Pacific Graphics 2025
M. Christie, N. Pietroni, and Y.-S. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 7

Swept Volume Computation with Enhanced Geometric Detail Preservation

Pengfei Wang*[1], Yuexin Yang*[1], Shuangmin Chen†[2], Shiqing Xin[1], Changhe Tu[1], Wenping Wang[3]

[1]Shandong University [2]Qingdao University of Science and Technology [3] Texas A&M University
* Equal Contribution; † Corresponding Author.

Abstract

Swept volume computation—the determination of regions occupied by moving objects—is essential in graphics, robotics, and manufacturing. Existing approaches either explicitly track surfaces, suffering from robustness issues under complex interactions, or employ implicit representations that trade off geometric fidelity and face optimization difficulties. We propose a novel inversion of motion perspective: rather than tracking object motion, we fix the object and trace spatial points backward in time, reducing complex trajectories to efficiently linearizable point motions. Based on this, we introduce a multi-field tetrahedral framework that maintains multiple distance fileds per element, preserving fine geometric details at trajectory intersections where single-field methods fail. Our method robustly computes swept volumes for diverse motions, including translations and screw motions, and enables practical applications in path planning and collision detection.

CCS Concepts
• ***Computing methodologies*** → ***Shape modeling;***

1. INTRODUCTION

Swept volumes—the spatial regions traversed by moving objects—are fundamental geometric primitives in computer-aided design [PPSZ05, AMS14, AMYB01, SG05], robotics [AMY97, JSS24, MFJQ*16a, ITA*23], and virtual reality [MGS17, KRL*07]. Figure 1 illustrates a swept volume generated by an object following a trajectory. And Figure 2 demonstrates two specific application scenarios: (a) robot vacuum path planning and collision avoidance, and (b) autonomous vehicle maneuvering in complex environments like underground parking facilities. Accurate swept volume computation significantly impacts collision detection [MFJQ*16b, WZZ*24, Xav97], spatial reasoning [AMYBJ06], and geometric modeling [Lar20, WHS00]. Despite their importance, efficiently generating high-quality swept volumes for complex shapes undergoing intricate motions remains challenging.

Traditional methods fall into two main categories: explicit geometry-based approaches that construct boundaries directly from meshes, and implicit field-based methods leveraging scalar fields. Explicit geometry-based approaches directly operate on mesh representations to construct swept volume boundaries, ranging from analytical solutions for restricted motions (such as Minkowski sums for translations) to sophisticated polyhedral sweeping algorithms for general rigid motions [Kau93, Kor86, WL90]. These methods face fundamental challenges in robustly handling complex surface patch interactions during arbitrary motion. Implicit field-based approaches represent swept volumes using scalar fields, typ-

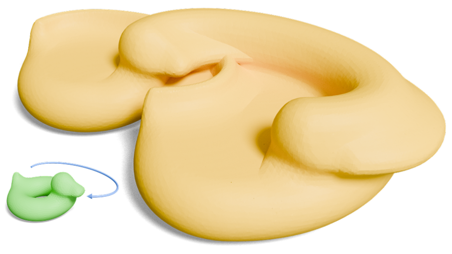
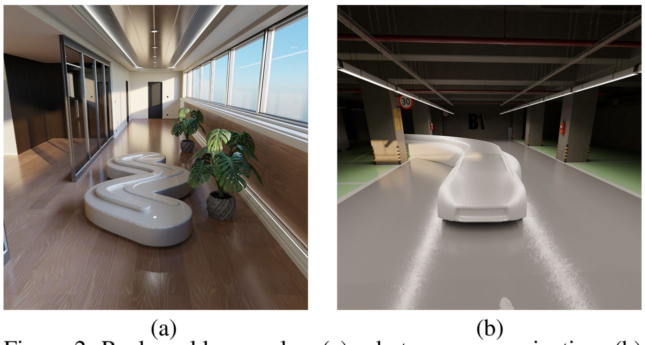


Figure 1: Swept volume visualization. This geometric entity is formed by the union of all spatial positions occupied by an object as it moves along a defined trajectory, creating a continuous volumetric region.


(a) (b)
Figure 2: Real-world examples: (a) robot vacuum navigation; (b) autonomous vehicle parking.

ically signed distance functions (SDFs), to avoid explicit tracking of surface intersections [SP96, SLL94, SAJ21]. Recent implicit approaches have made significant advances by attempting to obtain distance values from arbitrary spatial points to the resulting swept volumes through optimization. However, they are constrained by the complexity of input model SDFs and motion functions. The optimization process often converges slowly, becomes trapped in local minima, and lacks theoretical guarantees, especially for motions containing high-frequency details. Furthermore, traditional isosurface extraction strategies applied to the resulting SDFs struggle to preserve complex geometric details.

In this work, drawing inspiration from inverse analysis approaches in ray tracing methods [Bar86] and existing swept volume computation techniques [SAJ21], and addressing the limitations of single-field extraction methods, we propose a fundamentally different approach based on motion perspective inversion. Our method takes mesh models as both input and output. Rather than tracking objects through space, we fix the object and consider spatial points traversing inverse trajectories through time. This perspective shift transforms the challenging problem of analyzing complex object motion into the simpler task of studying individual point trajectories, which can be linearly approximated within sufficiently small temporal intervals. Leveraging this insight, we introduce a multi-field isosurface extraction method maintaining multiple scalar fields within each tetrahedral element, capturing complex geometric features missed by single-field methods.

Our contributions include:

- **Motion perspective inversion**: Transforming complex motion tracking into tractable point-wise operations.
- **Multiple distance field representation**: Preserving detailed geometric features via multiple scalar fields per tetrahedron.
- **Four-dimensional incremental cutting**: Robustly extracting detailed swept volume surfaces.

These innovations enable accurate, efficient swept volume computations suitable for practical applications.

2. RELATED WORK

Swept volume computation methods have evolved primarily along two trajectories: explicit geometry-based approaches, which directly manipulate mesh representations, and implicit field-based approaches, which utilize scalar fields to represent swept regions. We review key developments in both categories, highlighting their fundamental principles, advantages, and limitations.

2.1. Explicit Geometry-Based Methods

Explicit methods operate directly on geometric representations, typically triangle meshes, to construct swept volume boundaries. Analytical solutions exist for restricted motion types, notably the Minkowski sum for pure translations, where Kaul et al. [Kau93] demonstrated efficient computation of exact swept volumes for straight-line paths. However, this approach does not accommodate rotations or general spatial trajectories.

Early specialized approaches addressed specific motion types with varying degrees of success. Korein et al. [Kor86] developed techniques for single-axis rotations by identifying extreme vertex trajectories and constructing piecewise-planar patches. Weld et al. [WL90] established the foundational theoretical insight that the union of all face-sweeps equals the complete swept volume. This principle underpins many subsequent approaches but leaves unresolved the challenge of efficiently determining which patches contribute to the outer boundary.

For general rigid motions, Abrams et al. [AA00] introduced a robust polyhedral sweeping algorithm that directly constructs boundary representations. Their method approximates edge sweeps as ruled surfaces by sampling the motion at discrete time steps and triangulating the resulting patches. These patches are combined with face instances at critical times, and a clipped arrangement is computed to extract the outer boundary. Intelligent heuristics reduce computational complexity by skipping interior edges. Although effective for moderate model complexities, this approach still requires careful handling of geometric intersections and Boolean operations.

Campen et al. [CK10] advanced explicit methods through adaptive discretization, subdividing arbitrary motions into piecewise-linear segments and applying intelligent culling to eliminate redundant surface pieces. Their method handles complex motions more effectively than previous approaches, though combinatorial challenges remain significant in worst-case scenarios. More recently, Von et al. [vDHS12] employed compressed voxelizations and Delaunay refinement to enhance computational efficiency, particularly for industrial geometries.

Despite these advancements, explicit methods consistently face fundamental challenges: robustly managing complex surface interactions during arbitrary motion typically involves restrictive assumptions or computational approximations that can compromise accuracy. As motion complexity increases, the combinatorial explosion of potential feature interactions makes exact boundary calculation increasingly difficult.

2.2. Implicit Field-Based Methods

Implicit methods represent swept volumes using scalar fields, typically signed distance functions (SDFs), to avoid explicitly tracking surface intersections. In this paradigm, the swept volume comprises regions where the distance to the moving object becomes non-positive at any point during the motion.

Sourin et al. [SP96] pioneered functional representations of swept volumes in 4D space-time. Their approach treats moving objects as time-dependent implicit functions and constructs swept volumes either by analytically uniting shape instances or by solving directly for the envelope condition. While elegantly handling arbitrary motions and even deforming objects, this formulation incurs significant computational costs.

A more practical implementation involves discretized sampling. Schroeder et al. [SLL94] converted moving objects into distance fields on 3D grids, progressively updating these fields as objects move to accumulate the swept volume. This “voxel stamping” method robustly manages self-intersections but suffers from

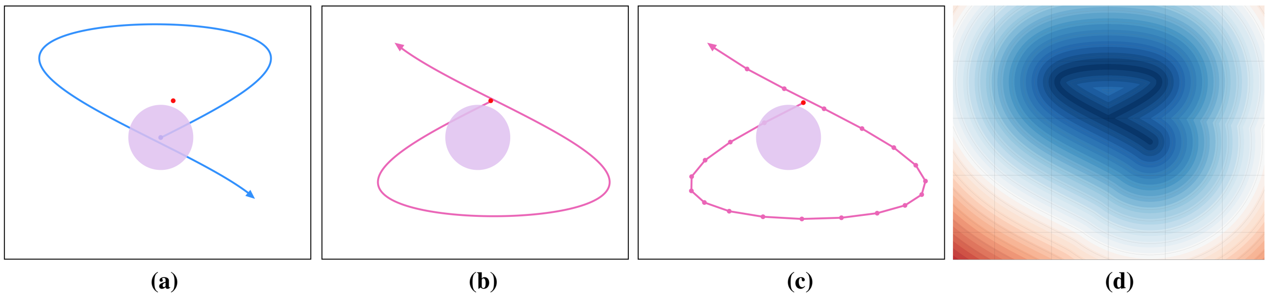


Figure 3: Distance Computation. (a) Given a model (purple circle as an example) and its sweeping trajectory, if we could compute the distance from any spatial point (exemplified by the red point) to the resulting swept volume, we would be able to construct the distance field. However, this is extremely challenging without the explicit swept volume result. (b) Exploiting the relativity of motion, we invert the motion perspective: the model remains stationary while the query point traces an inverse trajectory. The red point lies inside the swept volume if and only if some point along its inverse trajectory lies inside the static model. The distance from the red point to the swept volume can be determined through inverse analysis of trajectory-to-model distances. (c) For complex motions, we partition the inverse trajectory temporally and assume linear motion within each small interval, enabling efficient computation through inverse analysis of segment-to-model distances as detailed in Section 4.1 (d) Based on inverse motion analysis and linear approximation, we obtain the distance field for the swept volume. Note that this illustration shows a single distance field for demonstration; in practice, we employ a multi-field strategy where each distance field corresponds to a swept volume from a short time interval.

resolution-dependent artifacts and high memory demands. Researchers have mitigated these limitations through adaptive sampling and hierarchical data structures, enhancing efficiency while retaining the robustness advantages of field-based approaches.

The most significant recent advancement in implicit methods comes from [SAJ21], who developed a numerical continuation approach in 4D space-time. Their method represents both the moving object and its swept volume as SDFs, tracing the swept surface as connected solution curves in spacetime. However, these 4D space-time methods fundamentally require implicit function representations as input, significantly limiting their practical use in domains where explicit geometric representations dominate. Furthermore, computing distances to swept volumes of complex geometries undergoing intricate motions can encounter optimization difficulties, such as susceptibility to convergence to local minima, which undermines theoretical correctness guarantees.

3. OVERVIEW

Our method addresses two fundamental challenges in swept volume computation: (1) efficiently calculating the distance from a point to the swept volume, and (2) accurately preserving fine geometric details that emerge during the sweeping process.

To tackle the distance computation challenge, we invert the conventional perspective on motion analysis. Instead of tracking objects as they move through space, we keep the object fixed and analyze the inverse trajectories of spatial points. This transformation simplifies the complex problem of computing distances to swept volumes into the more manageable task of measuring distances between a static object and individual point trajectories, thereby providing a robust foundation for our extraction method. Figure 3 illustrates this concept through a 2D example for intuitive understanding.

To address the challenge of detail preservation, we employ tetrahedral discretization combined with a multi-field representation strategy. Unlike traditional methods that maintain a single scalar field within each element, our approach preserves multiple distance fields per tetrahedron—each corresponding to a swept volume segment generated during a specific time interval. To extract the final swept surface from these multiple fields, we introduce four-dimensional incremental cutting, processing the lower envelope of these fields in higher-dimensional space. This multi-field approach effectively captures intricate geometric features at trajectory intersections, overcoming the limitations inherent in single-field methods.

Section 4 presents our two core algorithms: distance computation via inverse motion and isosurface extraction using four-dimensional incremental cutting. Section 5 details the full implementation pipeline, demonstrating how these components effectively balance computational efficiency with geometric precision.

4. METHOD

4.1. Distance Computation via Inverse Motion

Computing the signed distance from an arbitrary point to the swept volume boundary constitutes a fundamental operation in swept volume extraction. However, given a short time interval $[t_0, t_1]$ and a query point $\mathbf{q} \in \mathbb{R}^3$, determining the distance from $\mathbf{q}$ to the swept volume of model $\mathcal{M}$ without explicitly constructing the swept volume presents significant computational challenges, as illustrated in Figure 4(a). In this subsection, we focus on scenarios with sufficiently short time intervals, while longer time intervals requiring piecewise linear approximation will be addressed in Section 5.

Motion Inversion. To determine the spatial relationship between a point and a swept volume we invert the conventional perspective on motion to establish a more tractable computational framework. Since motion is inherently relative, we can reformulate the problem by fixing the object's position and treating the static query point $\mathbf{q}$ as undergoing inverse motion, thus generating a trajectory through

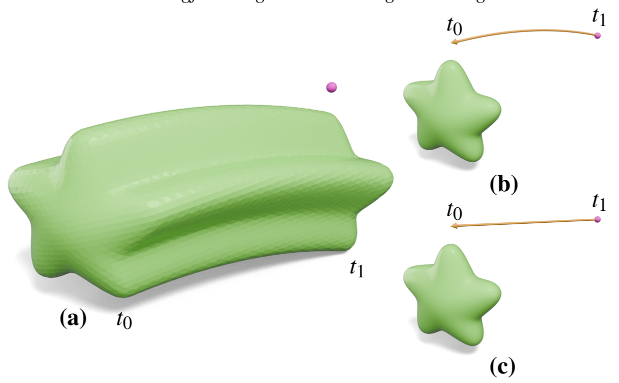


Figure 4: Illustration of our relative motion analysis for short time intervals. (a) Generate the swept volume by moving the object along the specified trajectory. (b) Keep the object stationary and move the observation point p along the reverse trajectory. (c) Approximate the short, curved trajectory with a straight-line segment. This reduces the determination of whether p lies within the swept volume to a simpler intersection test between the straight-line segment and the original stationary object. Note that we illustrate the concept using simple motion within a short time interval here for demonstration purposes. For longer time intervals or complex motions, we subdivide the time period into finer segments and assume linear inverse trajectories within each segment, as shown in Figure 8.

space (Figure 4(b)). Formally, if we define a continuous transformation function $\mathbf{f}(t) : [t_0,t_1] \to SE(3)$ that maps time to a rigid transformation matrix, then the position of model $\mathcal{M}$ at time t is given by $\mathcal{M}_t = \mathbf{f}(t)\mathcal{M}$.

Under our inverted perspective, the query point's trajectory can be expressed as:

$$\mathbf{q}(t) = \mathbf{f}^{-1}(t)\mathbf{q}, \quad t \in [t_0,t_1] \tag{1}$$

A critical observation is that a point lies inside the swept volume if and only if its inverse trajectory intersects the static model—an exact correspondence that forms the theoretical foundation of our approach. Based on this principle, we can determine the distance from $\mathbf{q}$ to the swept volume of $\mathcal{M}$ during $[t_0,t_1]$ through distance analysis between the trajectory of $\mathbf{q}(t)$ and the static model $\mathcal{M}$.

Linear Trajectory Approximation. For sufficiently small time intervals $[t_0,t_1]$, the query point's motion can be approximated as linear, forming a line segment $L(t_0,t_1)$ connecting the positions at the interval endpoints:

$$L(t_0,t_1) = \overline{\mathbf{q}(t_0)\mathbf{q}(t_1)} = \{\lambda\mathbf{q}(t_0) + (1-\lambda)\mathbf{q}(t_1) | \lambda \in [0,1]\} \tag{2}$$

As depicted in Figure 4(c), this simplification transforms our problem into computing the distance from line segment $L_{\mathbf{q}}$ to the static model $\mathcal{M}$:

$$d(\mathbf{q}, \mathrm{SweepVol}(\mathcal{M},[t_0,t_1])) \approx \min_{p \in L(t_0,t_1)} d(p,\mathcal{M}) \tag{3}$$

where d denotes the signed Euclidean distance.

This transformation provides a computationally efficient approximation that preserves the essential geometric characteristics of the swept volume with high fidelity. The trajectory-based formulation significantly simplifies the computation process while maintaining the robustness necessary for accurate swept volume reconstruction.

Note that we assume $[t_0,t_1]$ represents a very short time interval here, allowing the inverse trajectory to be approximated with a single line segment. In practice, the complete temporal motion is divided into multiple short intervals, with each inverse trajectory segment assumed to be linear, as detailed in Section 5.1.

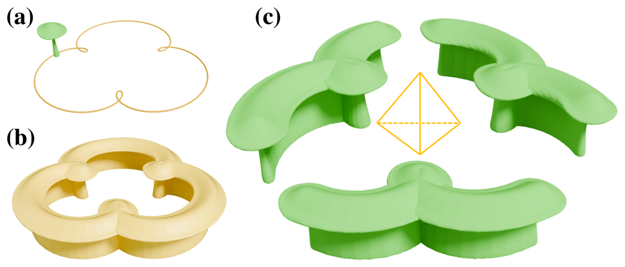


Figure 5: Multiple distance field representation in our approach. (a) Input model and motion trajectory. (b) Complete swept volume. (c) The entire motion is temporally partitioned into multiple segments, with each segment generating a partial swept volume. Each partial swept volume contributes a distance field, illustrated here with three segments providing three distinct distance fields to the tetrahedron in the discretization. In practice, finer temporal discretization is employed.

4.2. 4D Incremental Cutting for Isosurface Extraction

Isosurface extraction from implicit fields is a fundamental task in computational geometry [WMW86, dALJ*15]. Traditional isosurface extraction methods such as Marching Cubes [LC87] and Dual Contouring [JLSW02] operate on scalar fields where a single distance value is stored at each vertex of a spatial tessellation. These approaches process each volumetric element where the zero-isosurface passes through, using the distance values at element vertices (and potentially gradient information) to reconstruct surface fragments. However, a significant limitation of these conventional methods is their fundamental assumption of surface simplicity within each element, which renders them inadequate for capturing complex geometric features and intricate surface structures that emerge in swept volumes, particularly at trajectory intersections.

To overcome this limitation, our method employs tetrahedral tessellation of the computational domain and, critically, maintains multiple distance fields within each tetrahedron. Each of these fields corresponds to the signed distance field of a swept volume generated by model $\mathcal{M}$ during a different time interval of its motion. As shown in Figure 5, the complete trajectory is divided into three segments, with the swept volume formed by each trajectory segment providing a separate distance field for the tetrahedron.

Let us denote the swept volume of model $\mathcal{M}$ during time interval $[t_i^s, t_i^e]$ as $\mathrm{SV}_i(\mathcal{M})$. For a tetrahedron t, we denote the set of distance fields as $\{\pi_i\}_{i=1}^m$, where each π_i corresponds to the signed distance field to $\mathrm{SV}_i(\mathcal{M})$.

Following the approach introduced in [WSW*25, XWX*22], we

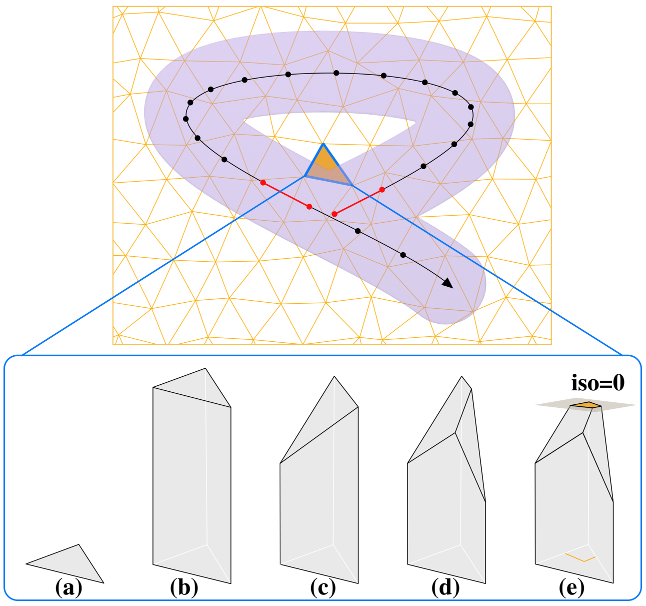


Figure 6: Feature preservation through multiple distance fields and incremental cutting in 2D swept volume extraction. Taking the highlighted triangle as an example, it stores two distance fields from swept volumes during different time segments, with trajectory segments from different time periods illustrated and contributing segments marked in red. (a) Original triangle with multiple distance fields defined within. (b) Unbounded triangular prism formed by extending the triangle infinitely along the z-axis. (c,d) Progressive incremental cutting operations using hyperplanes representing individual distance fields, where the height corresponds to distance field values, resulting in a lower envelope. (e) Intersecting the resulting prism with the $z = 0$ plane and projecting the intersection curve back to 2D space to obtain the zero-contour within the triangle.

assume linear variation of each distance field within the tetrahedron t. Specifically, each distance field π_i is encoded by four values $\pi_i = \{d_{1,i}, d_{2,i}, d_{3,i}, d_{4,i}\}$, where $d_{k,i}$ represents the signed distance from the k-th vertex of the tetrahedron to $\mathrm{SV}_i(\mathcal{M})$. For any point x within t with barycentric coordinates (v_1, v_2, v_3, v_4), the signed distance from x to $\mathrm{SV}_i(\mathcal{M})$ can be approximated through linear interpolation:

$$\pi_i(x) = v_1 d_{1,i} + v_2 d_{2,i} + v_3 d_{3,i} + v_4 d_{4,i} \tag{4}$$

The actual distance field within t, accounting for all component fields, is determined by taking the minimum value across all distance fields:

$$d(x) = \min_{i \in \{1,2,\dots,m\}} \pi_i(x) \tag{5}$$

Our goal is to compute the zero-isosurface of this combined distance field. For easier understanding, we consider a simplified 2D case. Figure 6 illustrates this concept through a 2D example for intuitive understanding, where the motion time is divided into multiple segments, with each segment's swept volume generating a distance field computed using the approach shown in Figure 4. In this scenario, multiple distance fields exist within the highlighted triangle, with each field visualized as a plane in 3D space where the z-axis represents the interpolated distance value at each point. We perform incremental linear cutting operations on an unbounded triangular prism (whose base is the original triangle, extending infinitely upward and downward) using these planes, continuously updating the lower envelope. The zero-contour is then obtained by intersecting this lower envelope with the $z = 0$ plane and projecting the resulting intersection curve back to the 2D domain.

Extending this approach to 3D, we implement incremental cutting in 4D space to extract the zero-isosurface of the distance field, following the approach described in [WSW*25]. Each tetrahedron, along with its associated distance fields, defines a set of hyperplanes in 4D space. By computing the lower envelope of these hyperplanes and intersecting it with the zero-hyperplane (where the distance equals zero), then projecting the result back to 3D space, we obtain the zero-isosurface structure within the tetrahedron. This 4D incremental cutting approach faithfully captures complex geometric features generated during the object's motion, including sharp features and intricate surface structures that would otherwise be lost with traditional isosurface extraction methods. Algorithm 1 presents the pseudocode.

Algorithm 1 Isosurface Extraction via 4D Incremental Cutting

```
Require: Tetrahedron $\tau$, Distance fields $\{\pi_1, \pi_2, \dots, \pi_m\}$ stored in $\tau$
Ensure: Swept volume surface within tetrahedron $\tau$
1: Initialize 4D prismatic structure $P$ with base $\tau$ and infinite height range $(-\infty, +\infty)$ in $w$-dimension
2: for each distance field $\pi_i$ in $\{\pi_1, \pi_2, \dots, \pi_m\}$ do
3:     Construct 4D hyperplane $H_i$ from distance values of $\pi_i$ at tetrahedron vertices
4:     Cut $P$ by hyperplane $H_i$ and retain the portion below $H_i$ (lower envelope)
5: end for
6: Intersect the final lower envelope structure $P$ with hyperplane $w = 0$
7: Extract intersection surface $S$ between $w = 0$ hyperplane and the lower envelope
8: Project surface $S$ back to 3D space to obtain swept volume boundary within $\tau$
9: return Swept volume surface mesh in $\tau$
```

5. IMPLEMENTATION

Our implementation consists of three phases:

- **Spatiotemporal Discretization**: We partition the spatial domain into tetrahedral elements and divide the motion into small time intervals that support our linear trajectory approximation.
- **Distance Field Propagation**: For each time interval, we propagate distance fields from seed tetrahedra throughout the tetrahedral discretization using a competition mechanism, ensuring each tetrahedron receives the necessary fields for isosurface extraction.

- **Swept Volume Extraction**: We apply four-dimensional incremental cutting to each tetrahedron independently, converting multiple distance fields into a coherent swept volume boundary.

We now describe each component of this pipeline in detail.

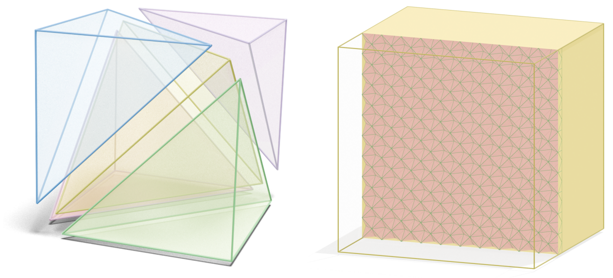


Figure 7: Spatial discretization. (a) A cube subdivided can be into five tetrahedra. (b) Tetrahedral tessellation of the computational domain where we first partition space into uniform cubes, then subdivide each cube as shown in (a).

5.1. Spatiotemporal Discretization

Our method employs discretization in both spatial and temporal dimensions to facilitate subsequent computational operations. This discretization strategy establishes the foundation for our swept volume computation framework.

Spatial Discretization. We utilize tetrahedral tessellation to discretize the computational domain. Specifically, we first partition the solution space using uniform cubic elements, then subdivide each cube into five tetrahedra as illustrated in Figure 7(a). Through careful arrangement, we maintain spatial consistency across the tetrahedral mesh. Figure 7(b) presents an example of our tetrahedral tessellation applied to a computational domain.

Temporal Discretization. In the temporal dimension, we divide the entire motion time interval into N segments. Within each temporal segment, we assume the inverse trajectories of spatial points can be well-approximated as linear segments, as shown in Figure 8. This piecewise linear approximation is consistent with the motion perspective inversion described in Section 4.1 and allows us to handle complex motion paths while maintaining computational efficiency. The resolution of this temporal discretization can be adjusted based on the complexity of the original motion and the desired accuracy of the final swept volume representation.

5.2. Spatial Propagation of Piecewise Linear Motion Distance Fields

We consider a normalized motion time interval $[0,1]$ discretized into N segments, with each segment spanning $[t_i,t_{i+1}]$. For model $\mathcal{M}$, the swept volume formed during time interval $[t_i,t_{i+1}]$ is denoted as $\mathrm{SV}(\mathcal{M},t_i,t_{i+1})$. Each segment-specific swept volume generates a distance field throughout the computational domain, with the complete swept volume defined by taking the minimum value among all these fields at each point.

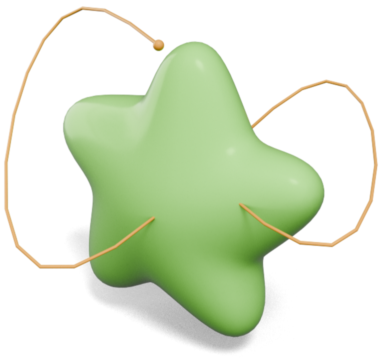


Figure 8: Temporal discretization. The green model remains stationary while orange curves show inverse trajectories of spatial points. Within each discrete time interval, these trajectories are approximated as linear segments.

Algorithm 2 Line Segment to Model Distance Query

```
Require: Line segment $L = \{p_0, p_1\}$, Model $\mathcal{M}$
Ensure: Signed distance $d$
 1: if line segment $L$ lies completely outside model $\mathcal{M}$ then
 2:     $d \leftarrow$ distance from $L$ to $\mathcal{M}$ using FCPW library
 3: else
 4:     Extract the portion of $L$ that lies inside $\mathcal{M}$ as $L_{\text{interior}}$
 5:     Sample 10 evenly distributed points along $L_{\text{interior}}$
 6:     $d \leftarrow +\infty$
 7:     for each sampled point $p$ do
 8:         Compute signed distance $d_p$ from $p$ to $\mathcal{M}$
 9:         $d \leftarrow \min(d, d_p)$
10:     end for
11: end if
12: return $d$
```

To compute distances from points to these swept volumes, we employ the trajectory-based approach described in Section 4.1, which reduces to analyzing the relationship between line segments and the static model. According to Equation 3, when the line segment lies completely outside the model, the minimum distance from all points on the segment to the model equals the segment-to-model distance, which we compute using the FCPW library [Saw21]. However, when portions of the segment lie inside the model, we discretize the interior portion along the segment into sample points, compute the signed distance from each point to the model, and take the minimum signed distance. This discretization approach is necessary because we require the minimum signed distance rather than the minimum absolute distance, and existing libraries cannot provide the required computation when segments lie partially inside the model. Algorithm 2 presents the pseudocode for line segment-to-model distance calculation. While this introduces some approximation in the absolute distance values, the sign of the distance remains correct, which is crucial for accurate inside/outside determination. In our experiments, we used 10 sample points for this discretization.

As described in Section 4.2, within each tetrahedron, we rep-

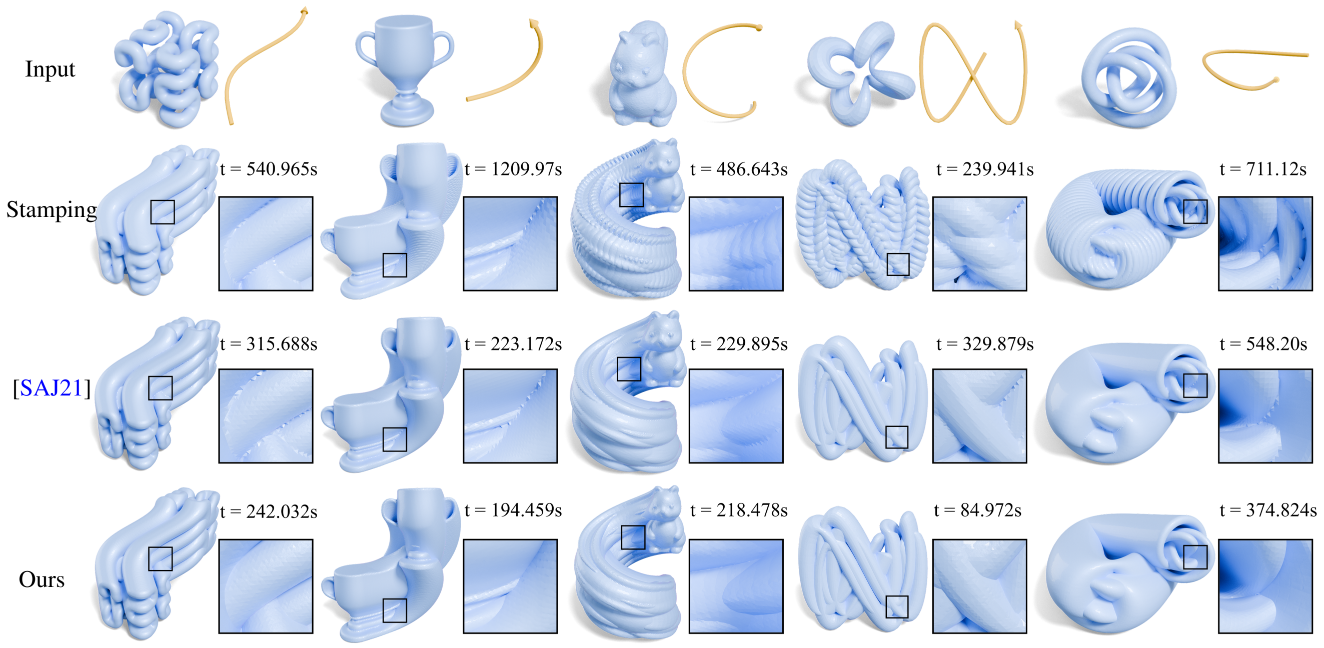


Figure 9: Visual comparison of swept volumes generated by our method against Stamping and [SAJ21] across different models and motion types. Our method preserves sharp features while avoiding the zigzag artifacts present in the Stamping approach.

Algorithm 3 Distance Field Propagation for Swept Volume Computation

```
Require: Tetrahedral mesh $\mathcal{T}$, Model $\mathcal{M}$, Global motion function $f(t)$, $N$ time intervals $[t_i, t_{i+1}]$
Ensure: Distance fields stored in each tetrahedron
 1: Initialize empty priority queue $Q$ for distance propagation
 2: for each time interval $[t_i, t_{i+1}]$ do
 3:     Randomly select fixed number of seed tetrahedra inside $\mathcal{M}(t_i)$
 4:     for each seed tetrahedron $\tau$ do
 5:         Add propagation event $(\tau, [t_i, t_{i+1}])$ to priority queue $Q$
 6:     end for
 7: end for
 8: while $Q$ is not empty do
 9:     Extract top-priority event $(\tau, [t_i, t_{i+1}])$ from $Q$
10:     Compute distance field $\pi$ within $\tau$ for swept volume during time interval $[t_i, t_{i+1}]$ using motion inversion
11:     if $\pi$ is not defeated by existing fields in $\tau$ and not all distance values of $\pi$ at vertices are positive then
12:         Store $\pi$ in the distance field list of $\tau$
13:         for each neighboring tetrahedron $\tau'$ of $\tau$ do
14:             Add propagation event $(\tau', [t_i, t_{i+1}])$ to $Q$
15:         end for
16:     end if
17: end while
```

resent each distance field π_i by four values corresponding to the signed distances at its vertices: $\pi_i = d_{1,i}, d_{2,i}, d_{3,i}, d_{4,i}$. To determine which distance fields contribute to each tetrahedron, we adopt the competition mechanism from prior work [XWX*22]. For two distance fields π_i and π_j within tetrahedron t, we consider π_j to be defeated by π_i if:

$$d_{1,i} < d_{1,j} \text{ and } d_{2,i} < d_{2,j} \text{ and } d_{3,i} < d_{3,j} \text{ and } d_{4,i} < d_{4,j}. \quad (6)$$

Our goal is to identify all non-defeated distance fields for each Useful tetrahedron, which collectively define the local swept volume boundary.

For each time interval $[t_i, t_{i+1}]$, we select a fixed number of seed tetrahedra that lie inside $\mathcal{M}(t)$ for some time t within this interval and initialize them with the corresponding segment's distance field. These distance fields propagate to neighboring tetrahedra only if they remain undefeated in current tetrahedron and not all four distance values at the tetrahedron vertices are positive (as fields with all positive values contribute no meaningful information to swept volume surface extraction within that tetrahedron). Each tetrahedron stores all undefeated distance fields it receives during this process, which collectively define its contribution to the swept volume boundary. This propagation process is efficiently managed using a priority queue, and the algorithm terminates when no distance fields remain to be propagated. This approach ensures that all tetrahedra containing relevant portions of the swept volume receive their appropriate distance fields. The detailed implementation of this propagation strategy follows the approach described in [XWX*22]. For better understanding, we provide the pseudocode for distance field propagation in Algorithm 3.

5.3. Isosurface Extraction

After determining all contributing distance fields for each tetrahedron, we extract the swept volume boundary using the four-dimensional incremental cutting approach described in Section 4.2. This process converts the multiple distance field representations within each tetrahedron into a coherent surface representation of the swept volume boundary.

Our implementation of the 4D incremental cutting procedure follows the approach presented in [WSW*25], adapted to the specific context of swept volume computation. A key advantage of our method is that isosurface extraction can be performed independently for each tetrahedron, without inter-element dependencies. This property makes the algorithm well-suited for parallel acceleration, enabling efficient execution on multi-core CPUs. In our implementation, we exploit this parallelism to substantially reduce computation time, particularly for complex models and motion paths where many tetrahedra contain non-trivial swept volume contributions.

6. EVALUATION

6.1. Experimental Setting

Hardware Environment. Our algorithm was implemented in C++ on a computing platform equipped with a 3.4 GHz Intel Core i7-14700K 20-Core CPU, 64 GB of memory, and Windows 11 operating system. We employed 28 threads for parallel acceleration during the isosurface extraction phase using the OpenMP framework. No post-processing smoothing was applied in our experiments.

Baseline Methods. We compared our approach against two representative methods:

- **Stamping**: This method, commonly employed in applications such as Adobe Medium, involves sampling the moving object at discrete intervals. At each sample, a signed distance field (SDF) representation is generated. These SDFs are combined by taking the minimum value at each spatial location, followed by extracting the zero-isosurface to form the swept volume boundary.
- **[SAJ21]**: This optimization-based approach calculates the minimal distance values at each spatial grid vertex throughout the sweeping process, utilizing dual contouring to extract the final swept volume surface.

Consistent discretization parameters were applied across all methods. The Spacetime Numerical Continuation method requires intermediate states for trajectory interpolation, while our method and the Stamping approach need temporal discretization. We uniformly adopted a sampling rate of 50 time steps for all experiments. Surface extractions were performed at a spatial resolution of 256^3.

Additionally, we initialized distance field computations from 100 randomly selected seed tetrahedra within each relevant time interval, subsequently propagating fields throughout the spatial discretization.

6.2. Comparisons

Visual Comparison. Figure 9 visually compares swept volumes generated by our method against Stamping and [SAJ21] across various models and motion types. Due to varying input parameter formats across different methods, we limit the motion to translation to maintain trajectory consistency. The Stamping method exhibits noticeable zigzag artifacts due to discrete temporal sampling. The method of [SAJ21], while maintaining temporal continuity via optimization-based distance computations, fails to capture complex geometric features due to its reliance on a single distance field. Our method, employing motion perspective inversion and 4D incremental cutting, preserves geometric details effectively, achieving superior accuracy with reduced computation time.

Table 1: Quantitative comparison for different methods. **Bold values** indicate optimal performance. HD and CD are normalized by the bounding box diagonal length.

	Stamping	[SAJ21]	Ours
CD (‰) ↓	0.6801	0.1421	**0.1345**
HD (%) ↓	1.199	0.680	**0.645**
Time (s) ↓	181	88.3	**80.0**

Quantitative Comparison. For quantitative evaluation, we selected a case with a known analytical solution: a sphere moving along a circular trajectory (Figure 11). We sampled points directly from the analytically defined swept surface to measure Chamfer Distance (L_1 CD) and Hausdorff Distance (HD). Table 1 summarizes these metrics, demonstrating that our method outperforms baseline approaches.

6.3. Time

The computational performance of our algorithm is governed by two primary discretization parameters: spatial resolution and temporal sampling density. To quantify their impact on algorithm efficiency, we conducted a comprehensive performance analysis using a dataset of 100 randomly selected geometric models under simple motions involving translation and rotation. We measured execution time across systematically varied parameter configurations to characterize performance scaling properties. For spatial resolution experiments, we maintained a fixed temporal discretization of 50 intervals while varying the spatial grid density. Conversely, for temporal discretization analysis, we established a constant spatial resolution of 256^3 voxels while adjusting the number of temporal intervals. Figure 12 presents the empirical performance data, illustrating computational complexity as a function of these fundamental discretization parameters.

6.4. Ablation Study

Our method contains two critical hyperparameters: 1) The temporal discretization parameter, determining the number of time intervals for piecewise linear motion approximation. 2) The spatial resolution parameter, controlling the grid density for extraction. Additionally, we include swept volume computation times for all different parameter settings.

Time Steps. Temporal discretization resolution directly influences the fidelity of our piecewise linear approximation and consequently the accuracy of computed swept volumes. Figure 13 illustrates the

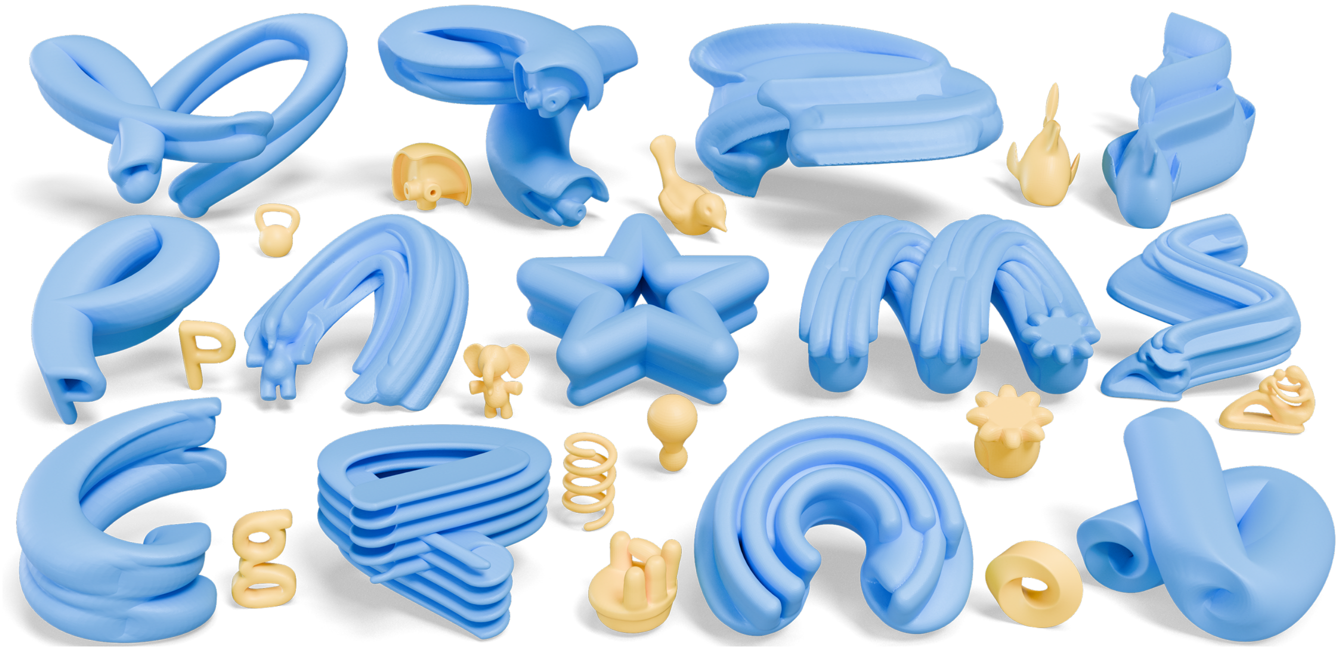


Figure 10: Diverse swept volume results generated by our algorithm for various geometric models undergoing different motion trajectories.

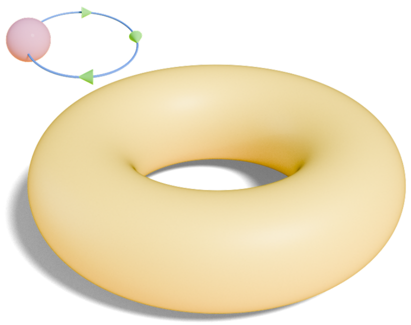


Figure 11: Swept volume generated by a sphere traversing a circular trajectory. This configuration yields an analytically defined surface, facilitating quantitative evaluation through direct point sampling without explicit construction of a reference model.

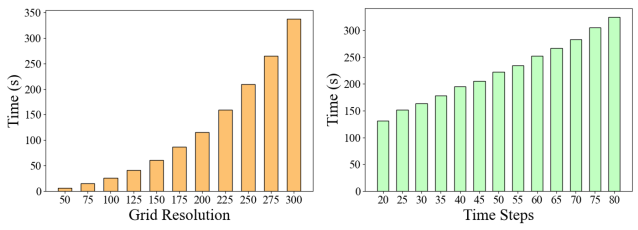


Figure 12: Performance with respect to spatial and temporal discretization. Results represent average computation times across 100 test models.

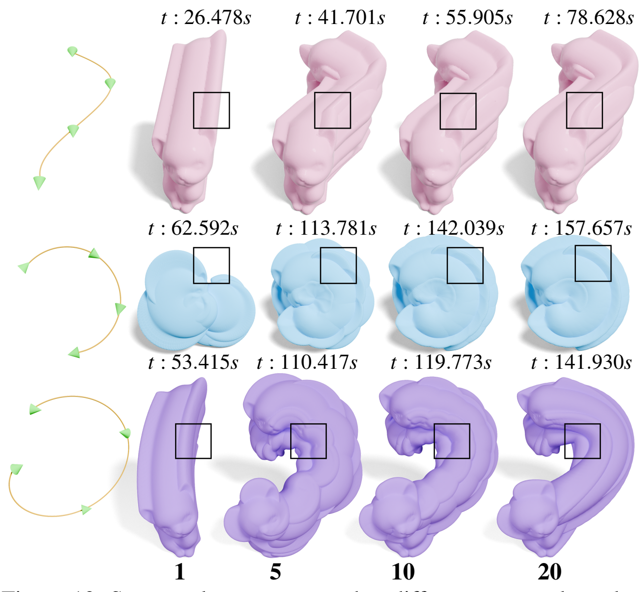


Figure 13: Swept volumes computed at different temporal resolutions with spatial resolution fixed at 256^3.

resultant swept volumes at temporal resolutions of 1, 5, 10, and 20 discrete intervals. At minimal temporal subdivision, the piecewise linear approximation becomes invalid as an accurate representation of the continuous motion, resulting in substantial geometric deviation from the theoretical surface. Progressive refinement of the temporal discretization demonstrates convergent behavior toward the analytical solution, with notable improvements in geometric continuity at inter-interval boundaries.

Grid Resolution. Figure 14 presents swept volume extraction results at multiple spatial discretization resolutions:25^3, 50^3, 100^3, and 256^3 voxels. Higher spatial resolution enables more precise approximation of the continuous distance fields, resulting in improved boundary representation and preservation of fine geometric details that would otherwise be lost at coarser discretization levels.

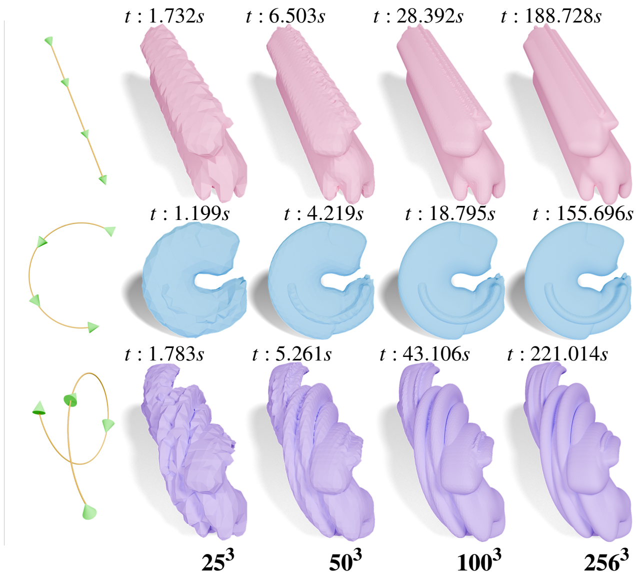


Figure 14: Extracted sweep volume at different spatial resolutions with temporal resolution fixed at 50. Flat rendering is used to better highlight the differences across resolutions.

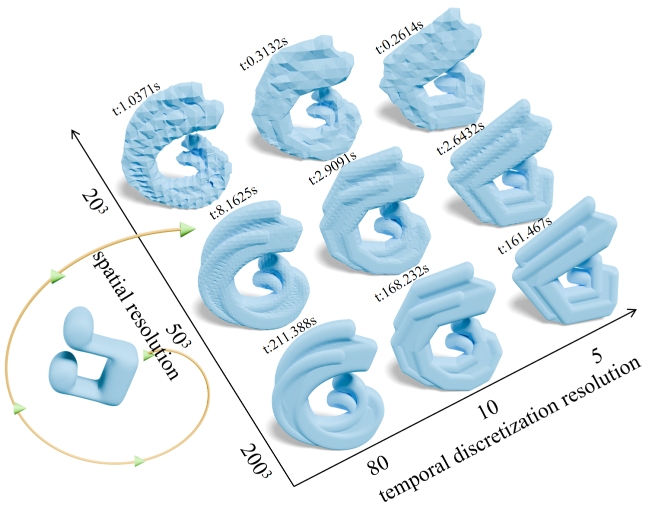


Figure 15: Visual comparison across different temporal and spatial discretization resolutions. Flat rendering is used to better highlight the differences across resolutions.

Figure 15 demonstrates results in a 3×3 grid where each row and column represents different spatiotemporal resolution settings. Additional examples showcasing our method's robustness with various models and motion patterns are provided in Figure 10.

7. LIMITATIONS AND FUTURE WORK

Our method currently has three primary limitations that represent important directions for future research.

Temporal Discretization. Our method currently employs uniform temporal discretization, dividing the entire motion into equally-sized time intervals. This represents a significant limitation, as it inefficiently allocates computational resources by treating periods of slow, smooth motion the same as rapid or complex motion phases. Adaptive temporal discretization would offer substantial benefits—using fewer time segments during slow movements where linear approximation is highly accurate, and employing finer discretization during rapid motions where more precision is required. Such an approach could significantly improve both computational efficiency and result quality while reducing artifacts at segment boundaries.

Spatial Resolution. Similarly, our framework relies on uniform spatial tessellation throughout the computational domain. This limitation becomes apparent when considering that swept volume surfaces contain both geometrically complex regions requiring high resolution and relatively flat areas where coarse tessellation would suffice. Adaptive spatial discretization could potentially yield substantial performance gains by concentrating computational resources in complex regions. However, this presents fundamental challenges in our framework since we extract swept volume surfaces independently from each tetrahedron. Adaptive tessellation would cause misalignment issues between adjacent elements with different resolutions, leading to inconsistent surface reconstruction. Investigating hybrid approaches that maintain surface consistency while enabling selective spatial refinement represents an important avenue for future research.

Complex Transformations. Our inverse trajectory analysis framework assumes that a unified inverse transformation can be applied to spatial points. However, when objects undergo scaling, animation, or deformation, different regions of the model exhibit distinct motion behaviors, making unified inverse transformation impossible. More complex motions such as animation and deformation present fundamental challenges to our current approach. Extending support for such transformations remains an important direction for future work.

8. CONCLUSION

In this paper, we introduced a novel swept volume computation framework leveraging motion perspective inversion and a multi-field extraction strategy. By fixing the object and analyzing inverse point trajectories, our approach simplifies the complex problem of trajectory analysis. Furthermore, our framework maintains multiple distance fields within each tetrahedral element and utilizes four-dimensional incremental cutting, effectively capturing intricate geometric features emerging during motion. Extensive experiments across a variety of models and motion types validate the superior effectiveness and efficiency of our proposed method compared to existing approaches.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Key R&D Program of China (2022YFB3303200), and the National Natural Science Foundation of China (U23A20312, 62272277).

References

[AA00] ABRAMS S., ALLEN P. K.: Computing swept volumes. *The Journal of Visualization and Computer Animation 11*, 2 (2000), 69–82. 2

[AMS14] ADSUL B., MACHCHHAR J., SOHONI M.: A computational framework for boundary representation of solid sweeps. *Computer-Aided Design and Applications 12* (03 2014). doi:10.1080/16864360.2014.962430. 1

[AMY97] ABDEL-MALEK K., YEH H.: Path trajectory verification for robot manipulators in a manufacturing environment. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 211*, 7 (1997), 547–556. 1

[AMYB01] ABDEL-MALEK K., YANG J., BLACKMORE D. L.: On swept volume formulations: implicit surfaces. *Comput. Aided Des. 33* (2001), 113–121. 1

[AMYBJ06] ABDEL-MALEK K., YANG J., BLACKMORE D. L., JOY K. I.: Swept volumes: Fundation, perspectives, and applications. *Int. J. Shape Model. 12* (2006), 87–127. 1

[Bar86] BARR A. H.: Ray tracing deformed surfaces. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, Association for Computing Machinery, p. 287–296. URL: https://doi.org/10.1145/15922.15918, doi:10.1145/15922.15918. 2

[CK10] CAMPEN M., KOBBELT L.: Polygonal boundary evaluation of minkowski sums and swept volumes. *Computer Graphics Forum 29*, 5 (2010), 1613–1622. doi:https://doi.org/10.1111/j.1467-8659.2010.01770.x. 2

[dALJ*15] DE ARAÚJO B. R., LOPES D. S., JEPP P., JORGE J. A., WYVILL B.: A survey on implicit surface polygonization. *ACM Comput. Surv. 47*, 4 (May 2015). URL: https://doi.org/10.1145/2732197, doi:10.1145/2732197. 4

[ITA*23] IWASAKI T., TAKASE Y., ARNOLD S., TAKESHITA K., YAMAZAKI K.: Online motion planning based on swept volume search with replanning using sequential quadratic programming. *Advanced Robotics 37* (2023), 737 – 750. 1

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (July 2002), 339–346. URL: https://doi.org/10.1145/566654.566586, doi:10.1145/566654.566586. 4

[JSS24] JOHO D., SCHWINN J., SAFRONOV K.: Neural implicit swept volume models for fast collision detection. *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024), 15402–15408. URL: https://api.semanticscholar.org/CorpusID:267897342. 1

[Kau93] KAUL A.: *Computing Minkowski sums*. PhD thesis, USA, 1993. UMI Order No. GAX93-33799. 1, 2

[Kor86] KOREIN J. U.: *A geometric investigation of reach*. MIT Press, Cambridge, MA, USA, 1986. 1, 2

[KRL*07] KIM Y. J., REDON S., LIN M. C., MANOCHA D., TEMPLEMAN J. N.: Interactive continuous collision detection using swept volume for avatars. *PRESENCE: Teleoperators and Virtual Environments 16* (2007), 206–223. 1

[Lar20] LAROCHE C.: An implicit representation of swept volumes based on local shapes and movements. *ArXiv abs/2003.11527* (2020). 1

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, Association for Computing Machinery, p. 163–169. doi:10.1145/37401.37422. 4

[MFJQ*16a] MURRAY S., FLOYD-JONES W., QI Y., SORIN D. J., KONIDARIS G. D.: Robot motion planning on a chip. In *Robotics: Science and Systems* (2016). 1

[MFJQ*16b] MURRAY S., FLOYD-JONES W., QI Y., SORIN D. J., KONIDARIS G. D.: Robot motion planning on a chip. In *Robotics: Science and Systems* (2016), vol. 6. 1

[MGS17] MCGRAW T., GARCIA E., SUMNER D.: Interactive swept surface modeling in virtual reality with motion-tracked controllers. In *Proceedings of the Symposium on Sketch-Based Interfaces and Modeling* (2017), pp. 1–9. 1

[PPSZ05] PETERNELL M., POTTMANN H., STEINER T., ZHAO H.: Swept volumes. *Computer-Aided Design & Applications 2* (01 2005). doi:10.1080/16864360.2005.10738324. 1

[SAJ21] SELLÁN S., AIGERMAN N., JACOBSON A.: Swept volumes via spacetime numerical continuation. *ACM Trans. Graph. 40*, 4 (July 2021). doi:10.1145/3450626.3459780. 2, 3, 7, 8

[Saw21] SAWHNEY R.: Fcpw: Fastest closest points in the west, 2021. 6

[SG05] SHIH F. Y., GADDIPATI V.: Geometric modeling and representation based on sweep mathematical morphology. *Information Sciences 171*, 1 (2005), 213–231. doi:https://doi.org/10.1016/j.ins.2004.04.002. 1

[SLL94] SCHROEDER W., LORENSEN W., LINTHICUM S.: Implicit modeling of swept surfaces and volumes. In *Proceedings Visualization '94* (1994), pp. 40–45. doi:10.1109/VISUAL.1994.346339. 2

[SP96] SOURIN A., PASKO A.: Function representation for sweeping by a moving solid. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (1996), 11–18. doi:10.1109/2945.489382. 2

[vDHS12] VON DZIEGIELEWSKI A., HEMMER M., SCHÖMER E.: High quality conservative surface mesh generation for swept volumes. In *2012 IEEE International Conference on Robotics and Automation* (2012), pp. 764–769. doi:10.1109/ICRA.2012.6224921. 2

[WHS00] WANG G., HUA X., SUN J.: The differential equation algorithm for general deformed swept volumes. *Journal of Computer Science and Technology 15* (2000), 604–610. 1

[WL90] WELD J. D., LEU M. C.: Geometric representation of swept volumes with application to polyhedral objects. *Int. J. Rob. Res. 9*, 5 (Sept. 1990), 105–117. doi:10.1177/027836499000900507. 1, 2

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer - VC 2* (08 1986), 227–234. doi:10.1007/BF01900346. 4

[WSW*25] WANG P., SONG J., WANG L., XIN S., YAN D.-M., CHEN S., TU C., WANG W.: Towards voronoi diagrams of surface patches. *IEEE Transactions on Visualization and Computer Graphics* (2025), 1–15. doi:10.1109/TVCG.2025.3531445. 4, 5, 8

[WZZ*24] WANG J., ZHANG T., ZHANG Q., ZENG C., YU J., XU C., XU L., GAO F.: Implicit swept volume sdf: Enabling continuous collision-free trajectory generation for arbitrary shapes. *ACM Transactions on Graphics (TOG) 43* (2024), 1 – 14. 1

[Xav97] XAVIER P. G.: Fast swept-volume distance for robust collision detection. *Proceedings of International Conference on Robotics and Automation 2* (1997), 1162–1169 vol.2. 1

[XWX*22] XIN S., WANG P., XU R., YAN D., CHEN S., WANG W., ZHANG C., TU C.: Surfacevoronoi: Efficiently computing voronoi diagrams over mesh surfaces with arbitrary distance solvers. *ACM Trans. Graph. 41*, 6 (Nov. 2022). doi:10.1145/3550454.3555453. 4, 7