

Continuous Implicit SDF Based Any-shape Robot Trajectory Optimization

Tingrui Zhang [†], Jingping Wang [†], Chao Xu, Alan Gao, and Fei Gao

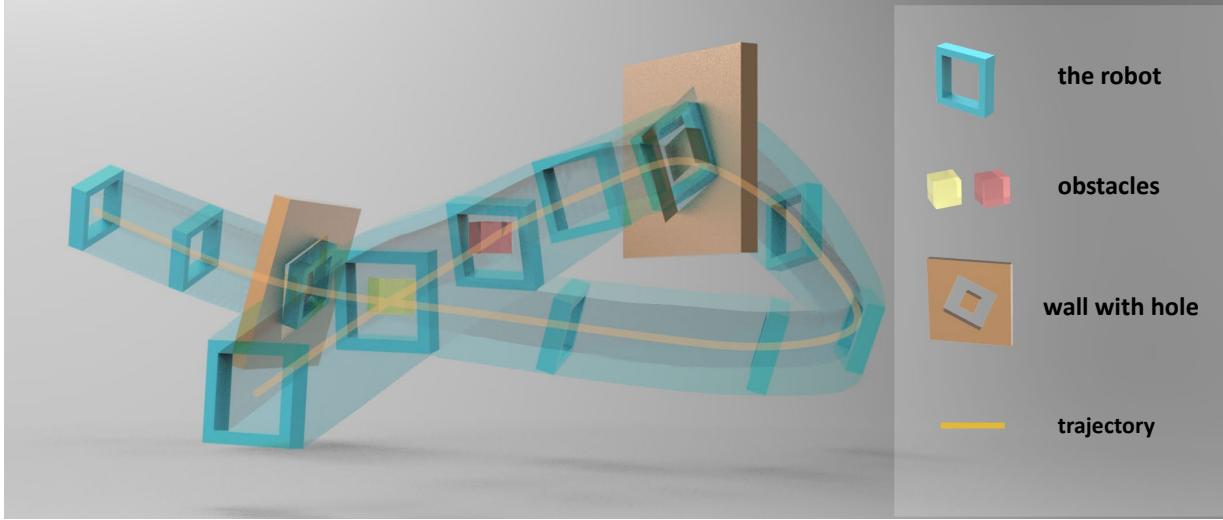


Fig. 1: By integrating our proposed implicit SDF into the trajectory optimization, the swept volume of the box robot can avoid obstacles. Taking advantage of our formulation, our algorithm applies to robots of any shape and achieves continuous collision avoidance.

Abstract—Optimization-based trajectory generation methods are widely used in whole-body planning for robots. However, existing work either oversimplifies the robot’s geometry and environment representation, resulting in a conservative trajectory, or suffers from a huge overhead in maintaining additional information such as the Signed Distance Field (SDF). To bridge the gap, we consider the robot as an implicit function, with its surface boundary represented by the zero-level set of its SDF. Based on this, we further employ another implicit function to lazily compute the signed distance to the swept volume generated by the robot and its trajectory. The computation is efficient by exploiting continuity in space-time, and the implicit function guarantees precise and continuous collision evaluation even for nonconvex robots with complex surfaces. Furthermore, we propose a trajectory optimization pipeline applicable to the implicit SDF. Simulation and real-world experiments validate the high performance of our approach for arbitrarily shaped robot trajectory optimization.

I. INTRODUCTION

Whole-body planning is critical for robots in dense environments. To this problem, optimization-based trajectory generation approaches are effective and have received much attention, with critical considerations including the shape of robots and leveraging environment information from maps.

Two main methods exist for representing a robot’s shape: enclosing it in simple geometric shapes, such as ellipsoids,

cylinders, polyhedrons [1, 2] or using surface samples to represent its geometry [3]. The former lacks precision and the represented shape is often larger than the actual size of the robot, resulting in conservative trajectories. The latter is limited by the resolution, potentially resulting in collision risks at low resolution or complex representation at high resolution. For map representations, optimization-based trajectory planning methods typically require additional information to be recorded in maps to construct safety constraints, such as SDFs [4] and safe corridors [5] of the environment. However, this introduces extra computational and memory overhead. Furthermore, SDFs with a low resolution cannot represent the complex environment precisely, thus adversely affecting the robot’s trajectory planning in dense environments. Safe corridors sacrifice a lot of solution space.

In conclusion, existing methods suffer from the following two problems:

- 1) It is hard to model a robot in a general and efficient way for whole-body planning since the robot’s shape may be complex e.g., non-convex or even changing over time.
- 2) Trajectory optimization usually requires additional information such as SDFs or safe corridors to construct safety constraints, with their corresponding drawbacks.

Based on the above issues, we find that there is still no unified framework that can effectively handle the trajectory generation for arbitrarily shaped robots.

To bridge this gap, we propose a novel approach to whole-body trajectory generation using a continuous implicit SDF

[†] Equal contribution.

All authors are from Zhejiang University, Hangzhou, 310013, China

and the Huzhou Institute of Zhejiang University, Huzhou, 313000, China.

Email:{tingruizhang, 22232111, fgaoaa}@zju.edu.cn

Corresponding Author: Fei Gao

representation. Our approach does not rely on simplification using simple geometric shapes or surface sampling to represent the robot. Instead, we use the original geometry to achieve accurate modeling. The key insight is that the surface boundary of any robot is represented by the zero-level set of its SDF. Moreover, our approach is applicable to different environment representations. We don't need to compute or store SDF for the whole map, is independent of resolution, and also enjoys much more solution space due to our formulation. We combine an implicit SDF representation with the concept of swept volume in computational geometry inspired by [6]–[8]. Based on this, we formulate a continuous implicit function and implement an optimization-based pipeline for any-shape robot trajectory generation.

Swept volume refers to the three-dimensional space occupied by an object as it moves through its entire range of motion. Our approach can evaluate the safety of the swept volume of an arbitrarily shaped robot and ensure that the interior of the swept volume remains free from contact with any obstacles, thus achieving the generation of collision-free trajectories. The SDF is implicitly constructed based on the swept volume and lazily evaluated only at some interested obstacle points with known coordinates. Therefore, it is applicable to different environment representations, such as point cloud maps, feature maps, and grid maps, as long as the coordinates of the obstacles are known. Moreover, no additional information from maps is required. To verify the feasibility and capability of the proposed approach, we perform simulation and real-world experiments on a quadrotor platform. The proposed algorithm is less conservative and offers a wider solution space in optimization, and does not require any complex environment representation. Moreover, benefiting from the continuous implicit SDF, our method can achieve continuous collision avoidance.

We summarize our contributions as follows:

- 1) We consider a robot as an implicit function and propose an algorithm to efficiently obtain the SDF of a swept volume by exploiting the continuity in space-time.
- 2) We propose an optimization-based planning pipeline for any-shape robots, which is based on the continuous implicit SDF and enables continuous collision avoidance.
- 3) We will open source our algorithm¹ for the reference of the community.

II. RELATED WORKS

A. Geometric Shape Representations For Motion Planning

Geometric representations and computations play an important role in robotics [9], especially for whole-body planning. Most research focuses on using some convex geometric shapes such as ellipsoids, polyhedrons, or cylinders to model configuration space or robots for efficient performance, but this sacrifices some solution space.

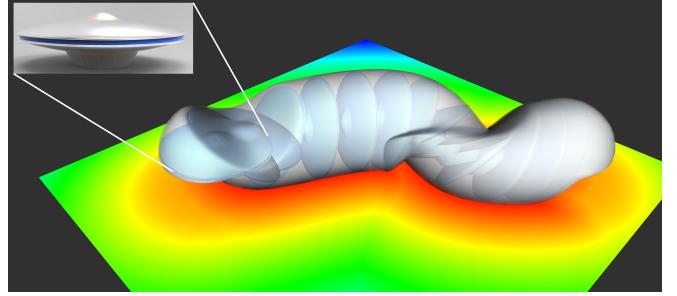


Fig. 2: The swept volume generated by a UFO robot moving along a trajectory with its corresponding SDF is shown here. For ease of visualization, only the horizontal plane at $z=0$ is rendered and the signed distance is displaced using rainbow colors.

To represent configuration space, using sets of polyhedrons to construct safe corridors has been widely adopted by some work [10]–[12]. This representation can be utilized to impose safety constraints for robots in motion planning. However, this simplification may be too conservative for some robots with non-convex shapes due to the introduced gap in collision evaluation. Recently, Tracy et al. [13] use ellipsoids, capsules, boxes, and their combinations as collision primitives to approximate complicated rigid bodies. By formulating a distance minimization problem and using its corresponding derivatives, whole-body planning is achievable. However, this method introduces approximation errors and the gradient computation is cumbersome and relatively expensive. Similarly, Wang et al. [14] model obstacles and robots as polyhedrons and use scale optimization to achieve collision evaluation and whole-body planning. While being exact and having no gap, this method is not suitable for non-convex robots or obstacles.

B. SDF Representations in Robotics

The SDF provides useful information about the distance between a robot and nearby obstacles, which has great potential for motion planning. Most work generates the SDF on pre-constructed maps. For example, the SDF can be constructed on grid maps using the method in [15]. [16] and [17] proposed a kind of fast incremental SDF construction method that can be applied in a dynamic environment. However, in robotics, such approaches have several drawbacks. First, constructing the SDF for an entire map consumes a significant amount of memory resources, making it a challenging task in terms of memory overhead. Second, such approaches require a specific resolution to strike a balance between system overhead and accuracy, which limits their applications in complex and complicated environments. Third, the SDF of a entire map is considered redundant, which is discussed in detail in [18]. Therefore, in robotics, lazy querying is much more attractive to minimize unnecessary computational or memory overhead.

III. CONTINUOUS IMPLICIT SDF GENERATION

A trajectory in $\mathbb{SE}(3)$ consists of a position trajectory $p(t)$ and an attitude trajectory $R(t)$, where $p \in \mathbb{R}^3$ and

¹<https://github.com/ZJU-FAST-Lab/Implicit-SDF-Planner>

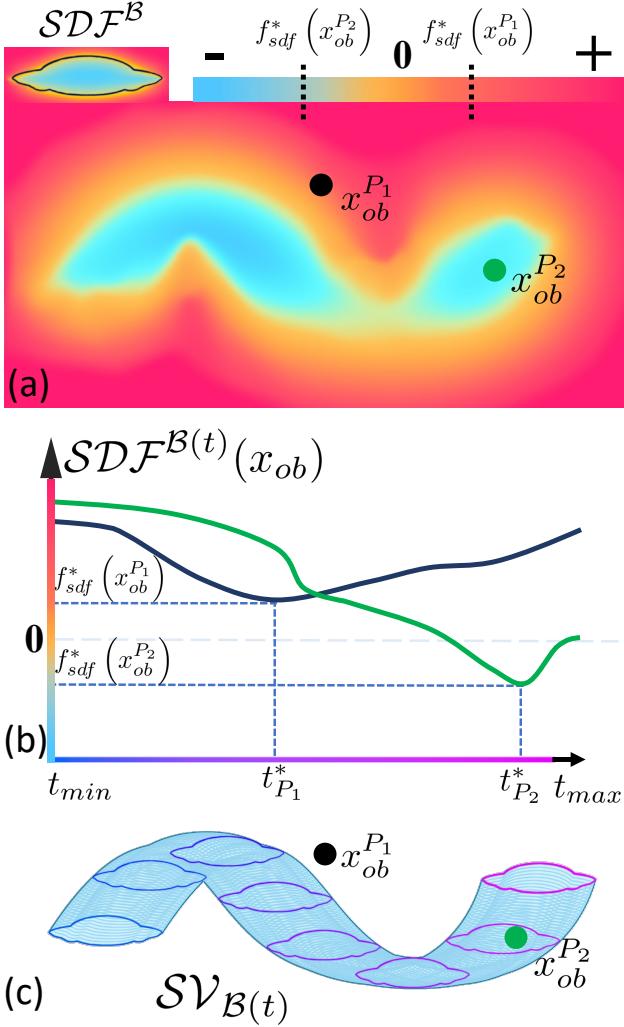


Fig. 3: $\mathcal{SV}_{\mathcal{B}(t)}$ of a UFO robot is shown in Fig.c with its corresponding SDF shown in Fig.a. Fig.b shows the value of $\mathcal{SDF}^{\mathcal{B}(t)}(\mathbf{x}_{ob})$ in the time domain when a query point \mathbf{x}_{ob} is given. Points P_1 and P_2 along with their respective f_{sdf}^* and t^* values are shown here.

$R \in \mathbb{SO}(3)$. Given a robot representation \mathcal{B} , its configuration in $\mathbb{SE}(3)$ can be computed as $\mathcal{B}(t) = R(t)\mathcal{B} + p(t)$. The swept volume generated by the motion of \mathcal{B} during its rigid transformation can be denoted as $\mathcal{SV}_{\mathcal{B}(t)}$.

As mentioned in chapter I, the idea of our algorithm is to guarantee that the $\mathcal{SV}_{\mathcal{B}(t)}$ does not collide with obstacles. Therefore, a metric is needed to evaluate the safety of the $\mathcal{SV}_{\mathcal{B}(t)}$. Signed distance is a commonly used safety metric in robotics and is very easy to be applied in trajectory optimization. In this chapter, we will effectively compute the signed distance of $\mathcal{SV}_{\mathcal{B}(t)}$ by exploiting the continuity in space-time. The lazily computed signed distance will be used for trajectory optimization in chapter IV to achieve safe trajectory generation for any-shape robots.

A. Implicit SDF Representation of Robots

Recall that the surface boundary of any robot is represented by the zero-level set of its SDF. Thus we use it as an implicit continuous function $\mathcal{SDF}^{\mathcal{B}} : \mathbb{R}^3 \rightarrow \mathbb{R}$ to represent an arbitrarily shaped robot that takes a negative value

inside \mathcal{B} ($\mathcal{SDF}^{\mathcal{B}}(\mathbf{x}_{ob}) < 0 : \mathbf{x}_{ob} \in \mathcal{B}$). The implementation of $\mathcal{SDF}^{\mathcal{B}}$ simply relies on off-the-shelf libraries.

In computational geometry, the method of using triangular meshes is the most general and mature way to represent an arbitrary shape [19]. For robot geometry representation, triangular meshes are used here to achieve accurate shape modeling. The implicit SDF is realized by using the winding number signed distance field [20]. Off-the-shelf algorithms such as generalized winding number [21] and semi-general purpose axis-aligned bounding box hierarchy within the LIBIGL² allow the computation of an implicit SDF along with its gradient. With this representation, given a robot \mathcal{B} of any shape, the signed distance $\mathcal{SDF}^{\mathcal{B}}(x)$ and the gradient $\nabla \mathcal{SDF}^{\mathcal{B}}|_x$ at any query point x can be computed efficiently with little overhead.

B. Implicit SDF Representation of Swept Volume

The time-invariant function $\mathcal{SDF}^{\mathcal{B}}$ transforms into a time-varying function as a result of the robot's motion:

$$f_{sdf}(\mathbf{x}_{ob}, t) = \mathcal{SDF}^{\mathcal{B}(t)}(\mathbf{x}_{ob}) = \mathcal{SDF}^{R(t)\mathcal{B} + p(t)}(\mathbf{x}_{ob}). \quad (1)$$

Based on the relativity of motion, equation (1) can be rewritten as:

$$f_{sdf}(\mathbf{x}_{ob}, t) = \mathcal{SDF}^{\mathcal{B}}(R^{-1}(t)(\mathbf{x}_{ob} - p(t))), \quad (2)$$

and its derivative with respect to t is:

$$\dot{f}_{sdf}|_{\mathbf{x}_{ob}} = (\nabla \mathcal{SDF}^{\mathcal{B}}|_{\mathbf{x}_{rel}})^T (R^{-1}\dot{R}R^{-1}(p - \mathbf{x}_{ob}) - R^{-1}\dot{v}). \quad (3)$$

The term \mathbf{x}_{rel} refers to $R^{-1}(\mathbf{x}_{ob} - p)$ and the symbol v represents the velocity, namely \dot{p} . As Figure 3 shows, for any given query point \mathbf{x}_{ob} , $\mathcal{SDF}^{\mathcal{B}(t)}(\mathbf{x}_{ob})$ is a time-variant function due to the motion of \mathcal{B} . Intuitively, if $f_{sdf}(\mathbf{x}_{ob}, t)$ reaches its minimum value in the time domain and the corresponding moment is t^* , then $\mathcal{SDF}^{\mathcal{B}(t^*)}(\mathbf{x}_{ob})$ is the signed distance of \mathbf{x}_{ob} to the $\mathcal{SV}_{\mathcal{B}(t)}$. Assuming that $p(t)$ and $R(t)$ are continuous then $\mathcal{SDF}^{\mathcal{B}(t)}(\mathbf{x}_{ob})$ enjoys continuity in space-time, which makes it easy to obtain the minimum value of f_{sdf} by some numerical methods.

Denote the signed distance of \mathbf{x}_{ob} with respect to $\mathcal{SV}_{\mathcal{B}(t)}$, namely the minimum value of f_{sdf} as follows:

$$f_{sdf}^*(\mathbf{x}_{ob}) \triangleq \min_{t \in [t_{min}, t_{max}]} \mathcal{SDF}^{\mathcal{B}}(R^{-1}(t)(\mathbf{x}_{ob} - p(t))), \quad (4)$$

where the associated argmin time is denoted as follows:

$$t^*(\mathbf{x}_{ob}) \triangleq \operatorname{argmin}_{t \in [t_{min}, t_{max}]} \mathcal{SDF}^{\mathcal{B}}(R^{-1}(t)(\mathbf{x}_{ob} - p(t))). \quad (5)$$

We employ a continuation technique to tackle the problem in equation (4). Rather than computing the minimum value directly, we focus on its associated argmin since we have observed that t^* exhibits piecewise continuity over the spatial domain as shown in Fig.4. The computation of f_{sdf}^* at \mathbf{x}_{ob} with respect to $\mathcal{SV}_{\mathcal{B}(t)}$ yields a new implicit function. This involves identifying the argmin t^* that corresponds to the

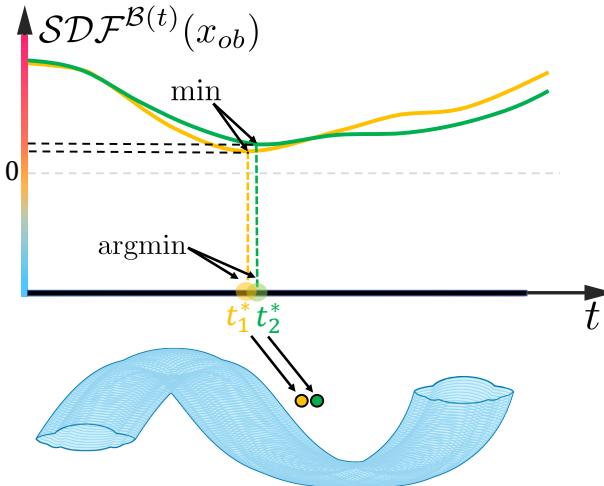


Fig. 4: The function $SDF^B(R^{-1}(t)(x_{ob} - p(t)))$ exhibits continuity in both the space and time domains. Furthermore, the argmin t^* exhibits piecewise continuity. The proximity of the yellow and green points in space is also reflected in the proximity of their respective argmin values, t_1^* and t_2^* in time as shown here.

minima of the implicit value, evaluated relative to the motion of the body \mathcal{B} along the entire trajectory.

C. Space-time Continuation Efficient SDF Computation

We compute t^* in equation (5) by a combination of gradient descent and Armijo line search [22] method. The convergence speed of the algorithm depends on the choice of the initial value t_{init} . We perform trajectory sampling at uniform time intervals, calculate the shortest distance from x_{ob} to the sampled robot \mathcal{B} along the trajectory, and obtain the corresponding moment as t_{init} , which is closed to t^* . In addition, when calculating f_{sdf}^* at several neighboring query points near x_{ob} , we use t^* of x_{ob} as t_{init} . These strategies substantially improve the efficiency of the computation, making each query take only microseconds. The complete computation can be stated as Algorithm 1.

Algorithm 1 Efficient Computation for argmin and SDF

Require: Implicit function SDF^B , $R(t)$, $\dot{R}(t)$, $x(t)$, $v(t)$

- 1: **Initialize:** $t \leftarrow t_{init}$, $x \leftarrow x_{ob}$, $\eta \leftarrow 0.02$, $c \leftarrow 0.5$
- 2: **procedure** PROPAGATE(t) ▷ line 2
- 3: search direction: $d \leftarrow -\dot{f}_{sdf}|_x^{t^k}$ ▷ using 3
- 4: **while** $f_{sdf}(x, t^k + \eta d) > f_{sdf}(x, t^k) + c\eta d\dot{f}_{sdf}|_x^{t^k}$ **do** ▷ using 2
- 5: $\eta \leftarrow \eta/2$
- 6: **end while**
- 7: update iterate $t^{k+1} \leftarrow t^k + \eta d$
- 8: **go to** line 2 unless the condition of convergence or termination is satisfied
- 9: get t^* and corresponding f_{sdf}^*
- 10: **end procedure**

²<https://libigl.github.io/>

IV. OPTIMIZATION-BASED TRAJECTORY GENERATION

We use whole-body planning of quadrotors as a case study here. Quadrotors have the property of differential flatness, which allows the attitude trajectory to be derived from the position trajectory, thus reducing the dimensionality of the trajectory optimization problem [23].

A. Trajectory Representation

In this work, we adopt $\mathfrak{T}_{\text{MINCO}}$ [12] to represent trajectories, which is a minimum control effort polynomial trajectory class defined as:

$$\begin{aligned} \mathfrak{T}_{\text{MINCO}} = & \{p(t) : [0, T_\Sigma] \rightarrow \mathbb{R}^m | \mathbf{c} = \mathcal{M}(\mathbf{q}, \mathbf{T}), \\ & \mathbf{q} \in \mathbb{R}^{(M-1)m}, \mathbf{T} \in \mathbb{R}_{>0}^M, \\ & \mathbf{c} = (\mathbf{c}_1^T, \dots, \mathbf{c}_M^T)^T \in \mathbb{R}^{2Ms \times m}, \\ & \mathbf{q} = (\mathbf{q}_1, \dots, \mathbf{q}_{M-1}) \in \mathbb{R}^{(M-1) \times m}, \\ & \mathbf{T} = (T_1, T_2, \dots, T_M)^T \in \mathbb{R}^M\}. \end{aligned} \quad (6)$$

The trajectory $p(t)$ is an m -dimensional polynomial with M pieces and degree $N = 2s - 1$, where s is the order of the relevant integrator chain. \mathbf{c} is polynomial coefficients and \mathbf{q} is intermediate waypoints. The time allocated for each piece is given by \mathbf{T} , and the total time is $T_\Sigma = \sum_{i=1}^M T_i$. The parameter mapping $\mathcal{M}(\mathbf{q}, \mathbf{T})$ is constructed based on Theorem 2 in [12].

An m -dimensional M -segment trajectory is described by the function as:

$$p(t) = p_i(t - t_{i-1}) \quad \forall t \in [t_{i-1}, t_i], \quad (7)$$

where the i^{th} segment of the trajectory is represented by a polynomial of degree $N = 5$:

$$p_i(t) = \mathbf{c}_i^T \beta(t) \quad \forall t \in [0, T_i]. \quad (8)$$

$\mathbf{c}_i \in \mathbb{R}^{(N+1) \times m}$ is the coefficient matrix, $\beta(t) = [1, t, \dots, t^N]^T$ is the natural basis, and $T_i = t_i - t_{i-1}$ is the time duration of the i^{th} segment. The trajectory representation $\mathfrak{T}_{\text{MINCO}}$ is uniquely determined by the pair (\mathbf{q}, \mathbf{T}) . The mapping $\mathbf{c} = \mathcal{M}(\mathbf{q}, \mathbf{T})$ converts the representation (\mathbf{q}, \mathbf{T}) into (\mathbf{c}, \mathbf{T}) , allowing any second-order continuous cost function $J(\mathbf{c}, \mathbf{T})$ to be expressed as $H(\mathbf{q}, \mathbf{T}) = J(\mathcal{M}(\mathbf{q}, \mathbf{T}), \mathbf{T})$. As a result, the partial derivatives $\partial H / \partial \mathbf{q}$ and $\partial H / \partial \mathbf{T}$ can be obtained from $\partial J / \partial \mathbf{c}$ and $\partial J / \partial \mathbf{T}$ with ease.

B. Optimization Problem Formulation

In this paper, we focus on trajectory generation for robots with quadrotor dynamics. To summarize, trajectory generation can be constructed as the following unconstrained optimization problem:

$$\min_{\mathbf{c}, \mathbf{T}} \lambda_s J_s + \lambda_m J_m + \lambda_d J_d + \rho J_t, \quad (9)$$

where the terms J_s , J_m , J_d , J_t are the safety, smoothness, dynamic feasibility, and total time penalties respectively. λ_s , λ_m , λ_d and ρ are their corresponding weights.

Typically, the safety penalty term J_s for optimization is a safety evaluation integral over the entire trajectory, e.g. $J_s = \int_{t_{min}}^{t_{max}} J_s(\mathbf{c}, \mathbf{T}, t) dt$. Since the integral result has

no analytical form, it is often approximated by discrete summation in practical applications. However, obtaining safety penalties by discrete sampling along the trajectory poses a risk of missing collisions between sampled instants, leading to the occurrence of the tunneling phenomenon [24]. Moreover, in a sparse environment, many sampling points are safe enough, evaluating these points is thus redundant, which reduces efficiency. In contrast, our approach does not require sampling along the trajectory. Due to the properties of the signed distance to the swept volume, we only need to evaluate the corresponding f_{sdf}^* at obstacle points. This approach theoretically avoids the tunneling phenomenon and has higher efficiency.

1) Safety Penalty J_s : The purpose of the trajectory optimization is to ensure that the swept volume completely avoids obstacles, i.e. that the corresponding f_{sdf}^* at all obstacle points are greater than a safety margin s_{thr} . Therefore, we construct a safety penalty using t^* and f_{sdf}^* derived from III-C to deform our trajectory. The penalty function is:

$$J_s = \sum_{i=1}^{N_{obs}} \mathcal{C}(\mathcal{G}_s(\mathbf{x}_{ob}^i)), \quad (10)$$

$$\mathcal{G}_s(\mathbf{x}_{ob}) = \begin{cases} 0, & f_{sdf}^*(\mathbf{x}_{ob}) > s_{thr}, \\ s_{thr} - f_{sdf}^*(\mathbf{x}_{ob}), & f_{sdf}^*(\mathbf{x}_{ob}) \leq s_{thr}, \end{cases} \quad (11)$$

$$f_{sdf}^*(\mathbf{x}_{ob}) = \mathcal{SDF}^B(R^{-1}(t^*)(\mathbf{x}_{ob} - p(t^*))), \quad (12)$$

$$p(t^*) = c_l^T \beta(t^* - T_0 - T_1 - \dots - T_{l-1}) \text{ located at } l_{th} \text{ piece,} \quad (13)$$

where s_{thr} is the safety threshold. \mathbf{x}_{ob} is the obstacle point near the trajectory selected by the Axis-aligned Bounding Box (AABB) algorithm and N_{obs} is the number of points selected. $\mathcal{C}(\cdot) = \max\{\cdot, 0\}^3$ is the cubic penalty.

The gradients are:

$$\frac{\partial J_s}{\partial \mathbf{c}} = 3 \sum_{i=1}^{N_{obs}} \mathcal{Q}(\mathcal{G}_s(\mathbf{x}_{ob}^i)) \cdot \left(\frac{\partial \mathcal{G}_s(\mathbf{x}_{ob}^i)}{\partial \mathbf{c}} \right) \Big|_{t_i^*}, \quad (14)$$

$$\frac{\partial J_s}{\partial \mathbf{T}} = 3 \sum_{i=1}^{N_{obs}} \mathcal{Q}(\mathcal{G}_s(\mathbf{x}_{ob}^i)) \cdot \left(\frac{\partial \mathcal{G}_s(\mathbf{x}_{ob}^i)}{\partial \mathbf{T}} \right) \Big|_{t_i^*}, \quad (15)$$

$$\frac{\partial \mathcal{G}_s(\mathbf{x}_{ob})}{\partial \mathbf{c}, \mathbf{T}} = \begin{cases} 0, & f_{sdf}^*(\mathbf{x}_{ob}) > s_{thr}, \\ -\frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial \mathbf{c}, \mathbf{T}}, & f_{sdf}^*(\mathbf{x}_{ob}) \leq s_{thr}, \end{cases} \quad (16)$$

$$(17)$$

where $\mathcal{Q}(\cdot) = \max\{\cdot, 0\}^2$ is the quadratic penalty.

For optimization purposes, we use a normalized quaternion, represented by $\mathbf{q} = [w \ x \ y \ z]^T$ to denote the rotation. The corresponding rotation matrix R , is given as:

$$R = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}. \quad (18)$$

Recall that $R^{-1}(t) = R(t)^T$. Given this property, the partial derivatives of $R^{-1}(t)$ and $R(t)$ with respect to $\mathbf{q}.*$ can be easily obtained, where $*$ denotes the elements $w \ x \ y \ z$ in quaternions. Differentiating the equation (2) with respect

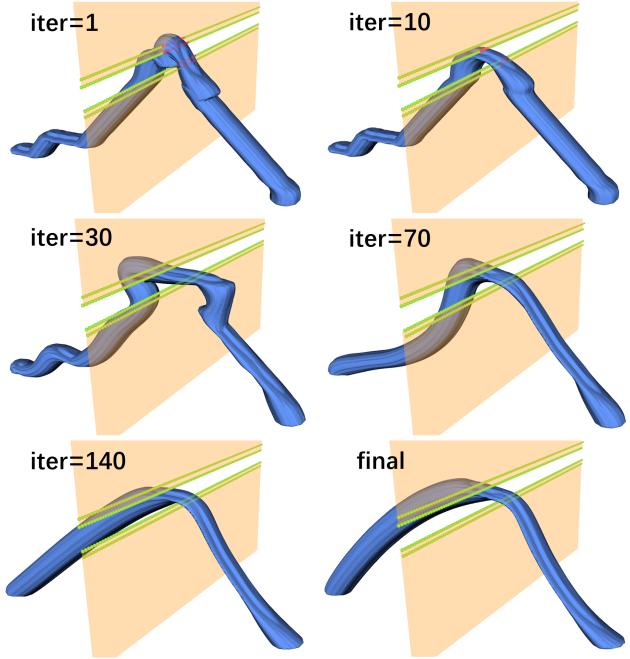


Fig. 5: This figure shows a trajectory snapshot along with its swept volume during the optimization process. A UFO robot manages to fly through narrow gaps, requiring whole-body planning. Red dots represent obstacles that do not satisfy safety constraints.

to $p(t)$ and $\mathbf{q}(t)$ gives the gradients of the signed distance evaluated at \mathbf{x}_{ob} with respect to rotations and translations:

$$\frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial p} = -(\nabla \mathcal{SDF}^B|_{\mathbf{x}_{rel}})^T \cdot R^{-1}(t), \quad (19)$$

$$\frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial \mathbf{q}.*} = (\nabla \mathcal{SDF}^B|_{\mathbf{x}_{rel}})^T \cdot \frac{\partial R^{-1}(t)}{\partial \mathbf{q}.*} \cdot (\mathbf{x}_{ob} - p(t)), \quad (20)$$

where $\nabla \mathcal{SDF}^B|_{\mathbf{x}_{rel}}$ is the gradient of the SDF of the robot body B at the point \mathbf{x}_{rel} .

By choosing $s = 3$ as the integrator chain and using the differential flatness property of quadrotors, it is possible to propagate the gradients with respect to rotation $\mathbf{q}.*$ to position, velocity, acceleration, and jerk. Furthermore, the gradients of the signed distance evaluated at \mathbf{x}_{ob} with respect to $\mathbf{c}_k, \mathbf{T}_k$ can be derived as follows, where ζ denotes position, velocity, acceleration, and jerk, respectively.

$$\frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial \mathbf{c}_k} = \sum_{\zeta=p, v, a_j} \frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial \zeta} \cdot \frac{\partial \zeta(t)}{\partial \mathbf{c}_k}, \quad (21)$$

$$\frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial \mathbf{T}_k} = \sum_{\zeta=p, v, a_j} \frac{\partial f_{sdf}^*(\mathbf{x}_{ob})}{\partial \zeta} \cdot \frac{\partial \zeta(t)}{\partial \mathbf{T}_k}. \quad (22)$$

Note that the trajectory deformation causes t^* to be different for the same \mathbf{x}_{ob} . Therefore, t^* is also a function of the optimization variables \mathbf{c}, \mathbf{T} . However, since t^* is obtained from an optimization problem, it is difficult to explicitly derive the derivatives of t^* with respect to \mathbf{c}, \mathbf{T} . Fortunately, sufficient conditions for optimality for deriving t^* allow us to obtain these derivatives implicitly. In essence, in the



Fig. 6: The real-world indoor experiment: The quadrotor must traverse three consecutive circles while precisely avoiding nearby obstacles. We highlight the most critical frames when the robot is closest to the obstacles.

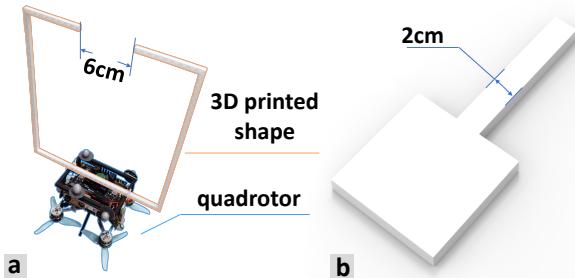


Fig. 7: The UAV platform and the obstacle in the experiment are shown in Fig.a and Fig.b. The velocity and acceleration of the drone in the real experiment are shown in Fig.c.

equation (3), $\dot{f}_{sdf} \equiv 0$ when $t = t^*$. By utilizing this identity equation, the required gradients can be derived. [25] has a description of this method. Implementation details can be found in Appendix VII.

2) *Smoothness Penalty* J_m : To ensure the smoothness of the trajectory, we minimize the integral of the third-order derivative of the trajectory:

$$J_m = \int_0^{T_\Sigma} j^2(t) dt, \quad (23)$$

where $T_\Sigma = \sum_{i=1}^M T_i$ is the total time, $j(t)$ denotes jerk. The gradients are:

$$\frac{\partial J_m}{\partial \mathbf{c}, \mathbf{T}} = 2 \int_0^{T_\Sigma} j(t) \frac{\partial j(t)}{\partial \mathbf{c}, \mathbf{T}} dt. \quad (24)$$

3) *Dynamical Feasibility Penalty* J_d : To satisfy the dynamic constraints of the robot, we limit the maximum velocity and thrust:

$$J_d = \int_0^{T_\Sigma} \mathcal{C}(\mathcal{G}_d(\xi(t))) dt, \quad (25)$$

$$\mathcal{G}_d(\xi(t)) = \begin{cases} 0, & \xi \leq \xi_{max}, \\ \xi - \xi_{max}, & \xi > \xi_{max}, \end{cases} \quad (26)$$

where ξ denotes velocity and thrust respectively. The gradients are:

$$\frac{\partial J_d}{\partial \mathbf{c}, \mathbf{T}} = 3 \int_0^{T_\Sigma} \mathcal{Q}(\mathcal{G}_d(\xi(t))) \frac{\partial \xi(t)}{\partial \mathbf{c}, \mathbf{T}} dt. \quad (27)$$

4) *Total Time Penalty* J_t : We minimize the total time $J_t = \sum_{i=1}^M T_i$ to improve the aggressiveness of the trajectory, the gradients are $\partial J_t / \partial \mathbf{c} = \mathbf{0}$ and $\partial J_t / \partial \mathbf{T} = \mathbf{1}$.

To solve this optimization problem, we use a numerical algorithm. Fig.5 shows the deformation of the trajectory for different iterations and the corresponding swept volume of the UFO robot.

V. RESULTS

A. Implementation details

We validate the approach on a quadrotor platform. Some additional structures are mounted on the quadrotor to simulate a vehicle with complex shapes. All computations are performed on an onboard computer: Nvidia Xavier NX.

We choose the L-BFGS³ algorithm [26] as a highly efficient quasi-Newton approach for solving numerical optimization problems and use the Lewis-Overton line search [27] to address instances of non-smoothness in the scale that may arise during the optimization process.

B. The Real-World Experiment

We conduct a real-world experiment in an indoor environment. Fig.7a shows the quadrotor robot in our experiment. We deliberately install a structure on the robot to give it a complex shape. We hang the obstacle shown in Fig.7b inside a ring so that the robot has to cross the ring from the correct position to ensure that it does not collide. The

³<https://github.com/ZJU-FAST-Lab/LBFGS-Lite/>

maximum safe distance is only 2cm, thus testing the accuracy of our algorithm. In this experiment, the maximum speed and acceleration of the quadrotor are limited to $2m/s$ and $3m/s^2$. Fig.6 and Fig.7c shows the result.

C. Simulation Experiments

To further validate the capability of the proposed algorithm, we construct more complex environments for simulations with a variety of robot shapes. Similarly, a dynamic model of quadrotors is used for optimization.

Two different shapes of robots denoted as \mathcal{B}^x and \mathcal{B}^y are shown in Fig.8. In the first experiment, the environment consists of randomly generated dense obstacles, and the robots traverse this area separately. It is worth noting that our algorithm also works for robots with a hollow shape like \mathcal{B}^y . In the second experiment, the environment consists of three sloping narrow gaps, and \mathcal{B}^x traverses the three gaps in turn while avoiding collisions. Fig.9 and Fig.10 show the swept volume corresponding to the final trajectory. Due to the limitations of the visualization, we recommend watching our video⁴ for a more detailed view of the experimental result.

VI. CONCLUSION AND FUTURE WORK

We present a novel approach to implicitly, lazily, and efficiently compute the signed distance to the swept volume constructed by a robot and its trajectory using the continuity in space-time. Furthermore, we also integrate the continuous implicit SDF into the whole-body optimization problem using quadrotors as a case study.

In principle, our methodology does not impose any restrictions on the class, shapes, or trajectories of robots. Taking advantage of the implicit representation and the analytic form of gradients, this method can also be implemented for any trajectory except polynomials, as long as it is differentiable with respect to time. It is also worth mentioning that we will consider time-variant deformable robots of any shape described by the implicit function. We will also validate this pipeline for different robots with different dynamics for the completeness of planning.

VII. APPENDIX

According to the differential flatness, there is $\dot{R} = R\hat{\omega}$ for quadrotors, we can simplify the equation (3) as follows:

$$\dot{f}_{sdf}|_{\mathbf{x}_{ob}} = (\nabla \mathcal{SDF}^{\mathcal{B}}|_{\mathbf{x}_{rel}})^T (\hat{\omega} R^T (p - \mathbf{x}_{ob}) - R^T v). \quad (28)$$

For simplicity, we use some symbols to represent some formulas above as follows:

$$\mathcal{X}(R, p) \triangleq \left(\nabla \mathcal{SDF}^{\mathcal{B}}|_{\mathbf{x}_{rel}} \right)^T, \quad (29)$$

$$\mathcal{Y}(R, \hat{\omega}, p, v) \triangleq \hat{\omega} R^T (p - \mathbf{x}_{ob}) - R^T v, \quad (30)$$

$$\mathcal{F}(t^*, \zeta) \triangleq \dot{f}_{sdf}|_{\mathbf{x}_{ob}} = \mathcal{X} \cdot \mathcal{Y} \equiv 0, \quad (31)$$

⁴<https://drive.google.com/file/d/1-QQZILtCd5WudsjGIY2Y1KUiTsSiGuco/view>

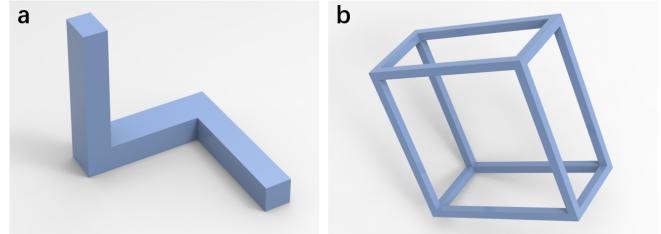


Fig. 8: Two robots with complex shapes in simulation experiments. For convenience, we call the robot in Fig.a \mathcal{B}^x and the robot in Fig.b \mathcal{B}^y .

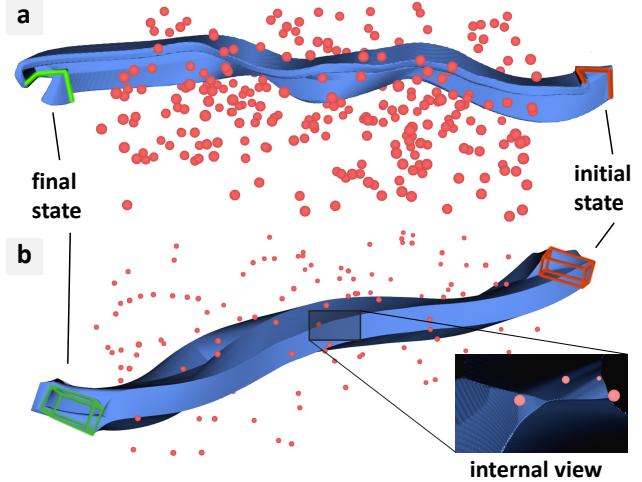


Fig. 9: The results of the first simulation experiment. Pink spheres represent obstacles. Fig.a and Fig.b denote the swept volume of the optimized trajectory of \mathcal{B}^x and \mathcal{B}^y separately. Since \mathcal{B}^y is hollow, there are also some cavities within its swept volume. The close-up of Fig.b shows the obstacles inside these cavities, demonstrating that our algorithm can make full use of the feasible space.

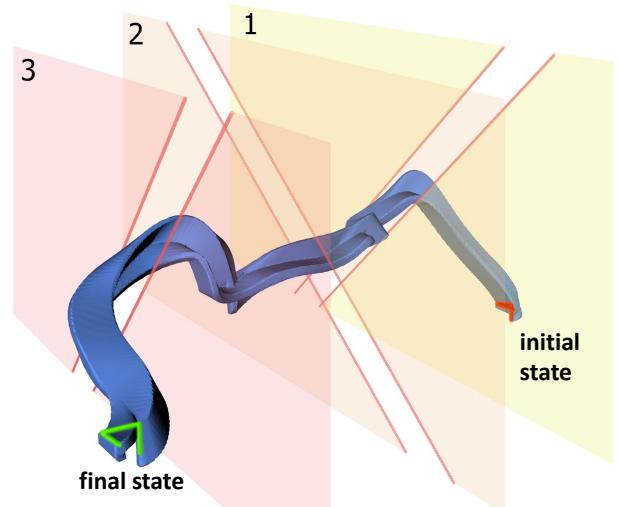


Fig. 10: The result of the second simulation experiment. Since the gaps are narrow, the robot must perform a large attitude maneuver to get through them.

where ζ denotes p, v, ω, R . Deriving the above equations we can obtain:

$$\begin{aligned}\frac{\partial \mathcal{X}}{\partial t^*} &= \left(\nabla^2 \mathcal{SDF}^B|_{\mathbf{x}_{rel}} \left(\frac{\partial R^T}{\partial t} (\mathbf{x}_{ob} - p) - R^T v \right) \right)^T, \\ \frac{\partial \mathcal{Y}}{\partial t^*} &= \frac{\partial \hat{\omega}}{\partial t} (R^T p) + \hat{\omega} \left(\frac{\partial R^T}{\partial t} p + R^T v \right) - R^T a \\ &\quad - \frac{\partial R^T}{\partial t} v - \hat{\omega} \frac{\partial R^T}{\partial t} \mathbf{x}_{ob} - \frac{\partial \hat{\omega}}{\partial t} R^T \mathbf{x}_{ob}.\end{aligned}\quad (32)$$

Recall that:

$$\mathcal{F}(t^*(\zeta), \zeta) \equiv 0, \quad (33)$$

$$\frac{d\mathcal{F}}{d\zeta} = \frac{\partial \mathcal{F}}{\partial t^*} \frac{\partial t^*}{\partial \zeta} + \frac{\partial \mathcal{F}}{\partial \zeta} \equiv 0. \quad (34)$$

$$\frac{\partial t^*}{\partial \zeta} = -\frac{\partial \mathcal{F}}{\partial \zeta} / \frac{\partial \mathcal{F}}{\partial t^*}, \quad (35)$$

$$\frac{\partial \mathcal{F}}{\partial t^*} = \mathcal{X} \frac{\partial \mathcal{Y}}{\partial t^*} + \frac{\partial \mathcal{X}}{\partial t^*} \mathcal{Y}. \quad (36)$$

Finally, we can get the corresponding gradients associated with t^* as follows:

$$\mathcal{K} \triangleq (\mathcal{X} \frac{\partial \mathcal{Y}}{\partial t^*} + \frac{\partial \mathcal{X}}{\partial t^*} \mathcal{Y}),$$

$$\frac{\partial t^*}{\partial v} = (\mathcal{X} R^T) / \mathcal{K}, \quad (37)$$

$$\frac{\partial t^*}{\partial \omega} = -\left(\mathcal{X} \frac{\partial \hat{\omega}}{\partial \omega} R^T (p - \mathbf{x}_{ob})\right) / \mathcal{K}, \quad (38)$$

$$\frac{\partial t^*}{\partial p} = \left(\nabla^2 \mathcal{SDF}^B|_{\mathbf{x}_{rel}} R^T \mathcal{Y} + \mathcal{X} \hat{\omega} R^T\right) / \mathcal{K}, \quad (39)$$

$$\begin{aligned}\frac{\partial t^*}{\partial \mathbf{q}_{.*}} &= \left(\nabla^2 \mathcal{SDF}^B|_{\mathbf{x}_{rel}} \frac{\partial R^T(t)}{\mathbf{q}_{.*}} (\mathbf{x}_{ob} - p) \mathcal{Y}\right) / \mathcal{K} \\ &\quad + \left(\mathcal{X} \left(\hat{\omega} \frac{\partial R^T(t)}{\mathbf{q}_{.*}} (p - \mathbf{x}_{ob}) - v\right)\right) / \mathcal{K}.\end{aligned}\quad (40)$$

Similar to equation (21), the final gradient $\partial t^* / \partial \mathbf{c}, \mathbf{T}$ can be obtained by the differential flatness property.

REFERENCES

- [1] Z. Zhang, Y. Zhang, R. Han, L. Zhang, and J. Pan, "A generalized continuous collision detection framework of polynomial trajectory for mobile robots in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9810–9817, 2022.
- [2] 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.
- [3] M. Przybylski, T. Asfour, and R. Dillmann, "Unions of balls for shape approximation in robot grasping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1592–1599.
- [4] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan, 2016.
- [5] J. Guo, Z. Xun, S. Geng, Y. Lin, C. Xu, and F. Gao, "Dynamic free-space roadmap for safe quadrotor motion planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10523–10528.
- [6] D. Blackmore and M. C. Leu, "Analysis of swept volume via lie groups and differential equations," *The International Journal of Robotics Research*, vol. 11, no. 6, pp. 516–537, 1992.
- [7] S. Sellán, N. Aigerman, and A. Jacobson, "Swept volumes via space-time numerical continuation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–11, 2021.
- [8] P. Xavier, "Fast swept-volume distance for robust collision detection," in *Proceedings of International Conference on Robotics and Automation*, vol. 2, April 1997, pp. 1162–1169 vol.2.
- [9] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of discrete and computational geometry*. CRC press, 2017.
- [10] 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.
- [11] 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.
- [12] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.
- [13] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," *arXiv preprint arXiv:2207.00669*, 2022.
- [14] Q. Wang, Z. Wang, and F. Gao, "A linear and exact algorithm for whole-body collision evaluation via scale optimization," *ArXiv*, vol. abs/2208.06331, 2022.
- [15] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.
- [16] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.
- [17] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4423–4430.
- [18] 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, 2020.
- [19] M. Botsch, M. Pauly, C. Rossli, S. Bischoff, and L. Kobbelt, "Geometric modeling based on triangle meshes," in *ACM SIGGRAPH 2006 Courses*, 2006, pp. 1–es.
- [20] H. Xu and J. Barbić, "Signed distance fields for polygon soup meshes," in *Graphics Interface 2014*. AK Peters/CRC Press, 2020, pp. 35–41.
- [21] G. Barill, N. G. Dickson, R. Schmidt, D. I. Levin, and A. Jacobson, "Fast winding numbers for soups and clouds," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.
- [22] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives," *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966.
- [23] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.
- [24] C. Ericson, *Real-time collision detection*. Crc Press, 2004.
- [25] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *arXiv preprint arXiv:1607.05447*, 2016.
- [26] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1–3, pp. 503–528, 1989.
- [27] A. S. Lewis and M. L. Overton, "Nonsmooth optimization via quasi-newton methods," *Mathematical Programming*, vol. 141, pp. 135–163, 2013.