Any-shape Real-time Replanning via Swept Volume SDF

Yijin Wang* and Tingrui Zhang*, Mengke Zhang, Shuhang Ji, Xiaoying Li, Fei Gao†

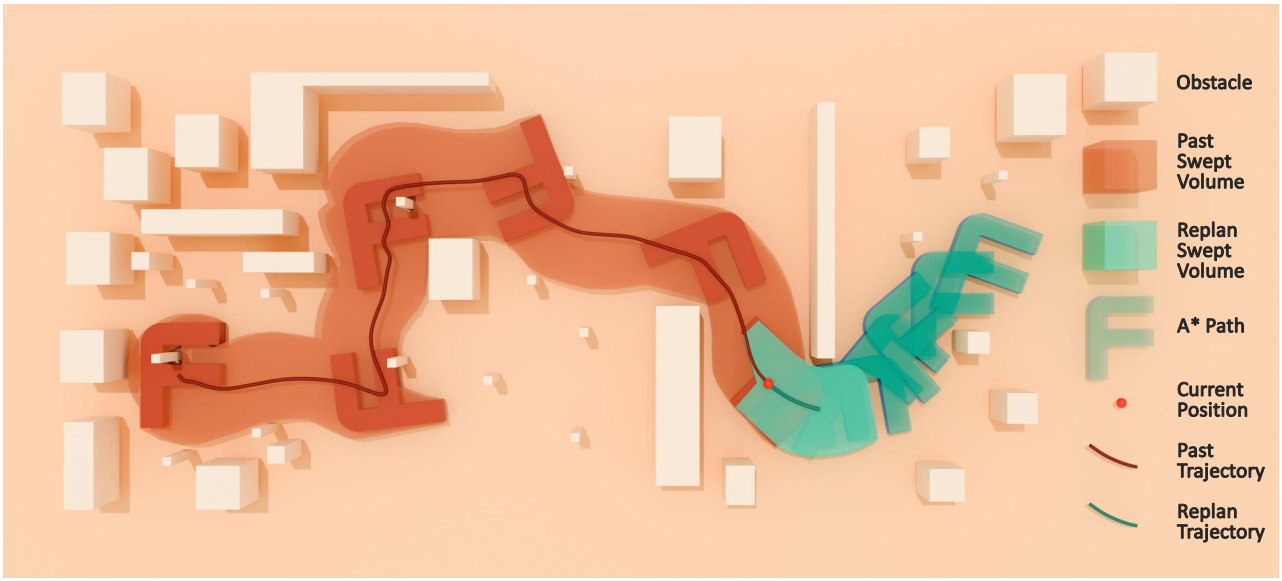

Fig. 1: The F-shaped robot utilizes the Swept Volume Signed Distance Field for replanning and optimizing its trajectory to avoid obstacles in an unknown environment. The figure illustrates the previously planned path and swept volume, alongside the path and swept volume planned at the current moment. The environment contains various small obstacles, leaving very limited feasible space for the robot's movement. As a result, the robot must perform high-precision planning to ensure continuous collision avoidance.

***Abstract*— Existing robotic trajectory planning frameworks typically approximate the robot's geometry and environmental constraints. While this improves computational efficiency, it sacrifices the solution space and frequently encounters failure in confined environments. However, attaining a precise geometric representation and a continuous collision-free trajectory usually necessitates greater computational expenditure. This paper proposes a methodology that utilizes the concept of swept volume to address the identified limitations. The implementation of an efficient Swept Volume Signed Distance Field computation algorithm and a B-spline trajectory representation results in a significant increase in computational efficiency while maintaining strict safety guarantees. The proposed method combines the advantages of efficiency and maximal exploitation of the solution space. Additionally, it ensures continuous obstacle avoidance, achieving real-time 10Hz replanning performance on i50000 NUC11TNK for arbitrarily shaped rigid objects in complex, unstructured environments.**

I. Introduction

Real-time path planning for robots in unknown environments is a critical issue, where agility, safety, and real-time performance are three key metrics [1]–[3].

The majority of existing methods rely on simplifying the robot's geometric shape to achieve real-time obstacle avoidance, which is not in line with the evolving trends in robotics. Robots are increasingly being designed with more diverse and complex geometric shapes. On the one hand, robots themselves are being designed with more intricate appearances to match their functionalities, such as ring-shaped drones [4], mobile robotic arm [5] and dragon-like robots [6]. Conversely, the geometric shape of robots is significantly influenced by the objects they carry, such as quadruped robots carrying loads, dexterous hands grasping objects, or logistics forklifts carrying goods. Although simplified geometric shapes can improve computational efficiency, this comes at the cost of sacrificing solution space, which can lead to planning failures in environments with narrow feasible regions or suboptimal safe paths in non-convex optimization scenarios.

The majority of existing collision detection methodologies rely on discrete sampling along the trajectory. While this technique offers computational efficiency, it often fails to meet the necessary safety guarantees. In unstructured environments, the presence of numerous fragmented obstacles – such as forests or cluttered indoor areas – introduces further

* **Equal contribution.**
All authors are from Zhejiang University, Hangzhou, 310013, China and the Huzhou Institute of Zhejiang University, Huzhou, 313000, China.
Email:wangyijin@bit.edu.cn,tingruizhang@zju.edu.cn, fgaoaa@zju.edu.cn
Corresponding Author: Fei Gao

Manuscript 3389 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2025.

complexity. Even if an obstacle is absent at all sampled robot poses, the trajectory may still carry a risk of collision. Specifically, if the robot does not collide with obstacles at the sampled discrete points, there remains a potential for collision in the unsampled regions. This can result in an unsafe trajectory, as the unaccounted sections may contain hazardous obstacles not captured during the sampling process [7]–[10].

In contrast to traditional methods that simplify geometry, the **S**igned **D**istance **F**ield (SDF) from computer graphics accurately captures the geometric features of both the robot and the environment. However, optimization-based methods that only consider the SDF of the robot itself may struggle with obstacle avoidance during gradient computation [11], [12]. To address this, we introduce the **S**wept **V**olume **S**igned **D**istance **F**ield (SVSDF), which models the robot's motion envelope. This method retains complete geometric information while enabling continuous collision detection along the trajectory, thus reducing the risk of missed collisions due to discrete sampling. Nevertheless, the high computational cost of SVSDF limits its real-time applicability [13], and the accuracy of SVSDF directly influences the safety of the planned path.

To address these conflicting requirements, we have developed an efficient SVSDF optimization framework that combines B-spline trajectory parameterization with a faster SVSDF computation algorithm. This framework ensures high-precision geometric modeling while maintaining real-time planning capabilities, providing a novel solution for complex robotic tasks. This approach opens up new possibilities for applications in areas such as industrial automation, medical robotics, and space exploration, where precision and real-time performance are paramount. Experimental results show that our method achieves real-time replanning with improved agility and safety, significantly outperforming existing SVSDF planners in the literature [11].

We summarize our contributions as follows:

1) We propose a more accurate and efficient SVSDF computation algorithm for the interior of the swept volume, obtaining the step in optimization size without predefined discretization precision of radius.
2) We propose an efficient SVSDF replanning framework, achieving 10Hz replanning on i50000 NUC11TNK in real-world environments.
3) We have integrated a system that balances agility, safety, and real-time performance. We opensource the code at `https://github.com/ZJU-FAST-Lab/Real-Time-SVSDF-Planner.git` to facilitate reproducibility.

II. Related Works

A. Collision Free Guranteen

Most of the existing motion planning methods require a simplification of the robot's shape. For example, Wu et al. approximated the mobile manipulator as a collection of collision balls [14]. Han et al. modeled the geometry of the vehicle as a rectangular prism [15]. Gao et al. simplified the underside of a symmetric quadrotor as a disc [16].

In addition, the feasible region of the robot is often oversimplified. Wang et al. set the feasible region as the union of multiple convex hulls [17]. Jon et al. represent the obstacle-free space as a parametric off-centered ellipse with Chebyshev polynomials [18].

Both simplifying the robot's shape and reducing the feasible region lead to a loss of solution space, making it challenging for the robot to find a viable solution in environments with extremely narrow feasible areas.

Additionally, many methods rely on discrete sampling to detect collisions. For example, Wang et al. sample the trajectory at 0.1-second intervals to check for collisions [19]. However, this approach cannot guarantee that no collisions will occur at any point along the trajectory, i.e., it does not provide absolute continuous safety. This is because at times not sampled along the trajectory, the robot may still collide with obstacles, leading to the problem of collision miss detection [7]. As a result, most previous methods struggle to effectively address the challenge of ensuring continuous collision-free trajectories, especially for complex geometries.

B. Unconstrained Optimization Based Replanning

Unconstrained optimization-based real-time trajectory planning has been widely applied. Such a pipeline can generally be summarized in the following steps. First, the robot's trajectory is parameterized using various methods such as polynomials [17], spline curves [20], and others. This can be a direct parameterization of the pose [17] [20] or an indirect parameterization of the arc length and radius [21]. Next, various indicators such as smoothness, safety, and dynamic feasibility, are converted into multiple penalty costs, where each cost is a function of the trajectory parameters. Finally, by minimizing these costs, a locally optimal solution can be obtained. By providing a good initial guess for the optimization problem, it is possible to converge to a solution that is closer to the global optimum. The optimization process often uses gradient-based optimizers.

This method is robust, computationally efficient and can be applied to devices with limited computational resources. We also employ gradient-based optimization.

C. Application of Swept Volume

In multibody robot path planning, the robot's swept volume is calculated in specific directions to identify potential collisions with environmental obstacles [22], where a tightly fitted ellipsoid is used to approximate the shape, and the Minkowski operation assesses the collision risk. However, it still simplifies the shape of objects to facilitate the computation of the swept volume.

In addition to the explicit modeling of the swept volume, Joho et al. implicitly model the symbolic distance field of the swept volume using neural networks to achieve fast collision detection [23]. They trained a network where the input consists of the initial configuration, the target configuration, and the query point. The network outputs the signed distance

Manuscript 3389 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2025.

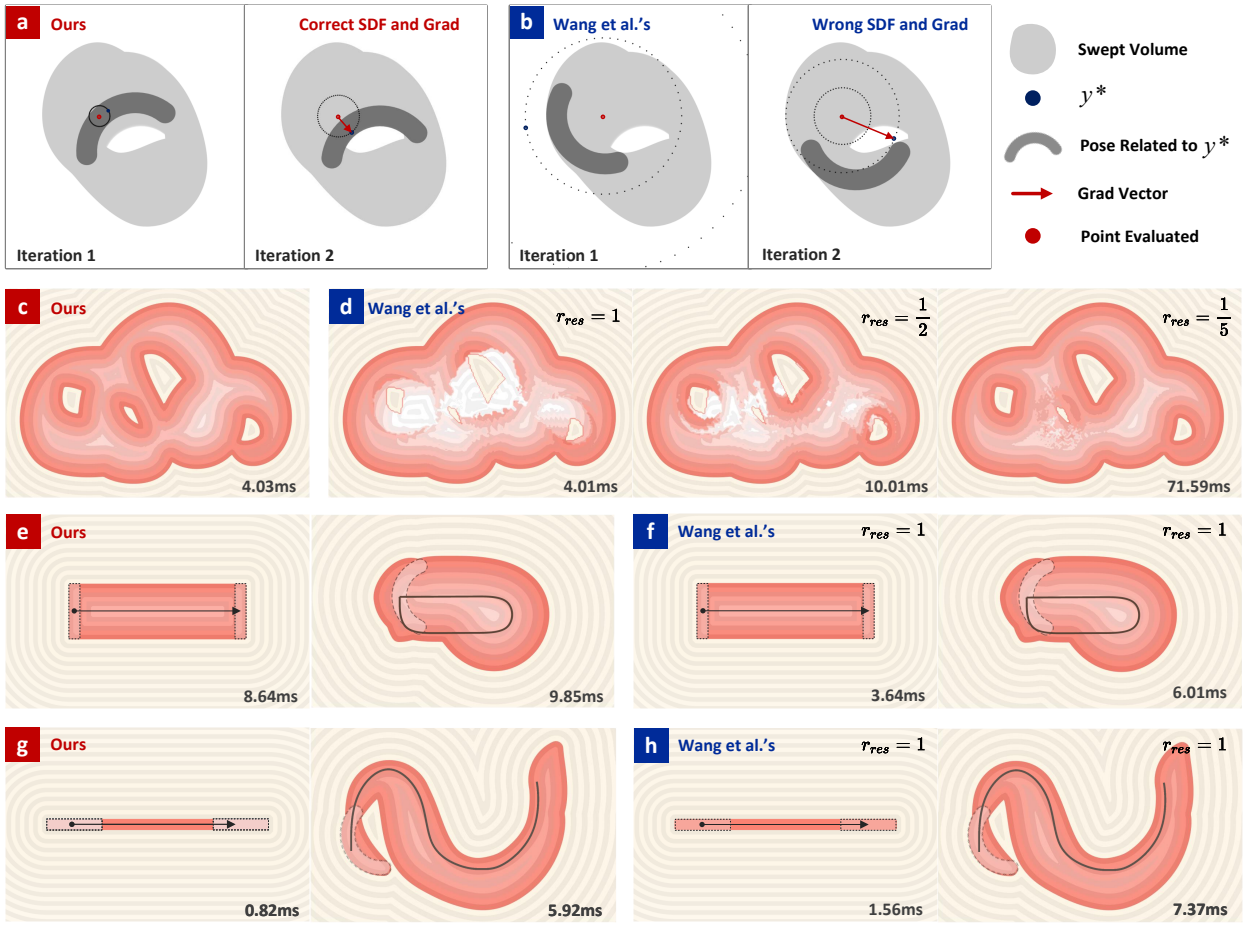

Fig. 2: The time data represents the average computation time for calculating the SVSDF of a point inside the swept volume. (a) Our iterative diagram (b)Wang et al's iterative diagram [11]. (c) The SVSDF result for an arc shape calculated by our algorithm. (d) The SVSDF result for an arc shape under the algorithm in Algorithm 1, with the results from left to right showing no discretization of r, a bisection discretization of r, and a five-way discretization of r. (e)(f)(g)(h)Comparison of SVSDF calculation of simply connected swept volume.

from the query point to the swept volume. However, the accuracy of fitting SVSDF through neural networks is relatively poor, which cannot meet the strict obstacle avoidance requirements for complex-shaped robots. Instead, this paper uses an approach based on numerical optimization to ensure precision.

III. Implicit Swept Volume SDF Calulation

A. Problem Formulation

The swept volume is the volume generated by a robot along a trajectory in 3D space or the area swept in 2D. For a rigid object M moving along a trajectory during t_0 to t_T, its swept volume $\mathcal{SV}$ is defined as

$$\mathcal{SV} = \bigcup_{t\in[t_0,t_T]} R(t)M + p(t), \tag{1}$$

where $R(t)$ denotes the rotation of the object M at time t, and $p(t)$ denotes the position at time t.

Swept **V**olume **S**igned **D**istance **F**ield(SVSDF) precisely computes the minimum signed distance from a point to the swept volume. Referring to Wang et al. [11], we have simplified this problem into an optimization problem. For a two-dimensional example, let $\boldsymbol{p}$ be a point and $\odot_{\boldsymbol{p}}(r)$ denote the circle centered at $\boldsymbol{p}$ with radius r. The boundary of $\mathcal{SV}$ is denoted by $\partial(\mathcal{SV})$, and $\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{p})$ represents the signed distance value of the point $\boldsymbol{p}$ with respect to $\mathcal{SV}$.

$$\begin{aligned}\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{p}) = \text{argmax} \quad & r\cdot\mathbb{1}(\boldsymbol{p}),\\ \text{s.t. } \odot_{\boldsymbol{p}}(r)\cap\partial(\mathcal{SV}) &= \emptyset\end{aligned} \tag{2}$$

where the indicator function $\mathbb{1}(\boldsymbol{p})$ equals 1 if $\boldsymbol{p}\notin\mathcal{SV}$, and -1 otherwise. Obstacle point $\boldsymbol{p}\in\mathcal{SV}$ can be used to push the trajectory and its swept volume away from it by utilizing the $\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{p})$ and its gradient $\nabla\mathcal{SDF}_{\mathcal{SV}}(p)$ as illustrated in Fig.3.

B. Numerical Calulation

For $\boldsymbol{p}\notin\mathcal{SV}$, $\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{p})$ is calculated by solving the minimization problem in equation (3), which means solving for a $t^\star$ that minimizes the signed distance between the robot

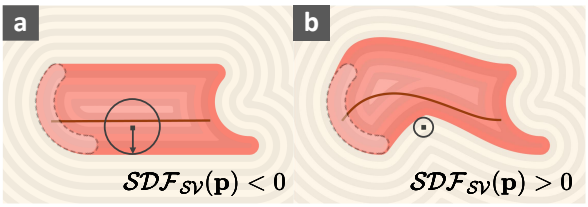

Fig. 3: The left image shows the gradient direction at the obstacle point $\boldsymbol{p} \in \mathcal{SV}$ affected by $\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{p})$, while the right image shows the result of the trajectory achieving a continuous, collision-free swept volume through gradient-based optimization.

and the point $\boldsymbol{p}$:

$$\mathcal{CSDF}_{\mathcal{SV}}(\boldsymbol{p}) = \min_{t\in[t_0,t_T]} \quad \mathcal{SDF}_M(R^{-1}(t)\boldsymbol{p} - p(t)). \quad (3)$$

Since the objective function is non-convex, gradient descent is applied at multiple discrete sampling time stamps to find $t^\star$ as inspired by work in section III.C of [12]. Implementation details are streamlined here, readers may refer to the reference for rigorous derivations. For $\boldsymbol{p} \in \mathcal{SV}$, $\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{p})$ cannot be obtained through above way, because $\mathcal{CSDF}_{\mathcal{SV}}(\boldsymbol{p})$ is conservative, not exact [13], [24]. Wang [11] obtains $\mathcal{SDF}_{\mathcal{SV}}(p)$ for $p \in \mathcal{SV}$ by solving a semi-infinite programming problem using the point set $\mathbf{Y}$:

$$\begin{aligned} \mathbf{Y} &= \boldsymbol{p} + [x_r, y_r]^T, \\ x_r &= r\sin(\theta) r_{res}, \quad . \\ y_r &= r\sin(\theta) r_{res}, \end{aligned} \quad (4)$$

θ represents the polar angle of the circle, while r_{res} denotes the discretization coefficient for the radius. By discretizing the angle and radius separately, a point set $\mathbf{Y}$ for solving the semi-infinite programming problem in Eq.(5) is constructed:

$$\begin{aligned} &\min \quad r, \\ &\text{s.t.}\forall \mathbf{p} \in \odot_{\boldsymbol{p}}(r),\ \mathcal{CSDF}_{\mathcal{SV}}(\mathbf{p}) < \epsilon^{+}_{Precision}. \end{aligned} \quad (5)$$

For a comprehensive explanation and theoretical foundation of this process, please refer to Section 3.2 in [11] for details.

However, this algorithm has two drawbacks: (1) The initial value of r is provided manually, which lacks generality. (2) For non-singularly connected swept volumes, both r and θ require a sufficiently large discretization to ensure accuracy.

It has been observed that the value of r in their algorithm decreases from a sufficiently big initial value until the termination condition is satisfied. This gives rise to the first problem mentioned above. Therefore, it can be surmised that the above problem can be circumvented by simply increasing r continuously from a small value to a large value. For the second problem, a solution can be found by employing the $\mathcal{CSDF}_{\mathcal{SV}}$ to discretize r passively. These are the two most central ideas of our algorithm in Eq.(6):

$$\begin{aligned} &\max \quad r, \\ &\text{s.t.}\forall \mathbf{p} \in \odot_{\boldsymbol{p}}(r),\ \mathcal{CSDF}_{\mathcal{SV}}(\mathbf{p}) < \epsilon^{-}_{Precision}. \end{aligned} \quad (6)$$

The pseudocode of our proposed algorithm is presented in Algorithm 1. For clarity, the main differences between the two algorithms are highlighted, with our novel contributions marked in red and the baseline method from previous work marked in blue. In contrast, our proposed algorithm offers two main advantages by solving the dual problem: (1) The initial value is provided by a conservative SDF, eliminating the need for manual specification. (2) By strategically using the conservative SDF value as the step size for r, only the discretization of θ is required, achieving significantly better accuracy at the same computational cost. We performed the comparison experiments using the open source code[1]. As shown in Fig.2, when the discretization accuracy of r is insufficient, Wang et al.'s method exhibits significant errors in the SDF value and its gradient. After applying a ten-part discretization of r, their method achieved a slightly inferior accuracy to ours but required 17.76 times the computational time. However, this method is not always advantageous. In the simply connected swept volume, due to the use of the conservative SDF value as the step size for increasing r, this algorithm is slightly less efficient for swept volumes of narrow objects moving along their short edge but has an advantage when moving along the long edge. However, considering that the method in [11] requires discretization of r, its actual computational time will be at least twice that shown in Fig.2(h)(f).

Overall, the algorithm we propose is more accurate and efficient.

Algorithm 1 *SVSDF* Computation

1: **function** SampleInCircle($r, \boldsymbol{p}$)
2: Sample a number of points inside the circle $\odot_p(r)$ to form a point set $\mathbf{Y}$ by discretizing θ uniformly referred to equation (4). ▷ discretizing r and θ uniformly
3: **return** $\mathbf{Y}$
4: **end function**
5: ———————————
6: **Input:** qurey point $\boldsymbol{p}$
7: **if** $\mathcal{CSDF}_{\mathcal{SV}}(\boldsymbol{p}) > 0$ **then**
8: **return** $\mathcal{CSDF}_{\mathcal{SV}}(\boldsymbol{p})$
9: **else**
10: $r \leftarrow -\mathcal{CSDF}_{\mathcal{SV}}(\boldsymbol{p})$ ▷ $r \leftarrow$a big initial value
11: $\mathbf{Y} \leftarrow$ SampleInCircle($r, \boldsymbol{p}$)
12: $\Delta r \leftarrow \max\ \mathcal{CSDF}_{\mathcal{SV}}(\boldsymbol{y}),\ \boldsymbol{y} \in \mathbf{Y}$
13: **if** $\Delta r > \epsilon^{-}_{Precision}$ **then** ▷ $\Delta r < \epsilon^{+}_{Precision}$
14: **return** $-r$
15: **end if**
16: $r \leftarrow r - \Delta r$.
17: **goto** line 11.
18: **end if**

IV. Replan System Design

A. Evaulation Points Selection

To initialize the optimization problem, we use the A* algorithm to generate a reasonable sequence of poses. In this process, both the map and the robot's shape are discretized

[1] https://github.com/ZJU-FAST-Lab/Implicit-SVSDF-Planner

into grids, and the robot's yaw angle is also discretized with a resolution of 20 degrees. By performing bitwise operations between the map kernel and the robot shape kernel, we can efficiently check whether a particular robot pose results in a collision, as shown in Fig.4. To speed up the computation, we use an array of 64-bit unsigned integers with 32 rows to represent the robot's occupancy on the grid.

For the selection of evaluation points, we apply a bitwise operation between the dilated robot shape kernel and the map kernel. Compared to the AABB box strategy used in [11], [25] , this approach significantly improves computational efficiency and reduces the number of unnecessary collision checks, as demonstrated by the comparison in Fig.4.

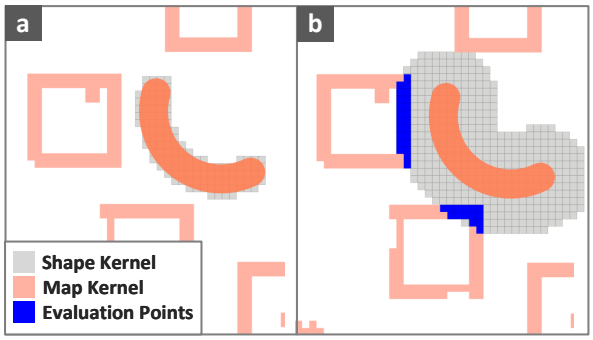

Fig. 4: Shape kernel and map kernel visualizations. (a) Shape kernel for determining the feasibility of a pose in an A* search.(b) Dilated shape kernel for obtaining evaluation points.

Note that the graph searched by the A* algorithm is cyclic in order to ensure the continuity of the yaw angle in the pose sequence. Specifically, if the yaw angle is -180 degrees, it can retrieve not only neighbors with a yaw angle of -160 degrees but also those with a yaw angle of 160 degrees.

B. Trajectory Represention

After obtaining the initial pose sequence from the front-end, we use a uniform B-spline curve to fit these poses as an initial guess for the optimization problem. A B-spline curve is generated by a small number of control points. To ensure the continuity of the acceleration, we use cubic B-spline. For a cubic B-spline defined by $\mathbf{Q} = \{Q_0, \ldots, Q_n\}$ and control points $\mathbf{t} = \{t_0, \ldots, t_{n+3}\}$, for $t \in [t_i, t_{i+1}]$, let the corresponding trajectory be $p_i(\mathbf{Q}, t)$, and define s as the normalization factor, $s = \frac{t-t_i}{t_{i+1}-t_i}$,$\mathbf{M}$ denotes the coefficient matrix:

$$\mathbf{M} = \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}, \tag{7}$$

$$p_i(\mathbf{Q}, s) = \frac{1}{6}\begin{bmatrix} 1 & s & s^2 & s^3 \end{bmatrix} \mathbf{M} \begin{bmatrix} Q_i \\ Q_{i+1} \\ Q_{i+2} \\ Q_{i+3} \end{bmatrix}. \tag{8}$$

B-spline exhibits strong local shape preservation properties [26], [27]. When using gradients to push the swept volume away from obstacles, modifying a subset of the control points does not affect the shape of the curve in regions unrelated to those control points. This behavior contrasts sharply with the performance of piecewise polynomial MINCO trajectories [17] during optimization, as shown in Fig.5. Additionally, B-splines are spatiotemporally coupled curves with relatively few parameters, which offers significant advantages for optimization. A quantitative comparison between the two approaches will be presented in Section V.

C. Optimization

We define three loss functions as penalties: safety, smoothness, and feasibility. The corresponding weighting coefficients are denoted by ω_0 ,ω_1, and ω_2, respectively:

$$\mathbf{Q}^\star = \underset{\mathbf{Q}}{\operatorname{argmin}}\ \omega_0 J_{\text{safety}} + \omega_1 J_{\text{smoothness}} + \omega_2 J_{\text{feasibility}}. \tag{9}$$

The safety metric is evaluated by applying a cubic penalty function L_3 to the SVSDF as derived earlier in this paper:

$$J_{\text{safety}} = \sum_{\boldsymbol{x}_{ob} \in \mathcal{O}} L_3\left[\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})\right], \tag{10}$$

$$L_3(x) = \begin{cases} (\alpha - x)^3 & , x < \alpha \\ 0 & , x \geq \alpha \end{cases}. \tag{11}$$

If the corresponding SVSDF value falls below the safety threshold α, a gradient is applied to the control points of the B-spline to push the trajectory away from obstacles. Thanks to the continuous collision representation provided by the SVSDF and the local support property of the B-spline [28], our improved SVSDF computation algorithm significantly improves both optimization accuracy and efficiency, marking the first integration of this approach into a real-time replanning system.

We apply a smoothness penalty to the integral of the square of the jerk of the trajectory, which helps make the trajectory smooth:

$$J_{\text{smoothness}} = \int_0^{T_\Sigma} \left\| \frac{\partial^3 p(\mathbf{Q}, s)}{\partial s^3} \right\|^2 dt. \tag{12}$$

To ensure the robot adheres to its dynamic constraints, we impose limits on the maximum velocity and acceleration as follows:

$$J_{\text{feasibility}} = \int_0^{T_\Sigma} L_3\left(\mathcal{G}_d(\xi(t))\right) dt, \tag{13}$$

$$\mathcal{G}_d(\xi(t)) = \begin{cases} 0, & \xi \leq \xi_{max}, \\ \xi - \xi_{max}, & \xi > \xi_{max}, \end{cases} \tag{14}$$

where ξ denotes velocity and acceleration, respectively. Detailed derivations of the relevant gradients are provided in the appendix in Section VII for reference.

V. Experiment

A. Benchmark Comparisson

The focus of this section is to compare the optimization efficiency of B-spline and MINCO in the context of obstacle avoidance via SVSDF. We conduct comparative experiments

with the state-of-the-art methods in existing literature [11].To avoid bias in the optimization process due to differences in the smoothness and feasibility of the different curves, both methods use only J_{safety} as the optimization objective function, as previously described in Section IV-C, with its ω set to 60. It is also noteworthy that both methods use the same L-BFGS optimizer [29], a choice that has been shown to be widespread in the existing literature. This consistency extends to the optimizer parameters, which are maintained across both approaches.

The experimental results are shown in Fig.5. It can be observed that when the B-spline representation of the trajectory is used, the objective function converges to 0 much faster, while the swept volume maintains continuous collision-free. Conversely, when the trajectory is represented by MINCO, the optimization process does not converge to 0 and takes approximately 1.41 times longer. If the optimization process is terminated upon reaching a cost of 0, the B-spline-based approach demonstrates a 1.89-fold increase in computational efficiency compared to the MINCO-based method. As demonstrated in Fig.5, the trajectory parameterized using B-spline is significantly superior to the trajectory expressed using MINCO in terms of optimization. This significant improvement can be attributed to the local support property of B-splines, which provides distinct advantages for optimizing continuous collision avoidance problems.

B. Real-World Experiments

1) Experimental Setup: We conducted experiments using a mecanum wheel omni-directional car with a PID controller with feedforward implemented for speed control. The car is equipped with an i50000 NUC11TNK and an Ouster 32-line LiDAR for mapping through FastLIO [30]. We constructed a cluttered environment using thin foam columns and high stools, where more than 70% of the neighbour obstacles have a lateral distance smaller than the height of the F-shaped robot.

To demonstrate the effectiveness of the algorithm, we constructed some load objects with typical structures, such as an F-shape with a deep gap and a star with multiple elongated structures.

The resolution of the map in the experiment is 0.1 m. Given the extremely confined nature of the feasible domain and the ease with which collisions can occur, the safety radius is set to 0.087 m. The maximum velocity of the car is 0.4 m/s, and the maximum angular velocity is 0.3 rad/s.

2) Real-Time Performance: As illustrated in Fig.6(a)(b), the F-shaped robot successfully navigates around a column to avoid obstacles, as indicated by the ring-shaped trajectory of its swept volume. The SVSDF algorithm 1 facilitates rapid and precise gradient computation. In comparison, if the algorithm in [11] is employed, it will fail in real-time replanning. Fig.6(c)(d) illustrates the real-time replanning trajectory of an A-shaped robot. The presence of a person suddenly appearing on its forward path prompts the robot to decelerate, back up, and turn around the person. The entire experiment demonstrates that the planned trajectory

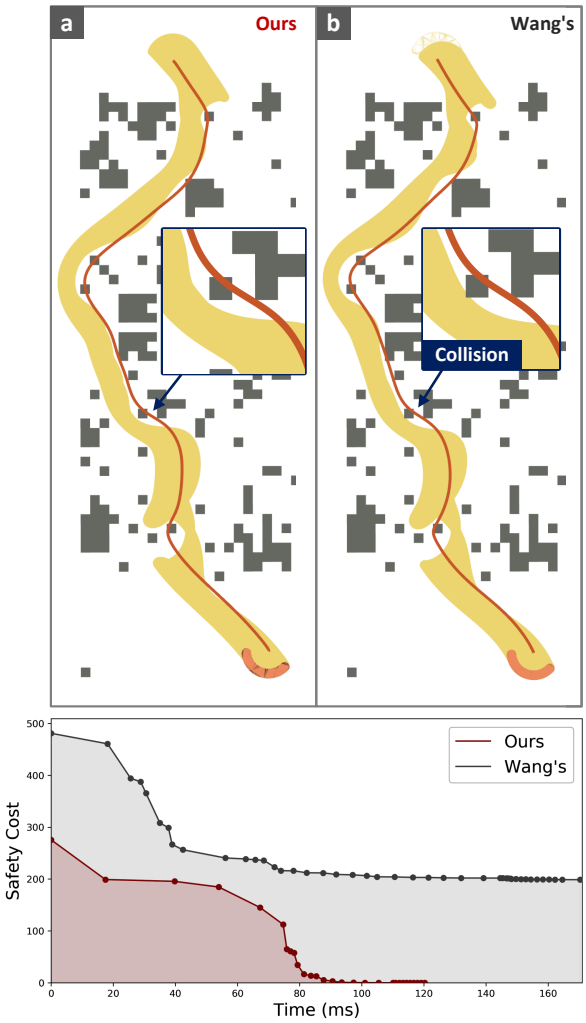

Fig. 5: Impact of using B-spline and MINCO [17] on trajectory optimization, both using smoothness, feasibility and safety cost. The line graphs below show the change in objective function values over time for both when using only exactly the same safety cost. (a) Trajectory parameterized by B-spline. (b) Trajectory parameterized by MINCO.

is smooth and collision-free, while also achieving real-time re-planning.

Experimental results demonstrate that, compared to existing shape-aware planners, our method is the first planning framework capable of real-time replanning for arbitrarily shaped objects. Notably, to the best of our knowledge, this framework represents the first-ever capability for **real-time replanning** of arbitrarily shaped robots in an **unknown environment**, thereby highlighting its dynamic online mapping and obstacle avoidance capabilities. This is a significant departure from previous work [11], [12], [25], which either assumes a known environment map, lacks the efficiency required for real-time replanning, or relies solely on simulation without conducting physical experiments.

VI. Conclusion

This paper presents an enhanced SVSDF computation method that significantly improves numerical accuracy

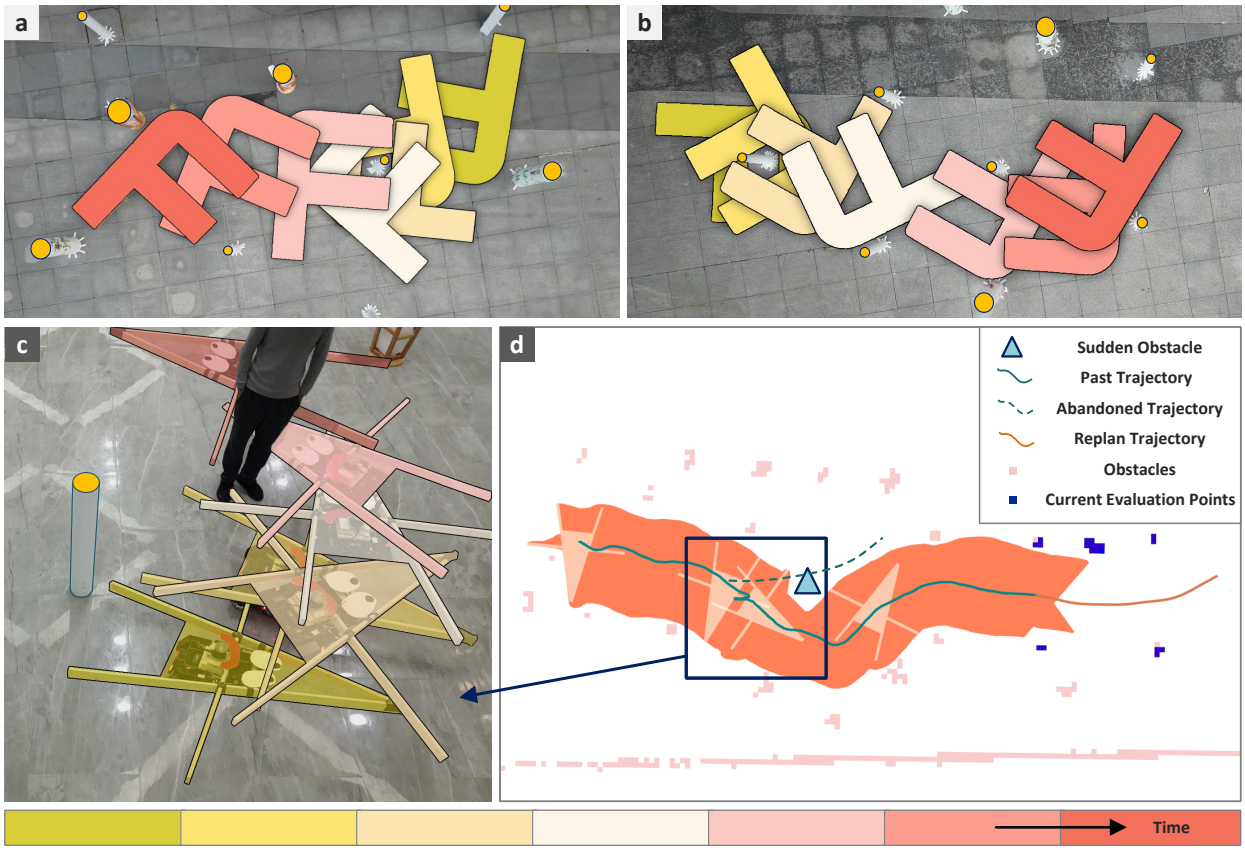

Fig. 6: Real-world experiments. (a)(b)An F-shaped robot navigates a complex environment and avoids clustered obstacles. (c)(d)Real-time replanning of an A-shaped robot. The experimental demonstrations are available in the supplementary video materials accompanying this paper.

within the same time frame, coupled with B-Spline trajectory parameterization to enhance optimization stability. Experimental results demonstrate that these approaches improve the solving efficiency of the optimization problem, enabling the application of SVSDF for real-time replanning tasks on arbitrarily shaped objects. To this end, we have conducted a series of real-world experiments to demonstrate the effectiveness of the proposed methods.

While proposed method effectively enhances collision avoidance performance using SVSDF, it still faces challenges in achieving guaranteed collision avoidance due to the non-convex nature of trajectory optimization problem. We are actively working on improving computational efficiency for real-time trajectory generation in 3D environments and exploring advanced obstacle representation techniques to better handle complex scenarios.

VII. Appendix

The derivative of the cubic penalty function L_3 is easily obtained as:

$$\dot{L}_3(x) = \begin{cases} -3(\alpha - x)^2 & , x < \alpha \\ 0 & , x \geq \alpha \end{cases}. \tag{15}$$

Based on the chain rule of differentiation, we can easily obtain the gradient of the safety term with respect to the control points as follows:

$$\frac{\partial J_{\text{safety}}}{\partial Q_j} = \sum_{\boldsymbol{x}_{ob} \in \mathcal{O}} \dot{L}_3 \left[\mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})\right] \frac{\partial \mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})}{\partial Q_j}, \tag{16}$$

$$\frac{\partial \mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})}{\partial Q_j} = \frac{\nabla \mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})}{\|\nabla \mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})\|} \frac{\partial p_i(\mathbf{Q}, s^*)}{\partial Q_j}, \tag{17}$$

where $\frac{\nabla \mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})}{\|\nabla \mathcal{SDF}_{\mathcal{SV}}(\boldsymbol{x}_{ob})\|}$ represents the unit gradient direction of the SVSDF as derived in section III-B, $\frac{\partial p_i(\mathbf{Q}, s^*)}{\partial Q_j}$ can be expressed analytically from Eq.(8).

The smoothness cost of the entire trajectory is obtained by summing the costs of the individual segments:

$$J_{\text{smoothness}} = \sum_{i \in Pieces} J^i_{\text{smoothness}}, \tag{18}$$

$$\frac{\partial J^i_{\text{smoothness}}}{\partial Q_j} = 2 \left(\int_{t_i}^{t_{i+1}} \frac{\partial^3 p_i(\mathbf{Q}, s)}{\partial s^3} \frac{\partial^4 p_i(\mathbf{Q}, s)}{\partial s^3 \partial Q_j} \mathrm{d}t \right). \tag{19}$$

here j refers to the control points involved in the trajectory segment $[t_i, t_{i+1}]$,meaning $j \in \{i, i+1, i+2, i+3\}$. $\frac{\partial^3 p_i(\mathbf{Q}, s)}{\partial s^3}$ and $\frac{\partial^4 p_i(\mathbf{Q}, s)}{\partial s^3 \partial Q_j}$ can be derived from Equ.(20) and (8).

Manuscript 3389 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2025.

$$\frac{\partial p_i(\mathbf{Q},s)}{\partial s} = \frac{1}{6}\begin{bmatrix}0 & 1 & 2s & 3s^2\end{bmatrix} \quad \mathbf{M}\begin{bmatrix}Q_i\\Q_{i+1}\\Q_{i+2}\\Q_{i+3}\end{bmatrix}$$

$$\frac{\partial^2 p_i(\mathbf{Q},s)}{\partial s^2} = \frac{1}{6}\begin{bmatrix}0 & 0 & 2 & 6s\end{bmatrix} \quad \mathbf{M}\begin{bmatrix}Q_i\\Q_{i+1}\\Q_{i+2}\\Q_{i+3}\end{bmatrix}. \tag{20}$$

$$\frac{\partial^3 p_i(\mathbf{Q},s)}{\partial s^3} = \frac{1}{6}\begin{bmatrix}0 & 0 & 0 & 6\end{bmatrix} \quad \mathbf{M}\begin{bmatrix}Q_i\\Q_{i+1}\\Q_{i+2}\\Q_{i+3}\end{bmatrix}$$

Finally, the gradients for kinematic feasibility utilize the same technical framework as the safety terms. This is because, fundamentally, the velocity and acceleration profiles correspond to the first- and second-order derivatives of the position B-spline trajectory. Specifically, Velocity is represented as a B-spline of degree n−1, derived by differentiating the original degree-n B-spline. Acceleration further reduces the degree to n−2, following the same derivative rules [27].

References

[1] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," *arXiv preprint arXiv:2401.17583*, 2024.

[2] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[3] D. Reyes-Uquillas and T. Hsiao, "Safe and intuitive manual guidance of a robot manipulator using adaptive admittance control towards robot agility," *Robotics and Computer-Integrated Manufacturing*, vol. 70, p. 102127, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584521000119

[4] Y. Wu, F. Yang, Z. Wang, K. Wang, Y. Cao, C. Xu, and F. Gao, "Ring-rotor: A novel retractable ring-shaped quadrotor with aerial grasping and transportation capability," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2126–2133, 2023.

[5] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[6] M. Zhao, T. Anzai, F. Shi, X. Chen, K. Okada, and M. Inaba, "Design, modeling, and control of an aerial robot dragon: A dual-rotor-embedded multilink robot with the ability of multi-degree-of-freedom aerial transformation," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1176–1183, 2018.

[7] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[8] T. Brochu, E. Edwards, and R. Bridson, "Efficient geometrically exact continuous collision detection," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–7, 2012.

[9] S. Redon, A. Kheddar, and S. Coquillart, "Fast continuous collision detection between rigid bodies," in *Computer graphics forum*, vol. 21, no. 3. Wiley Online Library, 2002, pp. 279–287.

[10] Y.-K. Choi, W. Wang, Y. Liu, and M.-S. Kim, "Continuous collision detection for two moving elliptic disks," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 213–224, 2006.

[11] J. Wang, T. Zhang, Q. Zhang, C. Zeng, J. Yu, C. Xu, L. Xu, and F. Gao, "Implicit swept volume sdf: Enabling continuous collision-free trajectory generation for arbitrary shapes," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 4, pp. 1–14, 2024.

[12] T. Zhang, J. Wang, C. Xu, A. Gao, and F. Gao, "Continuous implicit sdf based any-shape robot trajectory optimization," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 282–289.

[13] S. Sellán, N. Aigerman, and A. Jacobson, "Swept volumes via spacetime numerical continuation," *ACM Trans. Graph.*, vol. 40, no. 4, Jul. 2021. [Online]. Available: https://doi.org/10.1145/3450626.3459780

[14] C. Wu, R. Wang, M. Song, F. Gao, J. Mei, and B. Zhou, "Real-time whole-body motion planning for mobile manipulators using environment-adaptive search and spatial-temporal optimization," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 1369–1375.

[15] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen *et al.*, "An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1797–1814, 2023.

[16] Y. Gao, J. Ji, Q. Wang, R. Jin, Y. Lin, Z. Shang, Y. Cao, S. Shen, C. Xu, and F. Gao, "Adaptive tracking and perching for quadrotor in dynamic scenarios," *IEEE Transactions on Robotics*, vol. 40, pp. 499–519, 2024.

[17] 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.

[18] J. Arrizabalaga, Z. Manchester, and M. Ryll, "Differentiable collision-free parametric corridors," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 1839–1846.

[19] D. Wang, H. Ye, N. Pan, J. Huang, B. Zhang, Y. Mao, G. Huang, C. Xu, and F. Gao, "Flexible and topological consistent local replanning for multirotors," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 5348–5355.

[20] 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.

[21] M. Zhang, N. Chen, H. Wang, J. Qiu, Z. Han, Q. Ren, C. Xu, F. Gao, and Y. Cao, "Universal trajectory optimization framework for differential drive robot class," 2024. [Online]. Available: https://arxiv.org/abs/2409.07924

[22] S. Ruan, K. L. Poblete, H. Wu, Q. Ma, and G. S. Chirikjian, "Efficient path planning in narrow passages for robots with ellipsoidal components," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 110–127, 2023.

[23] D. Joho, J. Schwinn, and K. Safronov, "Neural implicit swept volume models for fast collision detection," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 15 402–15 408.

[24] Z. Marschner, S. Sellán, H.-T. D. Liu, and A. Jacobson, "Constructive solid geometry on neural signed distance fields," in *SIGGRAPH Asia 2023 Conference Papers*, ser. SA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3610548.3618170

[25] S. Geng, Q. Wang, L. Xie, C. Xu, Y. Cao, and F. Gao, "Robo-centric esdf: A fast and accurate whole-body collision evaluation tool for any-shape robotic planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 290–297.

[26] W. J. Gordon and R. F. Riesenfeld, "B-spline curves and surfaces," in *Computer aided geometric design*. Elsevier, 1974, pp. 95–126.

[27] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*. Springer Science & Business Media, 2002.

[28] C. De Boor, "On calculating with b-splines," *Journal of Approximation theory*, vol. 6, no. 1, pp. 50–62, 1972.

[29] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[30] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.