta \end{bmatrix} = \begin{bmatrix} \mathbf{E}_{n \times 1} \\ \mathbf{F}_{1 \times 1} \end{bmatrix}, \quad (8)$$

which can be written into a system of linear equations with two variables, where the variable $\boldsymbol{\alpha}_{n \times 1}$ can be eliminated and get an equation which only has one variable β :

$$\mathbf{C}_{1 \times n} \mathbf{A}_{n \times n}^{-1} (\mathbf{E}_{n \times 1} - \mathbf{B}_{n \times 1} \beta) + \mathbf{D}_{1 \times 1} \beta = \mathbf{F}_{1 \times 1}. \quad (9)$$

For a rigid body, the point set P_{body}^b in body frame is not related to the motion of the body. As for P_{obs}^b , in practice we can only directly obtain the obstacle's point set P_{obs}^w in world frame. In order not to lose generality, we define the center of rotation of the rigid body in the body frame as \mathbf{p}_{cen}^b , and define \mathbf{R} and \mathbf{t} for the rotation and translation of the body in the world frame. Based on the body's motion and P_{obs}^w , we can get the point in P_{obs}^b as

$$(\mathbf{p}_o^b)_j = \mathbf{R}^{-1} \left((\mathbf{p}_o^w)_j - \mathbf{t} - \mathbf{p}_{cen}^b \right), \quad (10)$$

Then we can conveniently use the implicit function in Eq.9, which contains the relationship between the scale and the motion of the rigid body, to obtain the partial derivatives of β w.r.t. \mathbf{R} and \mathbf{t} . When \mathbf{R} and \mathbf{t} are time-dependent, such as using a time-parameterized polynomial trajectory to represent motion, the gradient of β w.r.t. time can also be obtained.

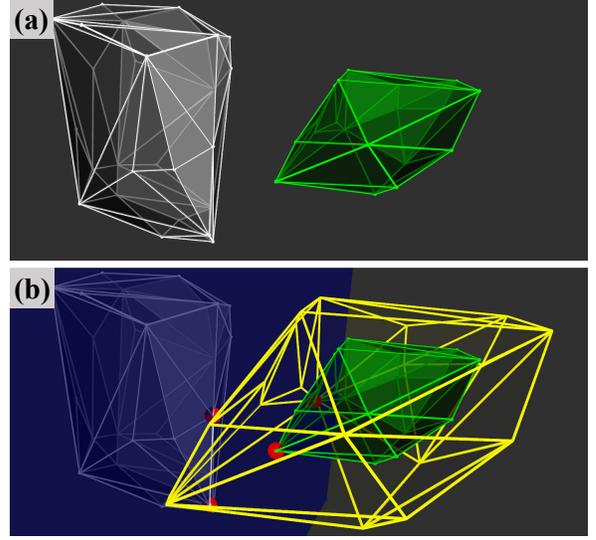


Fig. 5. Illustration of calculation of the minimum scale in V-representation. (a): The white and the green convex hull denote the obstacle and the body respectively. (b): The yellow convex hull is the scaled body obtained by enlarging the body to the minimum scale. The blue plane is the hyperplane splitting the scaled body and the obstacle, and the red points represent the active constraint points that satisfy one of the linear equations in Eq.6. Although the objects are visualized using convex hull, the method can be applied directly to the redundant point sets during the calculation.

V. APPLICATION ON AERIAL ROBOTS

To demonstrate the versatility of our method on aerial robots, we apply it to a whole-body multicopter trajectory optimization problem, based on the differential flatness of multicopter in Wang's work [7]. As shown in Fig.1, we model the multicopter with a green rectangle that measures $3 \times 2 \times 0.6$ m, and require it to fly through a narrow slit. We adopt a segmented polynomial to represent the flat-output trajectory, of which we use the MINCO [7] to conduct spatial-temporal deformation. In this SE(3) trajectory optimization, we set the maximum velocity $6m/s$ and acceleration $10m/s^2$, considering smoothness, safety and dynamic feasibility simultaneously. Moreover, we achieve obstacle avoidance by constraining the minimum scale defined in Sec. III greater than 1, which needs the gradient of the scale w.r.t. the motion of the multicopter.

A. Gradient Computation for SE(3) Motion

We derive the calculation of the gradient in detail. As defined in Sec. IV, we use \mathbf{R} and \mathbf{t} to represent the rotation matrix and translation of the body. For convenience, in optimization we use a normalized quaternion $\mathbf{q} = [w, x, y, z]^T$ to represent rotation. Referring to [22], the rotation matrix \mathbf{R} can be expressed by the quaternion \mathbf{q} and the partial derivatives $\frac{\partial \mathbf{R}}{\partial \mathbf{q}_*}$, $*$ = $\{w, x, y, z\}$ can be easily obtained.

In this case where $n = 3$, as mentioned in Sec. IV, the LP problem which defined in Sec. III-A should have 4 active constraints. The specific situations of active constraints (\mathbf{ac}) can be divided into three types:

- 3 $\mathbf{ac} \in P_{body}^b$, 1 $\mathbf{ac} \in P_{obs}^b$;
- 2 $\mathbf{ac} \in P_{body}^b$, 2 $\mathbf{ac} \in P_{obs}^b$;
- 1 $\mathbf{ac} \in P_{body}^b$, 3 $\mathbf{ac} \in P_{obs}^b$.

We then analyse each type in turn:

1) $3 \mathbf{ac} \in P_{body}^b$, $1 \mathbf{ac} \in P_{obs}^b$: In this case, the block matrix equation in Eq.8 can be written as

$$\begin{bmatrix} \begin{bmatrix} ((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_2 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_3 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \alpha_{3 \times 1} \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}. \quad (11)$$

Then based on Eq.9, we can get

$$\beta = ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \begin{bmatrix} \begin{bmatrix} ((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_2 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_3 - \mathbf{p}_s^b)^T \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{bmatrix}, \quad (12)$$

where only $(\mathbf{p}_{o.ac}^b)_1$ is related to \mathbf{R} and \mathbf{t} . Based on Eq.10, for brevity, we write this equation in Eq.12 as

$$\beta = ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \mathbf{A}^{-1} \mathbf{E}, \quad (13)$$

where \mathbf{A} and \mathbf{E} , defined in Eq.8, are constant in this case. Then we can get the gradient of β w.r.t. \mathbf{t} and \mathbf{q} as

$$\frac{\partial \beta}{\partial \mathbf{t}} = -\mathbf{R} \mathbf{A}^{-1} \mathbf{E}, \quad \frac{\partial \beta}{\partial \mathbf{q}_{*}} = -(\mathbf{p}_{o.ac}^b)_1^T \frac{\partial \mathbf{R}^T}{\partial \mathbf{q}_{*}} \mathbf{R} \mathbf{A}^{-1} \mathbf{E}. \quad (14)$$

2) $2 \mathbf{ac} \in P_{body}^b$, $2 \mathbf{ac} \in P_{obs}^b$: In this case, the block matrix equation in Eq.8 can be written as

$$\begin{bmatrix} \begin{bmatrix} ((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_2 - \mathbf{p}_s^b)^T \\ (\Delta \mathbf{p}_{o.ac}^b)^T \\ ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \alpha_{3 \times 1} \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad (15)$$

where the \mathbf{ac} corresponding to $\Delta \mathbf{p}_{o.ac}^b$ is

$$\begin{aligned} ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \alpha - \beta &= 0, \\ ((\mathbf{p}_{o.ac}^b)_2 - \mathbf{p}_s^b)^T \alpha - \beta &= 0, \end{aligned} \quad (16)$$

which can be combined to obtain

$$((\mathbf{p}_{o.ac}^b)_1^T - (\mathbf{p}_{o.ac}^b)_2^T) \alpha = (\Delta \mathbf{p}_{o.ac}^b)^T \alpha = 0. \quad (17)$$

Based on Eq.10, $\Delta \mathbf{p}_{o.ac}^b$ can be written as

$$\Delta \mathbf{p}_{o.ac}^b = \mathbf{R}^{-1} ((\mathbf{p}_o^w)_1 - (\mathbf{p}_o^w)_2). \quad (18)$$

Then based on Eq.9, we can get

$$\beta = ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \begin{bmatrix} \begin{bmatrix} ((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_2 - \mathbf{p}_s^b)^T \\ (\Delta \mathbf{p}_{o.ac}^b)^T \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix}, \quad (19)$$

which, for brevity, we write as

$$\beta = \mathbf{C} \mathbf{A}^{-1} \mathbf{E}, \quad (20)$$

where \mathbf{C} , \mathbf{A} and \mathbf{E} are defined in Eq.8. In this case, \mathbf{C} is related to \mathbf{R} and \mathbf{t} , \mathbf{A} is only related to \mathbf{R} , and \mathbf{E} is constant.

Then we can get the gradient of β w.r.t. \mathbf{t} as

$$\frac{\partial \beta}{\partial \mathbf{t}} = -\mathbf{R} \mathbf{A}^{-1} \mathbf{E}. \quad (21)$$

And we can get the gradient of β w.r.t. \mathbf{q} as

$$\begin{aligned} \frac{\partial \beta}{\partial \mathbf{q}_{*}} &= -(\mathbf{p}_{o.ac}^b)_1^T \frac{\partial \mathbf{R}^T}{\partial \mathbf{q}_{*}} \mathbf{R} \mathbf{A}^{-1} \mathbf{E} \\ &+ \mathbf{C} \mathbf{A}^{-1} \begin{bmatrix} \mathbf{0}_{1 \times 3} \\ \mathbf{0}_{1 \times 3} \\ (\Delta \mathbf{p}_{o.ac}^b)^T \frac{\partial \mathbf{R}^T}{\partial \mathbf{q}_{*}} \mathbf{R} \end{bmatrix} \mathbf{A}^{-1} \mathbf{E}. \end{aligned} \quad (22)$$

3) $1 \mathbf{ac} \in P_{body}^b$, $3 \mathbf{ac} \in P_{obs}^b$: In this case, the block matrix equation in Eq.8 can be written as

$$\begin{bmatrix} \begin{bmatrix} ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{o.ac}^b)_2 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{o.ac}^b)_3 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b)^T \end{bmatrix} & \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \alpha_{3 \times 1} \\ \beta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \quad (23)$$

Then based on Eq.9, we can get

$$-((\mathbf{p}_{b.ac}^b)_1 - \mathbf{p}_s^b)^T \begin{bmatrix} \begin{bmatrix} ((\mathbf{p}_{o.ac}^b)_1 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{o.ac}^b)_2 - \mathbf{p}_s^b)^T \\ ((\mathbf{p}_{o.ac}^b)_3 - \mathbf{p}_s^b)^T \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \end{bmatrix} \beta = 1, \quad (24)$$

which, for brevity, we write as

$$-\mathbf{C} \mathbf{A}^{-1} \mathbf{B} \beta = 1, \quad (25)$$

where \mathbf{C} , \mathbf{A} and \mathbf{B} are defined in Eq.8. \mathbf{C} and \mathbf{B} are constant in this case. Only \mathbf{A} is related to \mathbf{R} and \mathbf{t} .

Then we can get the gradient of β with respect to $\mathbf{t} = \{\mathbf{t}.x, \mathbf{t}.y, \mathbf{t}.z\}$, as

$$\mathbf{C} \mathbf{A}^{-1} \mathbf{B} \frac{\partial \beta}{\partial \mathbf{t}.x} + \mathbf{C} \mathbf{A}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{R} \mathbf{A}^{-1} \mathbf{B} \beta = 0. \quad (26)$$

Based Eq.25, we can obtain the gradient as

$$\frac{\partial \beta}{\partial \mathbf{t}} = \beta \mathbf{R} \mathbf{A}^{-1} \mathbf{B}. \quad (27)$$

And we can get the gradient of β w.r.t. \mathbf{q} as

$$\frac{1}{\beta} \frac{\partial \beta}{\partial \mathbf{q}_{*}} - \mathbf{C} \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \mathbf{q}_{*}} \mathbf{A}^{-1} \mathbf{B} \beta = 0, \quad (28)$$

which can be written as

$$\frac{\partial \beta}{\partial \mathbf{q}_{*}} = \mathbf{C} \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \mathbf{q}_{*}} \mathbf{A}^{-1} \mathbf{B} \beta^2, \quad (29)$$

where $\frac{\partial \mathbf{A}}{\partial \mathbf{q}_{*}}$ can be get by

$$\frac{\partial \mathbf{A}}{\partial \mathbf{q}_{*}} = - \begin{bmatrix} (\mathbf{p}_{o.ac}^b)_1^T \\ (\mathbf{p}_{o.ac}^b)_2^T \\ (\mathbf{p}_{o.ac}^b)_3^T \end{bmatrix} \frac{\partial \mathbf{R}^T}{\partial \mathbf{q}_{*}} \mathbf{R}, \quad (30)$$

B. Experiment Result

L-BFGS¹ [23] is adopted as an efficient quasi-Newton method to solve the numerical optimization problem. We use Lewis-Overton line search [24] to deal with the non-smoothness of the scale, which sometimes occurs during optimization. As the optimization result shows in Fig.1, the SE(3) whole-body trajectory generated by our method is collision-free and smooth.

¹<https://github.com/ZJU-FAST-Lab/LBFGS-Lite>

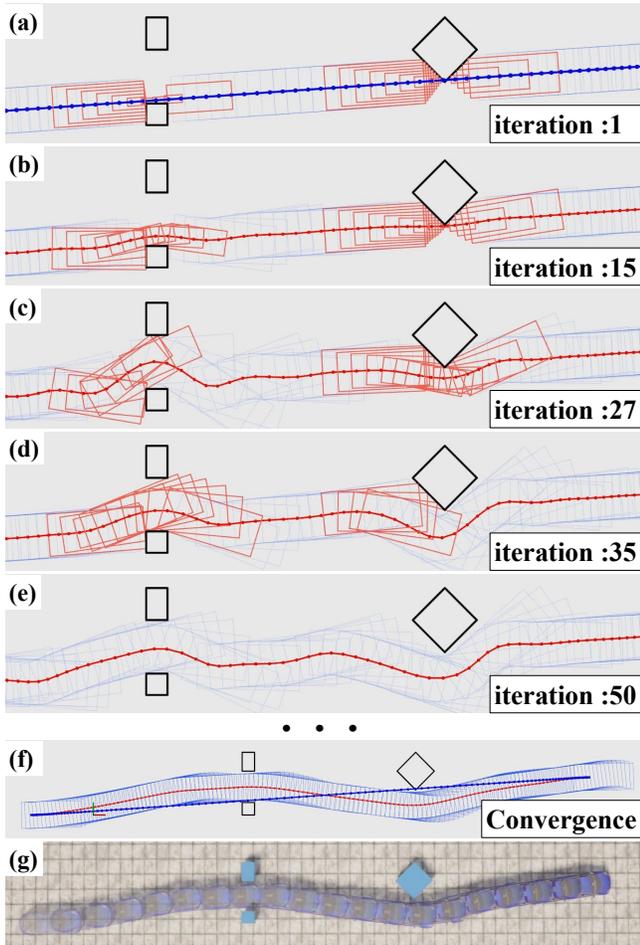


Fig. 6. Illustration of the trajectory optimization. The blue curve indicates the initial trajectory, the red curve is the trajectories in optimization. (a)-(e): Red squares indicate states on trajectories whose scale is small than 1, the light blue squares indicate the states that meet the scale constraint. (f): The converged trajectory is shown, as there are no states on the trajectory that violate the scale constraint after (e). (g): The snapshot of result in CARLA.

VI. APPLICATION ON CAR-LIKE ROBOTS

To demonstrate the applicability of our approach to car-like robots. As shown in Fig.6(g) and Fig.7, we perform the experiments in the physical simulator CARLA [25]. Similar to the application in Sec. V, we adopt a segmented polynomial deformed by MINCO [7] to represent the blue vehicle's trajectory and L-BFGS to solve the numerical problem of the optimization. The trajectory optimization simultaneously considers smoothness, safety and dynamic feasibility. The difference, however, is that the trajectory is 2-d and the nonholonomic constraints are taken into account in this application. Additionally, due to the dynamic environment in Fig.7, we constrain the minimum scale greater than 1 in the spatial-temporal trajectory optimization for safety. (In practice, we set the minimum scale to 1.1 for greater security.) The gradient of the scale w.r.t. the 2-d motion is similar to that in $SE(3)$ and will not be detailed here.

A. Experiment in Static Environment

As shown in Fig.6(g), we apply our method to the blue vehicle's whole-body trajectory optimization problem in a



Fig. 7. The snapshot of the experimental validation of our method applied in a vehicle robot. We plan the trajectory for the blue vehicle to navigate through the dynamic environment.

static environment. To demonstrate that our method is capable of obtaining a whole-body trajectory, using a mass point trajectory that even does not consider the robot's shape as the initial value. The trajectory optimization process for this experiment is illustrated in the Fig.6(a)-6(f). Based on a naive blue initial trajectory, as the number of iterations increases, the states on the trajectory that do not meet the scale constraint disappear. As indicated in the Fig.6(f) and Fig.6(g), the final optimization result is smooth and satisfies the whole-body requirements.

B. Experiment in Dynamic Environment

As illustrated in Fig.7, we employ our method to the blue vehicle's whole-body trajectory optimization problem, which requires the blue vehicle to traverse through the traffic flow consisting of five red vehicles with maximum speeds of 4, 1, 1.5, 4, 5.5 m/s . We set the maximum velocity $8m/s$ and acceleration $2m/s^2$ for the blue vehicle in optimization. For a clear demonstration of the navigating process in the dynamic environment, as shown in Fig.7, we use three temporally consecutive images to show that the result where the blue vehicle's trajectory is safe and smooth.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose an exact whole-body collision formulation via linear scale, which can be solved efficiently. Furthermore, we derive its analytic gradient and applied it to the trajectory optimization in aerial and vehicle robots.

In addition to the applications mentioned in Sec. V and Sec. VI, the proposed method can be applied to other kinds of robots, such as manipulators and legged robots. Moreover, benefiting from its scale-based design, this method can be implemented for deformable robots and swarm formations as well. The above applications of this method will be released in the near future. It is also worth mentioning that we will consider continuous collision formulation in trajectory optimization to improve the completeness of planning.

REFERENCES

- [1] E. G. Gilbert and C. J. Ong, "New distances for the separation and penetration of objects," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 579–586.
- [2] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," *arXiv preprint arXiv:2207.00669*, 2022.
- [3] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.
- [4] S. Cameron, "Enhancing gjk: Computing minimum and penetration distances between convex polyhedra," in *Proceedings of international conference on robotics and automation*, vol. 4. IEEE, 1997, pp. 3112–3117.
- [5] G. Van Den Bergen, *Collision detection in interactive 3D environments*. CRC Press, 2003.
- [6] J. Ji, Z. Wang, Y. Wang, C. Xu, and F. Gao, "Mapless-planner: A robust and fast planning framework for aggressive autonomous flight without map fusion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6315–6321.
- [7] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, 2022.
- [8] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.
- [9] M. Ryll, J. Ware, J. Carter, and N. Roy, "Semantic trajectory planning for long-distant unmanned aerial vehicle navigation in urban environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1551–1558.
- [10] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.
- [11] R. Seidel, "Small-dimensional linear programming and convex hulls made easy," *Discrete & Computational Geometry*, vol. 6, no. 3, pp. 423–434, 1991.
- [12] M. Lutz and T. Meurer, "Efficient formulation of collision avoidance constraints in optimization based trajectory planning and control," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 228–233.
- [13] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.
- [14] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [15] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—a local, continuous method," in *2014 IEEE intelligent vehicles symposium proceedings*. IEEE, 2014, pp. 450–457.
- [16] J. Ji, T. Yang, C. Xu, and F. Gao, "Real-time trajectory planning for aerial perching," *arXiv preprint arXiv:2203.01061*, 2022.
- [17] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se (3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.
- [18] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.
- [19] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for se(3) planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.
- [20] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, 2019.
- [21] S. Manziinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 232–248, 2020.
- [22] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.
- [23] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.
- [24] A. S. Lewis and M. L. Overton, "Nonsmooth optimization via quasi-newton methods," *Mathematical Programming*, vol. 141, no. 1, pp. 135–163, 2013.
- [25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.