# Reliable Sweeps

Xinyu Zhang  Young J. Kim
Ewha Womans University
Seoul, Korea
{zhangxy,kimy}@ewha.ac.kr

Dinesh Manocha
University of North Carolina
Chapel Hill, U.S.A.
dm@cs.unc.edu

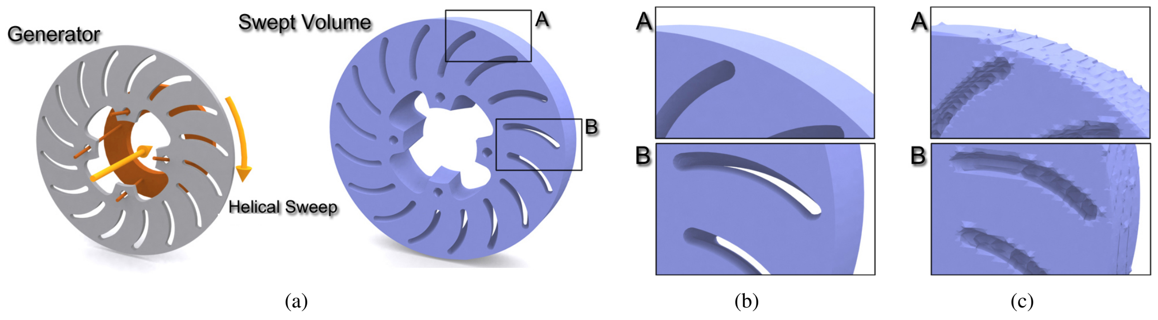


Figure 1: **Sweeping a Rotor. (a)** a rotor blade in gray (generator, 4736 tris) under helical sweep to be docked into another mechanical part in dark orange (left) and its $\mathbb{SV}$ (806K tris) using our new $\mathbb{SV}$ approximation using adaptive sampling (right); **(b)** Zoomed-in views of our $\mathbb{SV}$ approximation (1.7M samples); **(c)** Zoomed-in views of $\mathbb{SV}$ approximation using Kim et al. [2003] and uniform sampling (2M samples). Notice the aliasing effect along the sharp edge (region A) and the missing holes (region B) in (c) compared to those in (b), even though the number of used samples in (c) is higher than in (b).

## Abstract

We present a simple algorithm to generate a topology-preserving, error-bounded approximation of the outer boundary of the volume swept by a polyhedron along a parametric trajectory. Our approach uses a volumetric method that generates an adaptive volumetric grid, computes signed distance on the grid points, and extracts an isosurface from the distance field. In order to guarantee geometric and topological bounds, we present a novel sampling and front propagation algorithm for adaptive grid generation. We highlight the performance of our algorithm on many complex benchmarks that arise in geometric and solid modeling, motion planning and CNC milling applications. To the best of our knowledge, this is the first practical algorithm that can generate swept volume approximations with geometric and topological guarantees on complex polyhedral models swept along any parametric trajectory.

## 1 Introduction

The problem of computing the swept volume ($\mathbb{SV}$) of an object swept along a smooth trajectory frequently arises in geometric and solid modeling, robot motion planning and collision detection. The object may correspond to a solid or a collection of boundary surfaces and the goal is to compute an accurate representation of the *outer boundary* swept by these primitives along a parametric trajectory. The area of $\mathbb{SV}$ computation has been extensively studied in the literature and the mathematical formulation of $\mathbb{SV}$ computation is well-understood [Abdel-Malek et al. 2004].

The problem of exactly computing the outer boundary of the $\mathbb{SV}$ has high combinatorial and algebraic complexity. As a result, most prior algorithms have focused on computing an approximation of the $\mathbb{SV}$. Different approximation algorithms can be characterized based on whether they are limited to 2D objects, or perform an image-space projection or visualization of the $\mathbb{SV}$s, or use relatively coarse discretization of the boundary primitives. These algorithms are either slow for practical applications, or suffer from accuracy problems, or may compute a rather coarse approximation of the $\mathbb{SV}$.

A key issue in approximating the boundary of the $\mathbb{SV}$ is providing tight geometric and topological guarantees. Specifically, it is important that the approximate $\mathbb{SV}$ should have the same genus and no additional holes or handles are created due to the approximation. For example, $\mathbb{SV}$s are frequently used to perform continuous collision detection and the presence of extra holes can result in missed collisions. Furthermore, it is important to faithfully reconstruct the sharp features on the boundary of $\mathbb{SV}$ for CAD/CAM applications.

## 2 Related Work

### 2.1 Swept Volume Computation

Many researchers have formulated the mathematical formulation of $\mathbb{SV}$ computation using Singularity theory [Abdel-Malek and Othman 1999; Abdel-Malek and Yeh 1997], Sweep differential equation [Blackmore et al. 1997; Blackmore and Leu 1990; Wang et al. 2000], Minkowski sums [Elber and Kim 1999], Envelope theory [Martin and Stephenson 1990; Weld and Leu 1990; Peternell et al. 2005], implicit modeling [Schroeder et al. 1994], and kinematics [Jüttler and Wagner 1996]. In particular, Weld and Leu [Weld and Leu 1990] presented a geometric representation of $\mathbb{SV}$ for compact $n$-manifolds with an application to polyhedral objects. We refer the reader to [Abdel-Malek et al. 2004] for an extensive survey.

Due to the high complexity of computing the exact $\mathbb{SV}$, many polyhedral approximation algorithms have been proposed. These include approximate 2D sweep [Lee et al. 2002; Ahn et al. 1993], isosurface reconstruction-based sweep [Schroeder et al. 1994; Kim et al. 2003; Himmelstein et al. 2007; Xu et al. 2007], and coarse approximation of the arrangement of swept polyhedral surfaces [Abrams and Allen 1995; Raab 1999; Rossignac et al. 2007; Erdim and Ilieş 2008]. However, these algorithms may not be able to reconstruct the final $\mathbb{SV}$ in a geometrically- and topologically-correct manner or may require too many voxel grids (with high memory overhead) for accurate reconstruction.

## 2.2 Isosurface Reconstruction

Grid-based isosurface reconstruction has been extensively studied, starting from the seminal Marching Cubes algorithm [Lorensen and Cline 1987]. Many extensions and variants of the basic algorithm have also been proposed, including Enhanced Marching Cubes [Kobbelt et al. 2001], dual contouring [Ju et al. 2002], dual marching cubes [Nielson 2004; Schaefer and Warren 2004]. Wood et al. [2000] use surface wavefront propagation techniques to extract semi-regular meshes from volumes.

In order to address the problem of reconstructing a topologically reliable isosurface, Chernyaev presents a topological validation algorithm of Marching Cubes [1995]. Varadhan et al. [2004] propose a simple conservative test in terms of sampling and reconstructing geometrically- and topologically-correct isosurface, though it is limited to closed primitives. Zhang et al. [2004] present a topology-preserving approach for isosurface simplification using an enhanced cell representation. Boissonnat et al. [2004] propose a criterion to guarantee an isotopic mesh reconstruction to polygonize an implicit surface. Bremer et al. [2004] present a hierarchical data structure on functions defined over a 2D domain and extract a topology-valid approximation with multi-resolution support. Other surface reconstruction methods that provide topological guarantees for known topology types include [Mangin et al. 1995; Aktouf et al. 1996; Bischoff and Kobbelt 2002]. For non-genus-zero surface reconstruction, Sharf et al. [2007] present a user-guided, topology-aware reconstruction algorithm for scan data and Paiva et al. [2006] propose an Octree subdivision method to adapt to the topology of implicit surfaces. However, none of these approaches can reconstruct an arrangement of *open, complex surfaces* with *geometric and topological guarantees* on the approximation, so that they are not directly applicable to computing $\mathbb{SV}$.

## 3 Overview

In this section, we formulate the problem of computing the swept volume, $\mathbb{SV}$, of a closed, non-convex polyhedral model mathematically. We also describe some of the issues in approximate computation of the swept volume. Finally, we give an overview of our topology-preserving algorithm.

### 3.1 Mathematical Formulation

Let $\mathbf{\Gamma}$, also known as a generator, be a closed, non-convex polyhedron in $\mathbb{R}^3$. Let the sweep trajectory $\tau(t)$ be a tuple of ($\mathbf{\Psi}(t)$, $\mathbf{R}(t)$), where $\mathbf{\Psi}(t)$ is a time-varying, differentiable point in $\mathbb{R}^3$ and $\mathbf{R}(t)$ is a time-varying matrix in $SO(3)$. Here, both $\mathbf{\Psi}(t)$ and $\mathbf{R}(t)$ depend on a single variable, the time $t \in [0, 1]$. Furthermore, $\mathbf{\Psi}(0)$ corresponds to the origin, and $\mathbf{R}(0)$ is the identity matrix. Consider the following sweep equation of $\mathbf{\Gamma}(t)$:

$$\mathbf{\Gamma}(t) = \mathbf{\Psi}(t) + \mathbf{R}(t)\mathbf{\Gamma} \tag{1}$$

In the rest of the paper, the $\mathbb{SV}$ of the generator $\mathbf{\Gamma}$ along the trajectory $\tau(\mathbf{t})$ is defined as:

$$\mathbb{SV}(\mathbf{\Gamma}) = \bigcup_{t\in[0,1]} \mathbf{\Gamma}(t) \tag{2}$$

Our goal is to compute $\mathcal{S}$, the outer boundary of $\mathbb{SV}(\mathbf{\Gamma})$. It corresponds to computing the outer boundary[1] of the arrangement induced by the surface elements in $\mathbb{SV}(\mathbf{\Gamma})$. In other words, we do not compute the internal voids in the swept volume or give any guarantees related to them. Our primary focus is reliably computing the outer boundary.

Weld and Leu [1990] prove that the boundary of $\mathbb{SV}$ is obtained by computing the $\mathbb{SV}$ of individual faces in $\mathbf{\Gamma}$, and their union. Moreover, they also prove that, besides the trivial surfaces of $\mathbf{\Gamma}$ at initial and final positions during sweep (i.e., $\mathbf{\Gamma}(0)$ and $\mathbf{\Gamma}(1)$ in Eq. 1), there are only two types of surfaces that belong to the $\mathbb{SV}$ of individual faces (or individual polygons): ruled and developable surfaces. Thus, computing the exact $\mathbb{SV}$ boils down to explicitly computing ruled and developable surface primitives, and finally computing their arrangement.

### 3.2 Our Approach

Our approach builds upon the prior formulation by Kim et al. [Kim et al. 2003]. However, in [Kim et al. 2003] no guarantees are given on the topological accuracy of the iso-surface or whether it can indeed reconstruct the sharp features.

Instead of a uniform 3D grid, we use novel adaptive grid subdivision and front propagation algorithms that are based on three criteria: geometric deviation error, *extended complex-cell* tests and *extended star-shaped* tests. Intuitively, the complex cell criterion ensures that the surface intersects the grid cell in a simple manner in most cases. The star-shaped test combined with the complex cell test ensures that the surface inside each grid cell is homeomorphic to a topological disk [Varadhan et al. 2004]. We use Marching Cubes or any of its variants to extract the isosurface from the resulting grid. The extracted surface is an isotopic approximation of the given surfaces $\mathcal{E}$. We extend these two fundamental geometric tests from [Varadhan et al. 2004] to apply to open surface primitives such as ruled or developable surfaces. These two tests are described below.

**Extended Complex Cell Test** A cell is *complex* if it has a *complex voxel*, *face*, *edge*, or an *ambiguous sign configuration* [Varadhan et al. 2004; Varadhan and Manocha 2006]. We define a voxel (face) of a grid cell to be *complex* if it intersects $\mathcal{E}$ and the grid vertices belonging to the voxel (face) do not exhibit a sign change. The sign of a vertex is positive if it lies within $\mathcal{E}$, negative otherwise. An edge of the grid cell is said to be *complex* if $\mathcal{E}$ intersects the edge more than once. In order to perform the complex cell test on a given cell, we require signs at the vertices of the cell and an intersection test for the voxel/faces/edges of the cell. In Sec. 4.3, we show how to compute the signs at the cell vertices using front propagation.

**Extended Star-shaped Test** This test ensures that the surface $\mathcal{E}$ restricted to a cell is *star-shaped* within that cell. Let $S$ be a nonempty subset of $\mathbb{R}^d$. The set Kernel$(S)$ consists of all $\mathbf{s} \in S$ such that for any $\mathbf{x} \in S$, we have $\mathbf{s} + \lambda(\mathbf{x} - \mathbf{s}) \in S, \forall\lambda \in [0, 1]$. $S$ is *star-shaped* if Kernel$(S) \neq \emptyset$. Intuitively, a star-shaped primitive has a representative point (called the origin) such that all the points in the primitive are visible from the origin. The star-shaped test is applicable to the case where the surface $\mathcal{E}$ is defined over a set of primitives $\{P_i\}$. We can check whether $\mathcal{E}$ is star-shaped by testing whether there exists a point $\mathbf{o}$ such that $P_i$ is star-shaped w.r.t. $\mathbf{o}$ for $\forall\ i$. This is a conservative condition for star-shapedness of $\mathcal{E}$. Formally, we test if $\cap_i Kernel(P_i) \neq \emptyset$.

[1]Throughout the paper, we interchangeably use the outer boundary of the arrangement and the outer boundary of $\mathbb{SV}(\mathbf{\Gamma})$ to describe $\mathcal{S}$.

As mentioned earlier, in case of $\mathbb{SV}$ computation, the surface primitives (ruled and developable surfaces) are not closed. We assume that these primitive surfaces are orientable. As a result, the kernel of a surface primitive can have two separate components, one for each orientation of the surface (see Fig. 2). We take this fact into account while performing the star-shaped test for $\mathcal{S}$. When we compute $\cap_i Kernel(P_i)$ where $\{P_i\}$ is the set of primitives contributing to $\mathcal{S}$, it is possible that we may obtain two disjoint regions. One of them will lie inside and the other will lie outside $\mathcal{S}$ (see Fig. 2). As a result, the star-shapedness of $\mathcal{S}$ can be defined in two ways: one from inside and the other from outside. We require that $\mathcal{S}$ is star-shaped from outside. We use this property later during front propagation (in Sec. 4.3). We test if $\mathcal{S}$ is star-shaped from the outside by checking if the kernel component that lies outside is non-empty.

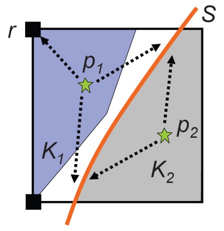


Figure 2: **Extended Star-shaped Test for an Open Surface**: an open surface $S$ may have two kernels, $K_1, K_2$. Since the origin $p_1$ of $K_1$ can see a front point $r$, $K_1$ is outside while $K_2$ is inside. Thus, $S$ is star-shaped from outside.

Let $K_1$ and $K_2$ be the two kernel components. To determine which of them lies outside, we require computing a *reference point* ($\mathbf{r}$). We choose two points $\mathbf{p_i} \in K_i, i = 1, 2$ and check whether $\mathbf{p_i}$ can see the reference point $\mathbf{r}$. Only one of them can see the reference point and the corresponding kernel component lies outside.

During adaptive grid generation, we perform the star-shaped test on different grid cells. Each time we perform the star-shaped test on a cell, we need to compute a reference point in that cell. In Sec. 4.3, we present an algorithm to compute this reference point using front propagation.

**Time Complexity Analysis** Assume that there are $N$ tessellated surface primitives intersected with a cell. The extended complex cell test requires two types of sub-tests: distance field computation and sign query. Determining the sign of each vertex in a cell takes $O(N)$ time, and the distance field computation requires $O(N)$ time using the max-norm distance computation. For a polyhedral primitive, the extended star-shaped test reduces to linear programming [Varadhan et al. 2004; Varadhan and Manocha 2006]. Thus, this step takes $O(N)$ expected time.

## 4 Adaptive Sampling and Front Propagation

In this section, we present our swept volume approximation algorithm based on the approach outlined in Sec. 3.2. Our novel algorithm consists of two steps: (1) adaptive sampling assisted by front propagation and (2) reconstruction from the distance field.

### 4.1 Notation

We first introduce some notations. As before, the exact swept volume boundary is denoted as $\mathcal{S}$. Our swept volume approximation is denoted as $\mathcal{A}$. $\mathcal{F}$ denotes the outermost cells in the arrangement of the surfaces enumerated in Sec. 3.1 that contribute to $\mathcal{S}$. We call $\mathcal{F}$ the *free space*. $\mathcal{S}$ corresponds to the boundary of $\mathcal{F}$. A point in $\mathcal{F}$ is referred to as a *free point*. We call a grid cell that contains a point in $\mathcal{F}$ a *free cell*. Similarly, a grid vertex that belongs to $\mathcal{F}$ is referred to as a *free vertex*. The letter $C$ denotes a grid cell used for sampling. A grid cell consists of a cube-shaped voxel, six faces, twelve edges, and eight vertices. In our case, $C$ is a closed set.

### 4.2 Adaptive Grid Generation

We generate an adaptive volumetric grid such that every grid cell satisfies the star-shaped and complex cell tests. The basic approach is to start with a single cell that bounds $\mathcal{S}$ and apply the extended star-shaped and complex cell tests recursively. If a grid cell does not satisfy either of the two tests, the grid cell is subdivided and the algorithm is applied to the children cells.

Our goal is to reconstruct just the outer boundary of the swept volume. Therefore it suffices to consider only the free cells during adaptive grid generation. The basic steps of the algorithm are:

1. Initialize the grid to be an axis-aligned cell that bounds $\mathcal{S}$.
2. Detect free cells in the grid.
3. For each unmarked free cell $C$ do
   - Mark $C$
   - Check whether $C$ satisfies the star-shaped and complex cell tests.
   - If either of the two tests fails, subdivide $C$.
4. If any grid cell has been subdivided in Step 3, jump to Step 2.

In order to apply the above algorithm, we need to identify the free cells in the grid. We use front propagation for that computation.

### 4.3 Front Propagation

Front propagation is a technique that is conceptually similar to propagating a wavefront through a medium in space, i.e. such as level set computation [Sethian 1996]. In our context, the medium is the free space $\mathcal{F}$. The front is initialized to a free point and allowed to propagate in all directions. Front propagation stops only when the front arrives at the swept volume boundary. As a result, the front visits all the free points.

We use a discrete representation for the front. The front is a finite set of free points(e.g. the black solid boxes shown in Fig. 3). Let $\Omega$ denote the front. We refer to a point in $\Omega$ as a *front point*. We refer to a cell that has been visited by the front as a *front cell*. Given a cell $C$, the *front set* $\Omega_C$ of cell $C$ refers to the set of front points contained in cell $C$, i.e., $\Omega_C = \Omega \cap C$. The front points in $\Omega_C$ need not necessarily be the vertices of $C$.

We use front propagation to detect the free cells during adaptive grid generation. We initialize $\Omega$ to be a free vertex of the initial grid (an axis aligned bounding box that encloses $\mathcal{S}$). Once the front is initialized, we allow it to propagate through the grid. A front can propagate in three ways, including propagation within a grid cell (Cell Propagation), from a grid cell to its neighboring cells (Vertex Propagation), or from a grid cell to its children cells when the cell is subdivided (Downward Propagation). In this manner, the front visits all the free cells in the grid.

We keep track of free cells in a queue ($Q$). When the front encounters a free cell, it is added to $Q$. The grid generation algorithm processes a free cell by extracting it from $Q$ and checking whether a given cell satisfies the complex cell and star-shaped tests. If either of the tests fails, the grid generation algorithm subdivides the cell. The front continues to propagate on the refined grid and visits the children of the free cells.

We now elaborate each of these three types of front propagation. These include:

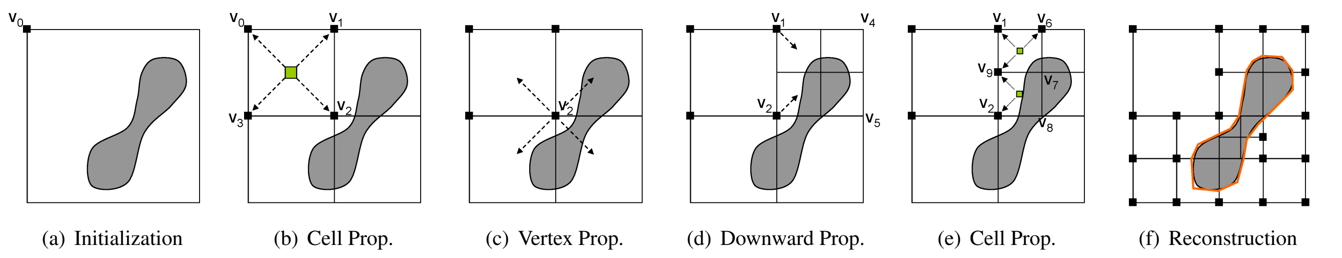


Figure 3: **Front Propagation**: The front is initialized to be a vertex (point $\mathbf{v}_0$) of an axis aligned bounding box that encloses $\mathcal{S}$ (Fig. (a)). Figs. (b) and (c) illustrate cell and vertex propagation respectively. In Fig. (b), the front propagates within a cell to all its free vertices (points $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$). Fig. (c) shows front propagation from a vertex (point $\mathbf{v}_2$) to all its neighboring cells. Fig. (d) illustrates downward propagation. The front propagates from cell $\mathbf{v}_1\mathbf{v}_4\mathbf{v}_5\mathbf{v}_2$ to its children cells. The front propagates to only those children cells that have a front point. Fig. (e) shows cell propagation in cells $\mathbf{v}_1\mathbf{v}_6\mathbf{v}_7\mathbf{v}_9$ and $\mathbf{v}_9\mathbf{v}_7\mathbf{v}_8\mathbf{v}_2$. Fig. (f) shows the final adaptive grid. The front has visited all the free cells i.e., cells that contain a point in $\mathcal{F}$. An isosurface is extracted from this grid to obtain a topologically correct approximation to the outer boundary of swept volume (shown in orange).

- **Vertex Propagation**: Vertex propagation refers to propagation of the front from a free vertex to all the neighboring cells that contain the vertex. If a vertex $\mathbf{v}$ of the grid is a front point then the front propagates to every neighboring grid cell that contains $\mathbf{v}$. If a cell $C$ contains $\mathbf{v}$ then $C$ is classified as a front cell and we update the front, i.e., $\Omega_C = \Omega_C \cup \{\mathbf{v}\}$. If $C$ is not already a front cell then it is yet to be processed. We add $C$ to $Q$ and thereby ensure that it is eventually processed. Fig. 3(c) illustrates vertex propagation.
- **Cell Propagation**: Cell propagation refers to propagation of the front within a cell. We determine the cell vertices that are in $\mathcal{F}$ and add them to the front. We perform cell propagation only on the cells that satisfy the star-shaped test. We use the star-shaped test to detect free vertices in the cell. By definition of star-shapedness, given a cell $C$ that satisfies the star-shaped property, there exists a point $\mathbf{o} \in C$ (the origin) that can see every free point in $C$. In particular, all the free vertices of $C$ are visible from $\mathbf{o}$. We test if a cell vertex $\mathbf{v}$ is a free vertex by testing if the distance to the nearest surface primitives along the direction of $\overrightarrow{\mathbf{ov}}$ is greater than the magnitude of $|\overrightarrow{\mathbf{ov}}|$. If $\mathbf{v}$ satisfies this test, then the front propagates to $\mathbf{v}$, i.e., $\Omega_C = \Omega_C \cup \{\mathbf{v}\}$. Figs. 3(b) and 3(e) illustrate cell propagation.
- **Downward Propagation**: If a cell $C$ does not satisfy either the star-shaped or the complex cell test then we subdivide $C$ into children cells $C_i$. The front propagates from $C$ into its children $C_i$. We refer to this type of propagation as downward propagation. The front propagates from $C$ to only those children cells that contain a front point, i.e., front propagates to $C_i$ only if $\Omega_{C_i} = C_i \cap \Omega \neq \emptyset$. Fig. 3(d) illustrates downward propagation.

In this manner, front propagation detects the free cells in the grid. During grid generation, the front may not reach or propagate through all the free cells in the grid. This is due to the presence of intermediate cells that do not satisfy the complex cell and star-shaped tests. These intermediate cells act as a barrier and may prevent the front from propagating to some of the free cells. However, as the grid is refined, eventually we will obtain a final adaptive volumetric grid $\mathcal{G}$ such that all the free cells in $\mathcal{G}$ satisfy both complex cell and star-shaped tests.

As we mentioned in Sec. 3.2, the star-shaped test requires a reference point that serves as the kernel. We perform the star-shaped test on a cell $C$ only after it has been visited by the front. This implies that its front is nonempty, i.e. $\Omega_C \neq \emptyset$. We choose one of the front points as a reference point for the star-shaped test. In order to perform the complex cell test, we need to know the signs of the cell vertices, i.e. whether or not they are free vertices. The complex cell test is performed on a cell $C$ only after we perform cell propagation on $C$. As a result, a vertex $\mathbf{v}$ of $C$ is a free vertex if and only if it is a point on the front.

### 4.4 Distance Fields and Reconstruction

All the free cells in the final adaptive volumetric grid $\mathcal{G}$ satisfy the complex cell and star-shaped tests. Moreover, for each grid point in $\mathcal{G}$, we compute signed, *directional* distance along six axis-aligned directions using a variant of max norm computation [Varadhan et al. 2004]. We determine the signs of grid points during front propagation. Finally, we apply Marching Cubes or its variant such as dual contouring [Ju et al. 2002] to $\mathcal{G}$ to obtain a topology-preserving approximation $\mathcal{A}$ of the exact swept volume boundary $\mathcal{S}$. Dual contouring also preserves sharp features in the $\mathbb{SV}$ boundary.

## 5 Analysis

In this section, we show how our algorithm can guarantee geometric and topological error bounds.

### 5.1 Topological Guarantees

We can prove that the front visits every free cell in the final adaptive volumetric grid generated by our algorithm. Moreover, it can be also shown that a cell in final adaptive volumetric grid is a free cell if and only if it is a front cell. Thus, since all the free cells satisfy the complex cell test and star-shaped tests, the approximated $\mathbb{SV}$ has the same topology as the exact $\mathbb{SV}$. More details on this proof is given in [Zhang et al. 2009].

### 5.2 Geometric Guarantees

We can extend our swept volume approximation algorithm to obtain arbitrarily tight two-sided Hausdorff error bounds on our swept volume approximation. Let $\beta$ denote the set of swept surfaces enumerated in Sec. 3.1 that contribute to the boundary of the swept volume. We make use of the fact that the swept volume boundary $\mathcal{S}$ is a subset of the boundaries of the swept surfaces in $\beta$. Given an error tolerance $\epsilon > 0$, we augment the grid generation algorithm with an additional criterion. Consider a grid cell $C$. Let $\beta_C$ denote the set of swept surfaces that intersect $C$. Let $\mathcal{A}_C$ denote the output of isosurface extraction applied to cell $C$. We subdivide a cell if the Hausdorff distance between $\mathcal{A}_C$ and any swept surface in $\beta_C$ is greater than $\epsilon$. More precisely, we calculate $d = \max\{\text{Hausdorff Distance}(\mathcal{A}_C, \beta_i) \mid \beta_i \in \beta_C\}$ and subdivide $C$ if $d > \epsilon$. This ensures that the two-sided Hausdorff distance between our approximation $\mathcal{A}$ and the exact swept volume $\mathcal{S}$ is less than $\epsilon$.

## 6 Implementation and Results

In this section, we describe the implementation of our SV computation algorithm and highlight its performance on different benchmarks. We have implemented our $\mathbb{SV}$ algorithm using C++ language under Windows XP. We use the public domain computational geometry library, CGAL[2], to maintain the mesh data structures for swept volume approximation. We also use the public-domain linear programming solver, QSopt[3], to find a kernel point in the extended star-shaped test. We have applied our new $\mathbb{SV}$ algorithm to the following benchmarks:

**Solid Modeling** (Fig. 4): a sphere under hypotrochoid sweep (left), a torus under helical sweep (middle) and a torusknot under helical sweep(right). For these models, we also show a green rod that indicates the translational direction of the helical motion.

**Robot Motion Planning** (Fig. 5): The $\mathbb{SV}$ of a Puma robot is generated as a result of motion planning. One may perform collision detection of the $\mathbb{SV}$ against an obstacle (the green car) to validate the correctness of motion planning, i.e. whether the computed path is collision-free.

**CNC Milling** (Fig. 6): A milling tool (modeled as a cylinder with a flat end) is driven by G-code[4] instructions to cut the underlying workpiece. The final workpiece is obtained by performing a Boolean operation between the initial workpiece and the $\mathbb{SV}$ of the milling tool. The driving G-code consists of more than 4.3K instructions, including linear and circular interpolations.

In Table 1, we show the geometric complexities of the benchmarking models as well the runtime performance statistics of our $\mathbb{SV}$ algorithm on these models. The timing was measured on a PC with dual AMD Athlon 64 FX-60 processor and 2G of memory. We have observed that our algorithm generates 20$\sim$70 times fewer grid cells than [Kim et al. 2003] while guaranteeing the topology and error-bounds on the result.

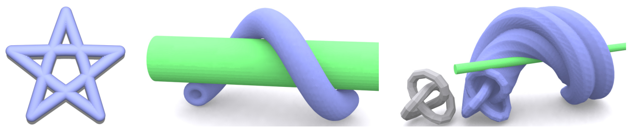


Figure 4: **Hypotrochoid Sweep of a Sphere.**

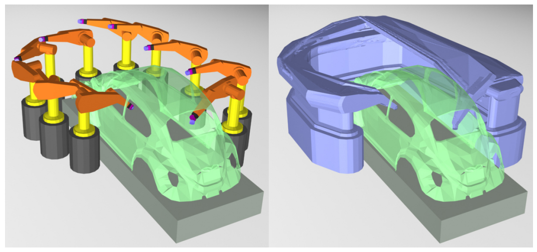
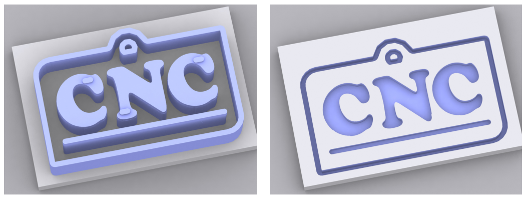


Figure 5: **Sweeping a Puma Robot.**

**Comparisons** In contrast to the earlier work on $\mathbb{SV}$ including the work [Kim et al. 2003] and [Varadhan et al. 2004], our new algorithm provides the following benefits (also see Fig.1). Our new sampling method provides a bounded-error approximation with the same topology as the exact $\mathbb{SV}$. However, [Kim et al. 2003] does not provide any guarantees on the final topology or the sharp features of the swept volume. Since our algorithm uses an adaptive grid for sampling, the memory requirement for our new algorithm is considerably lower than that of [Kim et al. 2003], while computing a more accurate representation of the boundary. Moreover,[Kim et al. 2003] uses a rather adhoc and overly conservative method for generating a uniform grid. Since the field grid is uniform, it may require high memory overhead. Our novel, front marching algorithm efficiently works on adaptive grids in terms of computing the signed distance field for an arrangement of open surface primitives. The earlier method by Varadhan et al. [Varadhan et al. 2004] is only able to handle primitives corresponding to closed surfaces (or solids) and their Boolean combination. Their results and formulation are not applicable to envelopes of surfaces, as the surface primitives are not closed. In fact, our proof for open surface primitives is very different from that used in [Varadhan et al. 2004], which is limited to Boolean combinations of closed primitives. Finally, our sign computation algorithm is also quite different from [Varadhan et al. 2004], since their algorithm assumes closed surface (or solids) as input and thus signs can be easily determined from this assumption.

[2]http://www.cgal.org/
[3]http://www2.isye.gatech.edu/~wcook/qsopt/
[4]A programming language that controls CNC machine tools.



(a) Swept Volume (b) Final Workpiece

Figure 6: **CNC Milling.**

**Limitations** Our $\mathbb{SV}$ algorithm may not be able to handle all degenerate inputs such as tangential contact between surface primitives or arbitrarily close surface primitives. In fact, these problems are inherent to the underlying sampling and reconstruction algorithm [Varadhan et al. 2004] that our $\mathbb{SV}$ approximation algorithm is also based on. For these degenerate inputs, the algorithm may require an excessive amount of subdivisions to accurately approximate the outer boundary of $\mathbb{SV}$. In practice, this behavior may be controllable at the expense of the accuracy of the result. Our subdivision scheme can be conservative and can generate more cells than are necessary. Finally, our approach is limited to only the outer boundary computation and not the internal voids.

## 7 Conclusions and Future Work

We have presented a simple algorithm to generate a error-bounded, topology-preserving approximation of the outer boundary of swept volume of a polyhedron swept along a given trajectory.

There are many avenues for future work. Swept volume has many applications as pointed out by [Abdel-Malek et al. 2004]. We would like to further explore how to utilize our $\mathbb{SV}$ approximation in these applications, e.g. four dimensional collision detection and shape modeling. In our current implementation, we do not perform any extra pre-processing; however, some preprocessing can accelerate the entire computation. For example, to find surface primitives that intersect with a cell, currently we perform a simple linear search and do not implement any sophisticated optimizations such as range query data structures. As a result, the runtime performance of our algorithm can be considerably improved. Finally, we are also interested in approximating the envelope of curved surfaces for other applications such as Minkowski sums and developing practical algorithms for a wide class of arrangement problems that arise in various geometric applications.

| Model | Combinatorial Complexity | | | Computational Performance (seconds) | | | | Grid Resolution(# of cells) | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mathbf{\Gamma}$ | # of Tris | $\partial SV(\mathbf{\Gamma})$ | Surf Gen | Sampling | Iso-Surf | Total | Ours | [Kim et al. 2003] |
| Sphere | 840 | 294K | 143K | 0.49 | 1.4K | 2.56 | 1.4K | 0.363M | 16.7M |
| Torus | 576 | 59K | 142K | 0.28 | 366 | 1.06 | 367 | 0.336M | 16.7M |
| Torus-Deform | 576 | 59K | 142K | 0.28 | 373 | 0.73 | 374 | 0.205M | 16.7M |
| Torusknot | 640 | 59K | 366K | 0.32 | 897 | 2.04 | 899 | 0.845M | 16.7M |
| Rotor | 4736 | 27K | 806K | 0.20 | 1.6K | 3.45 | 1.6K | 1.72M | 134M |
| PUMA | 708 | 48K | 324K | 0.30 | 874 | 2.18 | 876 | 0.774M | 16.7M |
| Milling | 120 | 1.32M | 191K | 0.83 | 856 | 5.96 | 862 | 0.774M | 16.7M |

Table 1: **Model Complexities of SV Benchmarks and Timing Statistics.** From the second to the fourth column, each column respectively shows the triangle count of a generator $\mathbf{\Gamma}$, the total number of triangles in the tessellated ruled and developable surfaces, and the triangle count of the boundary of swept volume computed by our algorithm $\partial SV(\mathbf{\Gamma})$. From the fifth to the seventh column, each column respectively shows a breakdown of the timing for surface primitive generation, adaptive sampling and isosurface extraction. The eighth column is the total time. The ninth and tenth columns list the grid resolutions required by our algorithm and by Kim et al. [2003] to achieve similar results.

## Acknowledgements

This research work was supported in part by the KRF Grant funded by the Korean Government (KRF-2007-331-D00400) and the IT R&D program of MKE/MCST/KOCCA (2008-F-033-02, Development of Real-time Physics Simulation Engine for e-Entertainment). Dinesh Manocha was supported in part by NSF, ARO, Intel and RDECOM. We would also like to thank Gokul Varadhan for helping us at the initial stage of this project.

## References

ABDEL-MALEK, K., AND OTHMAN, S. 1999. Multiple sweeping using the Denavit-Hartenber representation method. *Computer-Aided Design 31*, 567–583.

ABDEL-MALEK, K., AND YEH, H. J. 1997. On the determination of starting points for parametric surface intersections. *Computer-Aided Design 29*, 1, 21–35.

ABDEL-MALEK, K., BLACKMORE, D., AND JOY, K. 2004. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling 23*, 5, 1–25.

ABRAMS, S., AND ALLEN, P. 1995. Swept volumes and their use in viewpoint computation in robot work-cells. In *Proc. IEEE International Symposium on Assembly and Task Planning*, 188–193.

AHN, J.-W., KIM, M.-S., AND LIM, S.-B. 1993. Approximate general sweep boundary of a 2D curved objects. *Graphical Models and Image Processing 55*, 2.

AKTOUF, Z., BERTRAND, G., AND PERROTON, L. 1996. A three-dimensional holes closing algorithm. In *Proceedings of the 6th International Workshop on Discrete Geometry for Computer Imagery*, Springer-Verlag, London, UK, 36–47.

BISCHOFF, S., AND KOBBELT, L. P. 2002. Isosurface reconstruction with topology control. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 246.

BLACKMORE, D., AND LEU, M. C. 1990. A differential equation approach to swept volumes. In *Proc. of Rensselaer's 2nd International Conference on Computer Integrated Manufacturing*, 143–149.

BLACKMORE, D., LEU, M., AND WANG, L. 1997. Sweep-envelope differential equation algorithm and its application to NC machining verification. *Computer-Aided Design 29*, 629–637.

BOISSONNAT, J.-D., COHEN-STEINER, D., AND VEGTER, G. 2004. Isotopic implicit surface meshing. In *ACM Symposium on Theory of Computing*.

BREMER, P.-T., EDELSBRUNNER, H., HAMANN, B., AND PASCUCCI, V. 2004. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics 10*, 4.

CHERNYAEV, E. 1995. Marching cubes 33: Construction of topologically correct isosurfaces. *Institute for High Energy Physics, Russia, Report CN/95-17*.

ELBER, G., AND KIM, M.-S. 1999. Offsets, sweeps, and Minkowski sums. *Computer-Aided Design 31*, 3, 163.

ERDIM, H., AND ILIEŞ, H. T. 2008. Classifying points for sweeping solids. *Computer-Aided Design 40*, 9, 987–998.

HIMMELSTEIN, J. C., FERRE, E., AND LAUMOND, J.-P. 2007. Swept volume approximation of polygon soups. In *IEEE ICRA*.

JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual contouring of hermite data. In *SIGGRAPH 2002, Computer Graphics Proceedings*.

JÜTTLER, B., AND WAGNER, M. 1996. Spatial rational B-spline motions. *ASME Journal of Mechanical Design 118*, 193–201.

KIM, Y., VARADHAN, G., LIN, M., AND MANOCHA, D. 2003. Fast swept volume approximation of complex polyhedral models. *Proc. of ACM Symposium on Solid Modeling and Applications*, 11–22.

KOBBELT, L., BOTSCH, M., SCHWANECKE, U., AND SEIDEL, H. P. 2001. Feature-sensitive surface extraction from volume data. In *ACM SIGGRAPH*, 57–66.

LEE, J., HONG, S., AND KIM, M. 2002. Polygonal boundary approximation for a 2D general sweep based on envelope and Boolean operations. *Visual Computer 16*.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol. 21, 163–169.

MANGIN, J.-F., FROUIN, V., BLOCH, I., RÉGIS, J., AND LÓPEZ-KRAHE, J. 1995. From 3D magnetic resonance images to structural representations of the cortex topography using topology preserving deformations. *Journal of Mathematical Imaging and Vision 5*, 4, 297–318.

MARTIN, R., AND STEPHENSON, P. 1990. Sweeping of three-dimensional objects. *Computer-Aided Design 22*, 4.

NIELSON, G. 2004. Dual marching cubes. In *IEEE Visualization Conference*.

PAIVA, A., LOPES, H., LEWINER, T., AND DE FIGUEIREDO, L. 2006. Robust adaptive meshes for implicit surfaces. *19th Brazilian Symposium on Computer Graphics and Image Processing*, 205–212.

PETERNELL, M., POTTMANN, H., STEINER, T., AND ZHAO, H. 2005. Swept volumes. *Computer Aided Design and Applications 2*, 5, 599–608.

RAAB, S. 1999. Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes. In *Proc. 15th ACM Symposium on Computational Geometry*, 163–172.

ROSSIGNAC, J., KIM, J. J., SONG, S. C., SUH, K. C., AND JOUNG, C. B. 2007. Boundary of the volume swept by a free-form solid in screw motion. *Computer-Aided Design 39*, 9, 745–755.

SCHAEFER, S., AND WARREN, J. 2004. Dual marching cubes: Primal contouring of dual grids. In *Pacific Graphics*.

SCHROEDER, W., LORENSEN, W., AND LINTHICUM, S. 1994. Implicit modeling of swept surfaces and volumes. In *IEEE Visualization Conference*.

SETHIAN, J. 1996. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci.*, vol. 93, 1591–1595.

SHARF, A., LEWINER, T., SHKLARSKI, G., TOLEDO, S., AND COHEN-OR, D. 2007. Interactive topology-aware surface reconstruction. *ACM Trans. Graph. 26*, 3, 43.

VARADHAN, G., AND MANOCHA, D. 2006. Accurate Minkowski sum approximation of polyhedral models. *Graphical Models 68*, 4, 343–355.

VARADHAN, G., KRISHNAN, S., SRIRAM, T. V. N., AND MANOCHA, D. 2004. Topology preserving surface extraction using adaptive subdivision. In *Eurographics Symposium on Geometry Processing*.

WANG, G., SUN, J., AND HUA, X. 2000. The sweep-envelope differrential equation algorithm for general deformed swept volumes. *Computer Aided Geometric Design 17*, 5, 399–418.

WELD, J., AND LEU, M. 1990. Geometric representation of swept volume with application to polyhedral objects. *Int. J. of Robotics Research 9*, 5.

WOOD, Z., DESBRUN, M., SCHROEDER, P., AND BREEN, D. 2000. Semi-regular mesh extraction from volumes. In *IEEE Visualization 2000 Proceedings*.

XU, Z., CHEN, Z., YE, X., AND ZHANG, S. 2007. Approximate the swept volume of revolutions along curved trajectories. In *ACM Symposium on Solid and Physical Modeling*.

ZHANG, N., HONG, W., AND KAUFMAN, A. 2004. Dual contouring with topology-preserving simplification using enhanced cell representation. In *IEEE Visualization*.

ZHANG, X. Y., KIM, Y. J., AND MANOCHA, D. 2009. Analysis and implementation of reliable sweeps. Tech. rep., Ewha Womans University, Korea.