Precise Algebraic-based Swept Volumes for Arbitrary Free-form Shaped Tools towards Multi-axis CNC Machining Verification

Jinesh Machchhar[a], Denys Plakhotnik[b], Gershon Elber[a]

[a]*Department of Computer Science, Technion Israel Institute of Technology, Israel*
[b]*ModuleWorks GmbH, Germany*

Abstract

This paper presents an algebraic based approach and a computational framework for the simulation of multi-axis CNC machining of general freeform tools. The boundary of the swept volume of the tool is precisely modeled by a system of algebraic constraints, using B-spline basis functions. Subdivision-based solvers are then employed to solve these equations, resulting in a topologically guaranteed construction of the swept volume. The presented algebraic-based method readily generalizes to accept tools of arbitrary free-form shape as input, and at the same time, delivers high degree of precision.

Being a common representation in CNC simulations, the computed swept volume can be reduced to a dexels representation. Several multi-axis test cases are exhibited using an implementation of our algorithm, demonstrating the robustness and efficacy of our approach.

1. Introduction

This work addresses the problem of precise computation of the swept volume of a solid moving along a one parameter family of rigid motions, in $\mathbb{R}^3$, towards 5-axis computer numerically controlled (CNC) machining verification. Owing to the inherent mathematical and computational complexity, a robust and efficient implementation of sweeps has remained elusive until now. If an approach has been general, it did not deliver a high degree of precision [9] and if an approach yielded accurate solution, it imposed serious restrictions on the class of inputs [42]. The proposed method offers precision, while accepting tool of arbitrary shape as input. This is done by resorting to an algebraic representation and employing state-of-the-art constraint solvers [7]. While the resulting algorithm accepts any solid with free-form boundary as input, swept along a general path in $\mathbb{R}^3$, in this work we demonstrate its efficacy by sweeping any axis-symmetric tool, along spherical linear interpolated (SLERP) motions, for the purpose of verification of 5-axis CNC machining.

CNC machining is an ubiquitous manufacturing technique. 5-axis CNC commands require complex spatial motions of cutters, where the positions of milling cutters have 5 degrees of freedom (one degree of freedom is eliminated due to the cutter's rotational symmetry). Apparently, the problem of finding a consistent tool path in 5-axis space, while ensuring no collisions, is non-trivial. In a case of a faulty cutter movement, significant financial losses may occur because of damaging expensive production parts. In order to prevent such occasions, the industry has adopted an additional step of verifying CNC tool paths, wherein, the correctness of the tool path is verified by subtracting the swept volume of the tool from the stock and comparing the resulting part against the desired part.

The boundary of the swept volume, henceforth the envelope, is a surface which can be defined implicitly via an under-constrained system of non-linear equations [38]. One of the two main contributions of this work is the precise construction of these non-linear algebraic constraints using B-spline basis functions [14] and solving these using robust algebraic constraint solvers [7] to a desired accuracy. The constraint solvers provide a topological guarantee for the returned solution, and ensure finding all real solutions, up to a specified tolerance. The second contribution of this work is the generality of the inputs accepted. Most previous algorithms for CNC machining simulation restrict the tool shape to be composed of simple surface patches such as spheres, cylinders, cones or tori. Such assumptions restrict the applicability of the algorithm. For instance, router bits used for wood-working [13] have more general shape and may not be simulated and verified by such an approach. Our algebraic formulation using B-spline functions allows for such general shapes.

No non-trivial solid sweep admits an explicit, closed-form representation [4] and some form of approximation, within a desired tolerance, must be performed, while constructing the envelope surface of the swept volume. Most methods sample a set of points on the envelope, at seemingly arbitrary locations, which are then interpolated. In contrast, the proposed method integrates this tolerance into the envelope construction at an early stage to derive a computational advantage. While the proposed approach can be arbitrarily precise, we also support the practical representation of dexels [35], that is a common choice in CNC machining simulation. This is done by computing slices of the computed sweep envelope along pre-defined planes and lines in $\mathbb{R}^3$, towards a dexel representation of the swept volumes. Finally, self-intersections - a difficult problem in envelope construction - are naturally and efficiently handled.

The rest of the paper is organized as follows. A brief survey of previous CNC simulation methods is given in Section 2. Then, Section 3 formulates the problem and gives the required definitions, only to be followed by our method that consists of two main phases:

i The primary step consists of construction of the envelope condition as a system of algebraic equations, which are solved up to the desired precision using algebraic constraint solvers. This is discussed in Section 4.

ii The secondary step involves obtaining a dexel representation of the envelope from the precise solution of the above algebraic constraints. The boundary of the swept volume is thus represented in a discrete form. This, along with the overall algorithm is portrayed in Section 5.

The resulting algorithm is implemented with the aid of the IRIT [16] kernel and results on CNC machining simulation are given in Section 6. Section 7 concludes the paper with directions along which this work may be extended.

2. Previous work

Verification of CNC tool paths is a simulation of material removal (or material adding in 3D additive manufacturing processes). The most advanced simulation methods are based on the computation of swept volumes, which is a widely used technique for verification of 5-axis milling operations. First was the theory of envelopes [38] that modeled swept volumes by computing a family of silhouettes, also known grazing curves, of the moving tool and "stretching" surfaces over these silhouettes. However, this particular approach fails to handle cutters with sharp edges and self-intersections of the swept volumes.

In order to overcome these limitations, the sweep-envelope differential equation (SEDE) method was developed [26, 9] and is one of the most comprehensive methods for sweeps. This approach uses the trajectories of the sweep-envelope differential equation which start at the initial grazing points of the moving object. The SEDE method computes only one grazing curve at the initial position of the tool, so the computation complexity is drastically reduced. Intermediate grazing points are computed from differential equations based on the tool motion equation, using the Runge-Kutta method. The SEDE method assumes the tool profile to be smooth; therefore, precise verification for a flat-end tool (cylinder shape) requires approximation of the tool shape (rectangular) with relatively high order equations [26]. Further, the SEDE method was extended to calculate the swept volume generated by machining tools for a large class of sweeps [37], and to detect and model self-intersections [30] (ball-end mill only). However, the above methods suffer from lack of precision which arises from the numerical solution of differential equations.

Several approaches are based on the Jacobian rank deficiency [1, 2, 3, 43]. The boundary of swept volume is defined as the singular set of the Jacobian of the sweep map. The boundary to the resulting swept volume, in terms of enveloping surfaces, is obtained as the image of the singular set through the sweep map. Due to the generality of the Jacobian condition, this approach works for sweeps of multiple dimensions, however, due to symbolic computation used for finding the singular sets, only analytic surfaces are accepted as input.

Some researches optimized finding swept volumes for a generalized milling tool shape. They exploited the standard definition constraints applied on the cutter profile. By introducing rigid body motion theory to define a moving frame, the closed-form solutions of swept profiles for a generalized cutter geometry were obtained [11, 15, 45]. Another parametric approach models the envelope surface by decomposing the cutter into a set of characteristic and great circles which are generated by two-parameter families of spheres [6]. While the above methods give precise solutions, they restrict the tool shape to be composed of spheres, cylinders, cones and tori.

Further improvements were focused on special techniques to address a particular simulation issue. For instance, the problem of self-intersection, for a ball-end cutter, was partly resolved [23] using the triangle strip representation, which leads to a discretization of the boundary of the swept volume. Another example includes [31, 32] that focused on toroidal cutters, which was later extended to cutter shapes [27] that include spheres and cones. In [22], swept surface model of a cutter, in the shape of a drum, was discretized and represented with voxels, leading to imprecision in the solution. Clearly, the above methods are tailored for tools of specific shapes and do not accept general tools as input.

The authors in [12] derived the silhouette curve constraint for a toroidal cutter surface as a quartic polynomial and modeled the envelope surface using Z-buffers. Further, [21] defined a polygon boundary of the volume swept by using instantaneous helical motions, i.e., translation along an axis, compounded with rotation around the same axis. These methods rely on special tool shapes such as torus, ball- or flat end and do not accept general freeform surfaces as input.

Swept volume boundary was also derived with the aid of the Gauss map [24, 25]. It takes piecewise C^1-continuous tool shape into account and identifies self-penetration of the cutter at 5-axis movement. The swept volume of the cutter is generated by stitching up envelope profiles at discrete time instants. This method is limited to bull-nose, ball-end and flat-end tools.

In some cases, the envelopes of swept volumes are represented with various surfaces. [40] syntheses NURBS motion such that the swept surface of such a motion with a cylindrical cutter closely approximates a given ruled surface. Another approach, which fits a NURBS surface to a set of envelope profiles was introduced in [39] for toroidal cutters. Besides, Bèzier motion was used as well [41, 42], in order to derive an exact representation of the swept volume of a flat-end cylindrical cutter.

Peternell et al. [29] obtain a set of points on a sampled set of characteristic curves. A formula is derived for the evolution of the characteristic curve which ensures that no two characteristic curves are too far. This approach, however, does not yield a topological guarantee for the constructed envelope. Wallner et al. [36] compute the boundary of the swept volume of a given body along motion specified by a discrete pose cloud, however, the input object is restricted to be polyhedral. A procedural implicit description of the swept volume of a solid is given in [19]. This approach requires a curve-surface intersection computation in order to answer the membership query for each point in $\mathbb{R}^3$ with respect to the swept volume. Nelaturi et al. [28] generalize sweeps and Minkowski sums via operations of products and quotients of configurations spaces. Sullivan et al. [33] model the swept volume of a cutter using adaptive distance fields. In this approach a shape is modeled, either analytically or procedurally, by a signed Euclidean distance field. While such an approach is fast and also gives good numerical precision, topological guarantees are lacking. There have also been efforts to study the approximation of free-form objects by envelopes of surfaces of revolution [10, 5, 44, 8].

3. Preliminaries

In this section, we define some preliminary constructs which are used by our method. We adopt an algebraic approach in order to model the envelope. The property of the B-splines basis functions, being closed under addition and multiplication [17], is extensively exploited in order to derive a precise representation of the well-known envelope condition [38], that defines the boundary of the swept volume as an implicit surface in a 3-dimensional parameter space. Thanks to our B-splines representation, the envelope condition can be evaluated precisely and is solved up to the desired precision, using constraint solvers. This section provides a prelude to our algebraic construction.

Let T denote the tool in $\mathbb{R}^3$ undergoing a sweep motion. We use a parametric boundary representation (B-rep) form for T, wherein, T is represented by its boundary, ∂T, which separates the interior of T from its exterior and is composed of a number of C^1 continuous patches of surfaces and curves. In this work, all the curves and surface patches comprising ∂T are modeled using B-spline basis functions. T is swept along a path in $\mathbb{R}^3$ defined as follows:

Definition 1. A **one parameter family of rigid motions** in $\mathbb{R}^3$ is defined as a map $M : [0,1] \to (SO(3), \mathbb{R}^3)$, where $SO(3)$ is the group of 3×3 rotation matrices.

We assume here for simplicity, and without loss of generality, that the time interval of the sweep is $[0,1]$. For each $t \in [0,1]$, $M(t) = (A(t), b(t))$ where, $A(t)$ is a rotation matrix and $b(t)$ is a translation vector. The entries of the matrix $A(t)$ as well as the curve $b(t)$ are modeled, again, using B-spline basis functions. The motion M is assumed to be C^1 continuous and acts on a point $p \in \mathbb{R}^3$ at time t as $p \mapsto A(t)p + b(t)$. Further, the tool T at time t under M is given by $\{A(t)p + b(t) | p \in T\}$ and is denoted by $T(t)$.

Definition 2. Given a tool T and a motion M, the **swept volume** of T along M is defined as $\{\bigcup_{t\in[0,1]} T(t)\}$ and denoted by $\mathcal{V}$. The **envelope** is the boundary of $\mathcal{V}$ and denoted by $\partial\mathcal{V}$.

Since we use the B-rep form, it suffices to compute $\partial\mathcal{V}$ in order to obtain a complete representation of $\mathcal{V}$. Towards this, we define the boundary operator as follows:

Definition 3. Given a set Y and a subset $X \subset Y$, the **boundary operator** takes X and gives the boundary of X in Y. We denote the boundary operator within a given set Y by ∂_Y.

For instance, $\partial_{\mathbb{R}^3}(\mathcal{V}) = \partial\mathcal{V}$. In Section 4, the boundary operator is invoked within $\mathbb{R}^3$, as well as within lines and planes in $\mathbb{R}^3$.

4. The proposed method

At the core of our method lies the precise construction of the envelope condition and obtaining its solution using constraint solvers. In this section, we describe three alternative approaches to compute the envelope, which differ in the dimension of the solution space. The envelope is characterized by the necessary condition that the velocity vector at a point on the tool surface is tangent to the tool at that point [38]. This leads to a constraint with three variables in the parameter space of the sweep, whose solution, in general, is a set of 2-manifolds (surfaces). Section 4.1 describes the direct computation of the envelope that returns 2-manifold solutions. In Section 4.2, we compute slices of the envelope along planes and directly return 1-manifold (curves) solution slices on the envelope. The third alternative, which involves computing intersections of the envelope with lines, is described in Section 4.3.

4.1. Complete tool envelope computation

Consider a smooth regular parametric surface patch, $F(u,v) \subset \partial T$, for $(u,v) \in D$. Here, D is the domain of F and without loss of generality, assumed to be $[0,1]^2$. We use tensor-product B-spline surfaces to model F. One may use rational B-splines in order to precisely model surfaces such as spheres and cylinders. Rationals are seamlessly supported by our approach, since the denominator (that is positive throughout) may be ignored while computing the zero-set. With some abuse of notation, we denote the transformation of $F(u,v)$, at time $t \in [0,1]$, under the motion M as:

$$F(u,v,t) := A(t)F(u,v) + b(t). \tag{1}$$

B-spline functions are closed under addition and multiplication. Hence, the map $F(u,v,t)$ may be precisely expressed in an algebraic form using B-splines. The velocity of any point (u,v,t) is given by $V(u,v,t) := A'(t)F(u,v) + b'(t)$, where $'$ denotes the derivative with respect to t. Let $N(u,v)$ denote the unnormalized outward normal to F, at (u,v). The outward normal at $F(u,v,t)$ is given by $N(u,v,t) := A(t)N(u,v)$. Clearly, $N(u,v,t)$ and $V(u,v,t)$ are also B-spline forms. The envelope generated by the sweep of F is obtained as the solution of the following algebraic constraint (precisely representable in B-spline form) in three variables, (u,v,t) [38]:

$$\partial\mathcal{V} : \langle N(u,v,t), V(u,v,t)\rangle = 0. \tag{2}$$

Here $\langle , \rangle$ is the standard inner product in $\mathbb{R}^3$. Hence, Equation (2) is also amenable to a representation using B-spline functions. The solution to Equation (2) is, in general, a 2-manifold, whose image through the map $F(u,v,t)$ gives the envelope. Figure 1 shows in blue, a surface of revolution. Its envelope surface is shown in transparent red. Equation (2) is solved for all surface patches of ∂T in order to compute the corresponding component surface patches of $\partial\mathcal{V}$. Furthermore, curves along G^1 discontinuities of ∂T, i.e., where two faces of ∂T meet with G^1-discontinuity, also give rise to surface patches of $\partial\mathcal{V}$.

Let $C \subset \partial T$ be a C^1 discontinuous curve of ∂T. As T moves along M, C sweeps a surface in $\mathbb{R}^3$, part of which may appear on $\partial\mathcal{V}$. Let $C(r)$ denote the parametrization of C for $r \in I$. Here I is the domain of C. The transformation of $C(r)$ under M, at time $t \in [0,1]$, is given by

$$C(r,t) := A(t)C(r) + b(t). \tag{3}$$

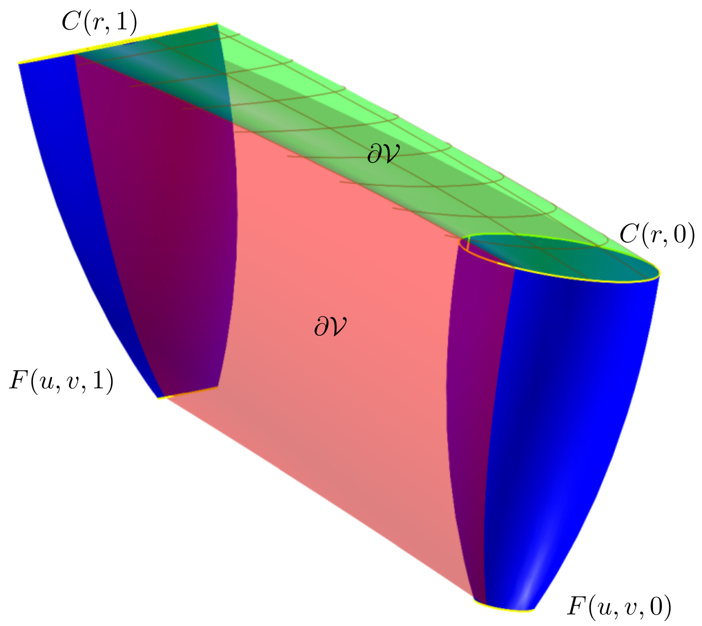


Figure 1: The envelope of a sweep of a surface patch, F, of ∂T is a 2-manifold. F is shown in blue at initial and final positions. The envelope surface generated by F is shown in transparent red. The C^1 discontinuity circular curve $C(r,t)$ at the top flat end of the tool, shown in yellow, generates the envelope surface which is shown in transparent green.

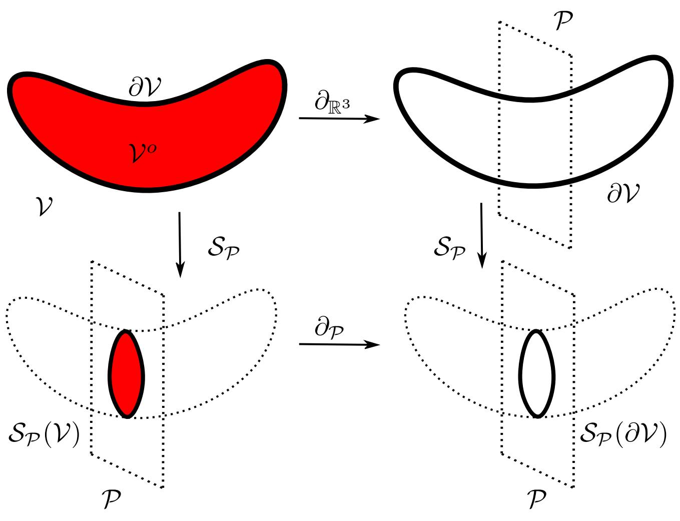

$\partial\mathcal{V}$
$\mathcal{V}^o$
$\mathcal{V}$
$\partial_{\mathbb{R}^3}$
$\mathcal{P}$
$\partial\mathcal{V}$
$\mathcal{S}_{\mathcal{P}}$
$\mathcal{S}_{\mathcal{P}}$
$\partial_{\mathcal{P}}$
$\mathcal{S}_{\mathcal{P}}(\mathcal{V})$
$\mathcal{S}_{\mathcal{P}}(\partial\mathcal{V})$
$\mathcal{P}$
$\mathcal{P}$

Figure 2: The above diagram commutes. The swept volume $\mathcal{V}$ is shown at the top left, with $\partial\mathcal{V}$ and interior, $\mathcal{V}^o$, shown in black and red respectively. The envelope, $\partial\mathcal{V}$, can be obtained by applying $\partial_{\mathbb{R}^3}$ to $\mathcal{V}$ and is shown at the top right. The plane $\mathcal{P}$ is indicated by a dotted parallelogram. The set $\mathcal{S}_{\mathcal{P}}(\partial\mathcal{V})$ is shown at the bottom right by a closed curve shown in black. An alternative, more efficient, way of computing $\mathcal{S}_{\mathcal{P}}(\partial\mathcal{V})$ is via $\mathcal{S}_{\mathcal{P}}(\mathcal{V})$, shown at the bottom left.

The envelope surface $C(r,t)$ is shown in transparent green in Figure 1 and is generated by the circular edge of the tool, shown in yellow. A few iso-curves of the surface $C(r,t)$ are also shown.

The envelope thus computed is precise up to the tolerance supplied to and provided by the solver. Note, however, that the envelope at this stage may be prone to local and global self-intersections, a difficult problem in general sweeps. Trimming of such self-intersections in our context is discussed in Section 5.

4.2. Sections of $\partial\mathcal{V}$ along planes

We now come to the direct computation of slices of $\partial\mathcal{V}$ along general planes, in $\mathbb{R}^3$. Towards this, we define the section operator as follows.

Definition 4. Given a set $X \subset \mathbb{R}^3$ and a plane $\mathcal{P} \subset \mathbb{R}^3$, the **section of** X **along** $\mathcal{P}$ is defined as the intersection $X \cap \mathcal{P}$ and denoted by $\mathcal{S}_{\mathcal{P}}(X)$. A section along a line, $\mathcal{L}$, in $\mathbb{R}^3$ is defined similarly, and denoted $\mathcal{S}_{\mathcal{L}}(X)$.

The following simple, yet powerful, Lemma is employed by our method in order to speed-up the computation of sections of $\partial\mathcal{V}$:

Lemma 5. *While acting on the swept volume, the section operator commutes with the boundary operator, i.e.,* $\mathcal{S}_{\mathcal{P}}(\partial_{\mathbb{R}^3}(\mathcal{V})) = \partial_{\mathcal{P}}(\mathcal{S}_{\mathcal{P}}(\mathcal{V}))$.

In other words, the diagram shown in Figure 2 commutes. It is much more efficient to compute $\partial_{\mathcal{P}}(\mathcal{S}_{\mathcal{P}}(\mathcal{V}))$ rather than to compute $\mathcal{S}_{\mathcal{P}}(\partial_{\mathbb{R}^3}(\mathcal{V}))$, since it is simpler to apply $\partial_{\mathcal{P}}$ compared to applying $\partial_{\mathbb{R}^3}$. Here, we also compare the univariate sections of the envelope along planes with univariate characteristic curves [29], which are subsets of the envelope for fixed time instants. The characteristic curves are more geometrically meaningful compared to sections along planes since the later contain points of the envelope from many time instants. Although, sections along planes are more computationally efficient.

Let $\mathcal{P}$ be a plane in $\mathbb{R}^3$ given by $A_p x + B_p y + C_p z + D_p = 0$. Let F be a C^1-continuous, regular, surface patch of ∂T with a parametrization, as described in Section 4.1. The section, $\mathcal{S}_{\mathcal{P}}(\partial\mathcal{V})$, of the sweep of F is obtained as the solution of the following two simultaneous algebraic constraints (in B-spline form) in three variables, (u,v,t):

$$\begin{aligned} \langle N(u,v,t), V(u,v,t)\rangle &= 0, \\ \langle F(u,v,t), (A_p, B_p, C_p)\rangle + D_p &= 0. \end{aligned} \tag{4}$$

The first constraint in Equation (4), identical to Equation (2), ensures that point (u,v,t) satisfies the envelope condition. The second constraint ensures that point $F(u,v,t)$ lies on $\mathcal{P}$. We use the univariate constraint solver [7] of the IRIT [16] kernel to solve the above set of equations. The solution to the above set of equations, in general, is a 1-manifold. The solution is a set of curves in the parameter space, $D \times [0,1]$, and its image through $F(u,v,t)$ gives the section of the envelope along the plane. This is illustrated in Figure 3. The plane is shown in transparent cyan. The face F at initial and final positions is shown in blue. The envelope is shown in transparent red. The section, $\mathcal{S}_{\mathcal{P}}(\partial\mathcal{V})$, is indicated by curve shown in black. Note that, an axis orthogonal plane will result in the same computational complexity as any general plane, $\mathcal{P}$, since the complexity of the resulting algebraic constraints remains the same in both the cases.

We now come to the sections of envelope surfaces generated by C^1 discontinuity curves of ∂T. Let $C(r)$ be a C^1 discontinuity curve of ∂T with parametrization as described in Section 4.1. We compute the section of surface,

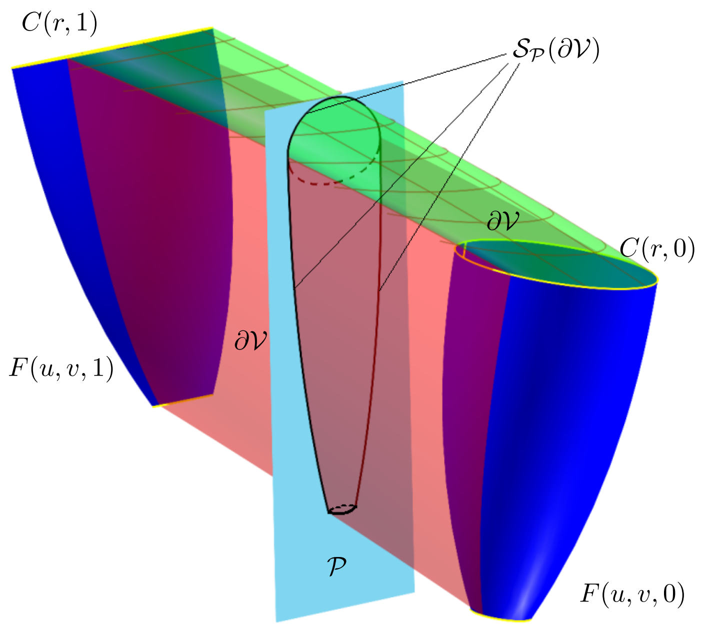
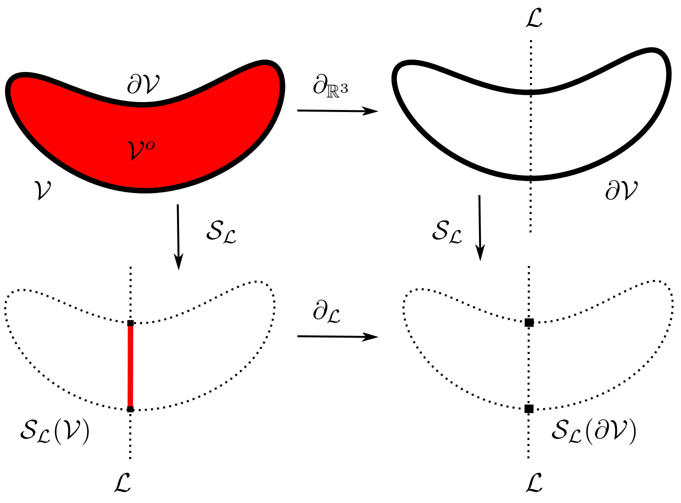


Figure 3: The section of the envelope along a plane. The envelope surfaces, generated by the face $F \subset \partial T$ (shown in blue) and curve $C \subset \partial T$ (shown in yellow), are shown in transparent red and green, respectively. The plane, $\mathcal{P}$, is shown in transparent cyan and the section, $\mathcal{S}_{\mathcal{P}}(\partial\mathcal{V})$, is indicated by curves shown in black.



$\partial\mathcal{V}$
$\mathcal{V}^o$
$\mathcal{V}$
$\partial_{\mathbb{R}^3}$
$\mathcal{L}$
$\partial\mathcal{V}$
$\mathcal{S}_{\mathcal{L}}$
$\mathcal{S}_{\mathcal{L}}$
$\partial_{\mathcal{L}}$
$\mathcal{S}_{\mathcal{L}}(\mathcal{V})$
$\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$
$\mathcal{L}$
$\mathcal{L}$

Figure 4: The above diagram commutes. The swept volume $\mathcal{V}$ is shown at the top left, with $\partial\mathcal{V}$ and $\mathcal{V}^o$ shown in black and red respectively. The envelope, $\partial\mathcal{V}$, is obtained by applying $\partial_{\mathbb{R}^3}$ to $\mathcal{V}$ and is shown at the top right. The line $\mathcal{L}$ is indicated by a dotted line. The set $\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$ is shown at the bottom right by a pair of points shown in black. An alternative way of computing $\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$ is via $\mathcal{S}_{\mathcal{L}}(\mathcal{V})$, shown at the bottom left, by a red line-segment.

$C(r,t)$, defined in Equation (3), along the plane, $\mathcal{P}$, by solving the following algebraic equation in two variables, (r,t):

$$\langle C(r,t),(A_p,B_p,C_p)\rangle + D_p = 0. \tag{5}$$

This is also illustrated in Figure 3. The envelope is shown in transparent green, generated by the curve, shown in yellow. The envelope shown in green is only partially visible, since it obstructed by the envelope surface shown in pink, generated by the surface, $F(u,v)$, shown in blue. The plane, $\mathcal{P}$, is shown in transparent cyan. The section of the envelope along $\mathcal{P}$ is a closed curve and is shown in black. Note that some portion of this curve does not lie on the boundary of the swept volume and is indicated by a dashed curve.

4.3. Sections of $\partial\mathcal{V}$ along lines

The computation of sections of $\partial\mathcal{V}$ along lines in $\mathbb{R}^3$, follows similar to the sections along planes, as described in Section 4.2. The section $\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$ along any line $\mathcal{L}$ is, in general, a set of points. We again exploit the fact that the boundary operator commutes with the section operator along lines, as illustrated schematically in Figure 4.

Let the line $\mathcal{L}$ be defined by the intersection of planes $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_1 \neq \mathcal{P}_2$, where $\mathcal{P}_i$ is given by $A_p^i x + B_p^i y + C_p^i z + D_p^i = 0, i = 1,2$. The section $\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$ for the sweep of C^1 continuous, regular, surface patch F of ∂T is obtained as the solution of the following three simultaneous algebraic equations (in B-spline form) in three variables, (u,v,t):

$$\begin{aligned} \langle N(u,v,t),V(u,v,t)\rangle &= 0,\\ \left\langle F(u,v,t),(A_p^1,B_p^1,C_p^1)\right\rangle + D_p^1 &= 0,\\ \left\langle F(u,v,t),(A_p^2,B_p^2,C_p^2)\right\rangle + D_p^2 &= 0. \end{aligned} \tag{6}$$

The first constraint in Equation (6), identical to Equation (2), ensures that the point (u,v,t) satisfies the envelope condition. The second and third constraints ensure that the point $F(u,v,t)$ lies on the planes $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively, i.e., on the line $\mathcal{L}$. In order to ensure that the solver is well-behaved, the planes $\mathcal{P}_1$ and $\mathcal{P}_2$ are chosen so that the angle between them is $\frac{\pi}{2}$. The solution set, which is a set of points in the parameter space $D \times [0,1]$ is mapped to the object space, by the function $F(u,v,t)$. This is illustrated in Figure 5. The planes $\mathcal{P}_1, \mathcal{P}_2$ are shown in transparent cyan. The section $\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$ is indicated by a point shown in black on the envelope surface shown in transparent red.

The section of the envelope surfaces generated by a C^1 discontinuity curve, $C(r) \subset \partial T$, is obtained as the solution to the following two algebraic equations in three variables, (r,t):

$$\begin{aligned} \left\langle C(r,t),(A_p^1,B_p^1,C_p^1)\right\rangle + D_p^1 &= 0,\\ \left\langle C(r,t),(A_p^2,B_p^2,C_p^2)\right\rangle + D_p^2 &= 0. \end{aligned} \tag{7}$$

In the example shown in Figure 5, the line $\mathcal{L}$ intersects the surface $C(r,t)$ in two nearby locations, which are indicated by a pair of black points on the surface $C(r,t)$ shown in transparent green. The curve $C(r,0)$ is shown in yellow.

5. Overall algorithm

Section 4 presented three schemes to precisely evaluate the envelope of a general tool, along a 5-axis motion. In this section, we show how this formulation can be used towards a representation common in CNC machining simulation, namely dexels (depth pixels) [35]. We give a dexels based algorithm for computing the swept volume of a tool

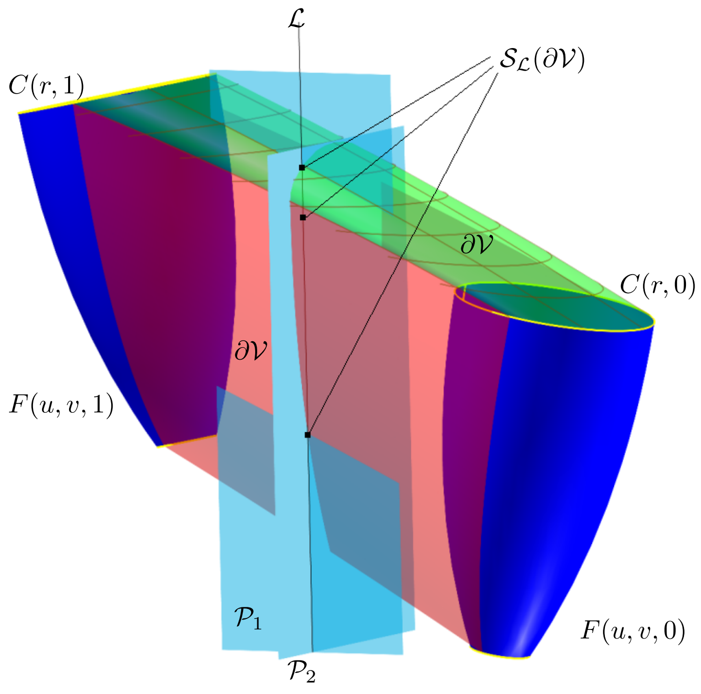


Figure 5: The section of the envelope along a line. The line $\mathcal{L}$ is the intersection of two planes, $\mathcal{P}_1, \mathcal{P}_2$, shown in transparent cyan. The section $\mathcal{S}_{\mathcal{L}}(\partial\mathcal{V})$ is indicated by points shown in black.

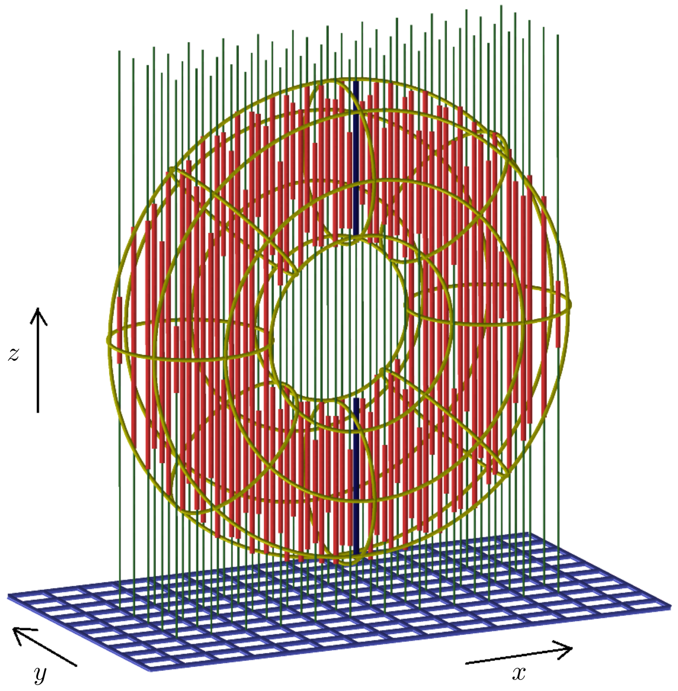


Figure 6: A dexel-grid with 16×11 dexels, representing a torus. The dexels intervals which represent the interior of the torus are shown in red. One of the dexels has been highlighted in blue.

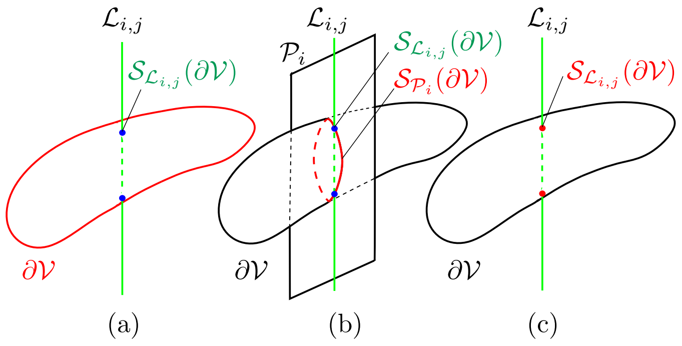

$\mathcal{L}_{i,j}$
$\mathcal{S}_{\mathcal{L}_{i,j}}(\partial\mathcal{V})$
$\partial\mathcal{V}$
(a)
$\mathcal{L}_{i,j}$
$\mathcal{P}_i$
$\mathcal{S}_{\mathcal{L}_{i,j}}(\partial\mathcal{V})$
$\mathcal{S}_{\mathcal{P}_i}(\partial\mathcal{V})$
$\partial\mathcal{V}$
(b)
$\mathcal{L}_{i,j}$
$\mathcal{S}_{\mathcal{L}_{i,j}}(\partial\mathcal{V})$
$\partial\mathcal{V}$
(c)

Figure 7: The difference between the three algorithmic variants is illustrated. The solution returned by these algorithms, which is of dimension two, one and zero, in (a), (b) and (c), respectively, is shown in red in each case.

towards 5-axis CNC machining simulation, using the results of Section 4. Dexels are generalizations of Z-buffers. While a Z-buffer maintains a single height value at each point of a 2D grid, a dexel-grid may store multiple height intervals at each point of the 2D grid.

Without loss of generality, we assume that the dexels are parallel to the z-axis. Let X and Y denote the number of columns and rows respectively for the dexel grid, $\mathcal{D}$, so that $\mathcal{D}$ is divided into $X \times Y$ dexel locations. Figure 6 illustrates a dexel-grid with $X = 16, Y = 11$, representing a torus. Each dexel, $\mathcal{D}_{i,j}$, corresponds to a point $(x_i, y_j, 0) \in \mathbb{R}^3$ and maintains a list of real z-intervals which represent the material of the stock to be machined. The distances between dexels along the x and y directions determine the fineness of approximation of the swept volume and are supplied by the user. These intervals are updated as the computation of the envelope proceeds.

The three different algebraic formulations described in Section 4 give rise to three possible algorithmic variants, which are illustrated schematically in Figure 7. The three differ in the dimension of the solution returned, which is shown in red in Figure 7. The algorithm which uses the formulation of Section 4.1 is summarized in Algorithm 1 and is illustrated in Figure 7(a). It takes as input, the dexel grid, $\mathcal{D}$, representing the raw stock which is to be machined, the tool, T, in B-rep form, the motion, M, and the numeric tolerance value, ϵ.

First, for each surface patch, $F \subset \partial T$, the envelope surface is obtained as a bivariate solution to Equation (2). This is done by the function EnvelopeSrf2D in lines 2-4 of Algorithm 1. Secondly, for each C^1 discontinuity curve $C \subset \partial T$, the envelope surface is obtained by constructing Equation (3). This is done by the function EnvelopeCrv2D in lines 5-7 of Algorithm 1. The list, *Envl2D*, now contains all the envelope surfaces derived from all the surface patches as well as C^1 discontinuity curves of ∂T.

Now begins the dexel approximation phase. Let $\mathcal{L}_{i,j}$ denote the line along the dexel $\mathcal{D}_{i,j}$. Each line $\mathcal{L}_{i,j}, i = 0, \ldots, X-1, j = 0, \ldots, Y-1$, is intersected with each of the envelope surfaces in *Envl2D* using a line-surface intersection routine. This gives a set of points, along with the normals to the respective envelope surfaces at each of these points. This is done in line 13 of Algorithm 1. To this set of points which lie on line $\mathcal{L}_{i,j}$, the boundary operator, $\partial_{\mathcal{L}_{i,j}}$, is applied in line 15.

The boundary operator, $\partial_{\mathcal{L}}$, for any line, $\mathcal{L}$, introduced in Definition 3, discards the points which do not belong to the boundary of the swept volume, after sorting them by their z-coordinate, into paired intervals. Such points arise due to local and global self-intersections in the envelope. For example, in the sweep shown in Figure 5, the intersection of line $\mathcal{L}$ with the envelope surfaces gives three points, shown in black. One of these points, viz., the one in the middle, does not lie on $\partial\mathcal{V}$.

The operator $\partial_{\mathcal{L}}$ works by inspecting the normals to the envelope at the points it received. Since the dexels are assumed to be along the z-axis, a point with negative or positive z-coordinate of the normal is considered as the start or end of a real interval, respectively. The boundary operator is summarized in Algorithm 2. The set of intervals thus obtained is the section of the swept volume of the tool along the line $\mathcal{L}_{i,j}$, i.e., $\mathcal{S}_{\mathcal{L}_{i,j}}(\partial\mathcal{V})$. This set of intervals is subtracted from the dexel $\mathcal{D}_{i,j}$ representing the stock, using a 1D Boolean subtraction operator in line 16 of Algorithm 1. This subtraction is equivalent to the removal of material from the stock by the tool.

Algorithm 1 ComputeEnvelope2D($\mathcal{D}, T, M, \epsilon$)

```
1:  $Envl2D \leftarrow \emptyset$;
2:  for all $F \in \partial T$ do
3:      $Envl2D \leftarrow Envl2D \cup$ EnvelopeSrf2D($F, M, \epsilon$);
4:  end for
5:  for all $C \in \partial T$ do
6:      $Envl2D \leftarrow Envl2D \cup$ EnvelopeCrv2D($C, M$);
7:  end for
8:  for all $i \in \{0, \ldots, X-1\}$ do
9:      for all $j \in \{0, \ldots, Y-1\}$ do
10:         $Pts \leftarrow \emptyset$;
11:         $\mathcal{L}_{i,j} \leftarrow$ LineZ($i, j$);
12:         for all $Srf \in Envl2D$ do
13:             $Pts \leftarrow Pts \cup$ LineSrfInter($Srf, \mathcal{L}_{i,j}$);
14:         end for
15:         $Intrvls \leftarrow$ BoundaryOp($Pts$);
16:         BooleanSubtract($\mathcal{D}_{i,j}, Intrvls$);
17:     end for
18: end for
```

The second variant of our algorithm, which exploits the algebraic formulation given in Section 4.2, is summarized in Algorithm 3 and is illustrated schematically in Figure 7(b). In this case, univariate sections of the envelope are obtained along planes, $\mathcal{P}_i, i = 0, \ldots, X-1$, which are parallel to the yz-plane. This is done in lines 3-5 of Algorithm 3 by the function EnvelopeSrf1D which takes as input, a surface patch $F \subset \partial T$, the motion, M, the x-intercept, x_i, of the plane $\mathcal{P}_i$, and the numeric tolerance ϵ and solves Equation (4). The univariate sections of the envelope generated by C^1 discontinuity curves of ∂T are computed in lines 6-8 of Algorithm 3, by the function EnvelopeCrv1D, by solving Equation (5).

We thus have 1-dimensional slices of the envelope, computed along a set of parallel planes in $\mathbb{R}^3$. The number of such planes as well as the distance between two consecutive planes can be dictated by the desired dexels' tolerance. Each section of the envelope is topologically guaranteed, up to the tolerance specified to the solver. Each section, $\mathcal{S}_{\mathcal{P}_i}(\partial\mathcal{V})$, which is a set of curves, is intersected with each of lines $\mathcal{L}_{i,j}, j = 0, \ldots, Y-1$. For each line, a set of points is obtained. This is followed by the dexel approximation and is similar to that described for Algorithm 1.

We now come to the final variant of our algorithm, which uses the 0-dimensional formulation described in Section 4.3

Algorithm 2 BoundaryOp($PtList$)

```
1:  $Intrvls \leftarrow \emptyset$;
2:  $IntrvlStart \leftarrow NULL$;
3:  $IntrvlEnd \leftarrow NULL$;
4:  $ZSortedPtList \leftarrow$ SortByZCoord($PtList$);
5:  for all $Pt \in ZSortedPtList$ do
6:      if $Pt.Nrml.z < 0$ then
7:          if $IntrvlStart = NULL$ then
8:              $IntrvlStart \leftarrow Pt$;
9:          else if $IntrvlEnd \neq NULL$ then
10:             $Intrvls \leftarrow Intrvls \cup$
                        $[IntrvlStart, IntrvlEnd]$;
11:             $IntrvlStart \leftarrow Pt$;
12:             $IntrvlEnd \leftarrow NULL$;
13:         end if
14:     else if $Pt.Nrml.z > 0$ then
15:         if $IntrvlStart \neq NULL$ then
16:             $IntrvlEnd \leftarrow Pt$;
17:         end if
18:     end if
19: end for
20: return $Intrvls$;
```

Algorithm 3 ComputeEnvelope1D($\mathcal{D}, T, M, \epsilon$)

```
1:  for all $i \in \{0, \ldots, X-1\}$ do
2:      $Envl1D \leftarrow \emptyset$;
3:      for all $F \in \partial T$ do
4:          $Envl1D \leftarrow Envl1D \cup$
                        EnvelopeSrf1D($F, M, x_i, \epsilon$);
5:      end for
6:      for all $C \in \partial T$ do
7:          $Envl1D \leftarrow Envl1D \cup$
                        EnvelopeCrv1D($C, M, x_i, \epsilon$);
8:      end for
9:      for all $j \in \{0, \ldots, Y-1\}$ do
10:         $Pts \leftarrow \emptyset$;
11:         $\mathcal{L}_{i,j} \leftarrow$ LineZ($i, j$);
12:         for all $Crv \in Envl1D$ do
13:             $Pts \leftarrow Pts \cup$ LineCrvInter($Crv, \mathcal{L}_{i,j}$);
14:         end for
15:         $Intrvls \leftarrow$ BoundaryOp($Pts$);
16:         BooleanSubtract($\mathcal{D}_{i,j}, Intrvls$);
17:     end for
18: end for
```

and is illustrated schematically in Figure 7(c). In this case, for each of the surface patches of ∂T, the section of the envelope is computed along each of the lines $\mathcal{L}_{i,j}, i = 0, \ldots, X-1, j = 0, \ldots, Y-1$, by solving Equation (6). The same is done for C^1 discontinuity curves of ∂T, by solving Equation (7). This gives a set of points on which the boundary operator is applied, as in the previous two cases. Planes $\mathcal{P}_1$ and $\mathcal{P}_2$, being orthogonal to the x- and y-axes respectively, yield a set of points on the envelope, with uniform distance along the x- and the y-axes. Some of the points may not lie on the boundary of the swept volume.

Culling of such points is, again, done by the boundary operator. The algorithm is summarized in Algorithm 4.

Algorithm 4 ComputeEnvelope0D($\mathcal{D}, T, M, \epsilon$)

```
1:  for all $i \in \{0, \ldots, X-1\}$ do
2:      for all $j \in \{0, \ldots, Y-1\}$ do
3:          $Envl0D \leftarrow \emptyset$;
4:          $\mathcal{L}_{i,j} \leftarrow$ LineZ($i, j$);
5:          for all $F \in \partial T$ do
6:              $Envl0D \leftarrow Envl0D \cup$ EnvelopeSrf0D($F, M, x_i, y_j, \epsilon$);
7:          end for
8:          for all $C \in \partial T$ do
9:              $Envl0D \leftarrow Envl0D \cup$ EnvelopeCrv0D($C, M, x_i, y_j, \epsilon$);
10:         end for
11:         $Intrvls \leftarrow$ BoundaryOp($Envl0D$);
12:         BooleanSubtract($\mathcal{D}_{i,j}, Intrvls$);
13:     end for
14: end for
```

The difference between the three variants of our algorithm is illustrated schematically in Figure 7. The dimension of the solution manifolds returned by the solvers is two, one and zero in Algorithms 1, 3 and 4, respectively, and is shown in red, in Figure 7(a), 7(b) and 7(c). The performance of these three variants is compared in the next section.

6. Results

In this section, we present results from an implementation of the algorithm proposed in Section 5. First, we compare the three algorithmic variants, viz., Algorithms 1, 3 and 4, given in Section 5. Then, we give four examples of 5-axis machining simulations using analytic as well as freeform tool shapes, along SLERP motions.

The motion of the tool is specified as a series of positions and orientations of the tool, $(p_i, o_i) \in \mathbb{R}^3 \times S^2, i = 0, \ldots, K$, where, p_i is the position of the tool center in $\mathbb{R}^3$ and o_i is the orientation of the axis of the tool, specified as a unit-length vector in $\mathbb{R}^3$. For each pair of consecutive positions (p_i, o_i) and (p_{i+1}, o_{i+1}), the tool motion, $M_i(t) = (A_i(t), b_i(t)), t \in [0, 1]$ is derived as follows. The translation $b_i(t)$ is specified by the linear interpolation of the points p_i and p_{i+1}, i.e., $b_i(t) = (1-t)p_i + tp_{i+1}$. The rotation, $A_i(t)$, is specified by SLERP between vectors o_i and o_{i+1}. Let θ_i denote the angle between o_i and o_{i+1}. The matrix, $A_i(t)$, prescribes a rotation around the axis $o_{i+1} \times o_i$ by an angle, $\alpha_i(t)$, so that, $\alpha_i(0) = 0$ and $\alpha_i(1) = \theta_i$. The entries of the matrix $A_i(t)$ are composed of the trigonometric functions $\cos(\alpha_i(t))$ and $\sin(\alpha_i(t))$. We construct these trigonometric functions as rational B-spline functions by creating an arc of angle θ_i of a unit circle whose two coordinates furnish the functions $\cos(\alpha_i(t))$ and $\sin(\alpha_i(t))$.

We compare the running times of Algorithms 1, 3 and 4 on three types of tools, viz., ball-end, flat-end and freeform. Each of these tools is swept along two kinds of motions. Two different dexel grid sizes are used for comparison. We begin by describing how each of these tools is processed by our framework.

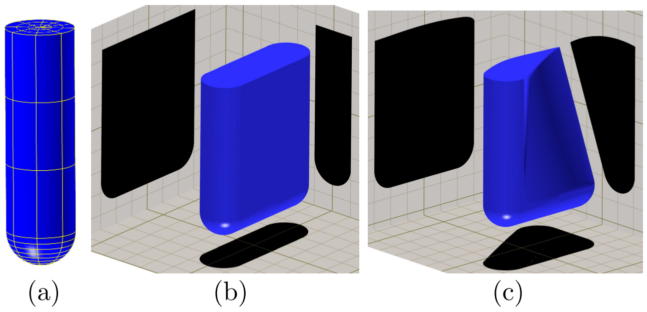


Figure 8: Swept volume of a ball-end tool. (a) The tool is composed of hemispherical, cylindrical and planar surface patches. (b) The swept volume of the tool shown in (a) along a purely translational motion. (c) The swept volume of the tool along a motion having rotational as well as translational components. The shadows of the swept volume are shown in black.

The boundary of the ball-end tool is composed of a hemispherical surface patch at the bottom, a cylindrical surface patch at its flank and a circular, planar face at its top. The top planar face does not play a role in machining and hence may be ignored in the simulation. The hemispherical surface patch is approximated by a tensor-product polynomial B-spline surface [20] of order 4×4 and having 13×4 control-points. This gives an order 3 approximation. As explained in Section 4.1, one may use rational B-splines to get an accurate representation. The cylindrical surface is modeled using a B-spline surface patch of order 4×2 and having 13×2 control-points, with a similar approximation order. Each of these surface patches are used in Equations (2), (4) and (6), in Algorithms 1, 3 and 4, respectively, as explained in Section 5. The ball-end tool is illustrated in Figure 8(a). This tool is swept along two different motions. The first motion, referred to as Sweep1, is purely translational, and the resulting swept volume is shown in Figure 8(b). The second motion, referred to as Sweep2, involves rotation as well as translation. The resulting swept volume is shown in Figure 8(c).

The running times for the ball-end tool for the three algorithmic variants, along the two motions, using two different dexel grid sizes, are given in Table 1. All the times are in seconds. All runs were executed on a 4.2 GHz CPU with 8 GB memory, on a single thread. As can be seen from Table 1, Algorithm 3 is at least an order of magnitude faster than the other two variants. Part of the reason for its efficiency may be that the envelope along multiple dexels is computed from a single univariate slice of the envelope returned by the solver. On the other hand, computing the complete 2-manifold $\partial\mathcal{V}$ in Algorithm 1, is very time consuming. Also, Algorithm 1 takes about the same amount of time for both grid sizes since in this case the solver is invoked only once for each surface patch of the tool, irrespective of the grid size, and the grid processing times are negligible, compared to the algebraic solvers' costs.

A flat-end tool is composed of a circular planar disk at the bottom, a G^1 discontinuity curve bounding the planar disk, a cylindrical surface at the flank and a planar circular face at the top. This is illustrated in Figure 9(a). The surface patches are processed as in the case of a ball-end tool. In addition, the G^1 discontinuity curve is used in Equations (3), (5) and (7) in the three algorithms. The swept volumes of the flat-end tool shown in Figure 9(a) along Sweep1 and Sweep2 are shown in Figure 9(b) and 9(c), re-

Algorithm	Dexel Grid Size	Sweep1	Sweep2
Algorithm1	200 x 200	10.32	13.46
Algorithm3	200 x 200	0.35	0.63
Algorithm4	200 x 200	5.27	22.40
Algorithm1	500 x 500	10.34	13.92
Algorithm3	500 x 500	0.75	1.20
Algorithm4	500 x 500	35.40	134.50

Table 1: Running times of Algorithms 1, 3 and 4 for ball-end tool on the two sweep examples, shown in Figure 8, with different dexel grid sizes. All times are in seconds.

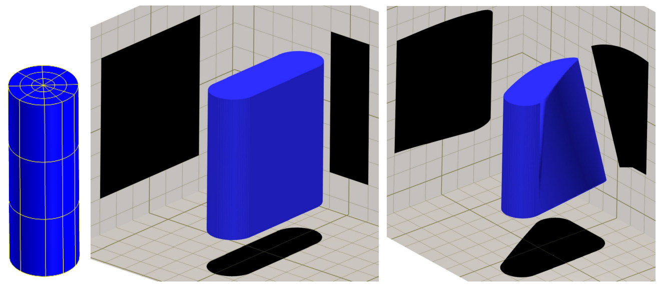


Figure 9: The swept volume of a flat-end tool. (a) The tool is composed of cylindrical and planar surface patches, as well as G^1 discontinuity, circular curves at the bottom and the top. (b) The swept volume of the tool along a translational path. (c) The swept volume of the tool along a motion containing translation as well as rotation. Note here that, due to the rotation, the right side of the swept volume appears thin, being away from the screen. The shadows of the swept volume are shown in black.

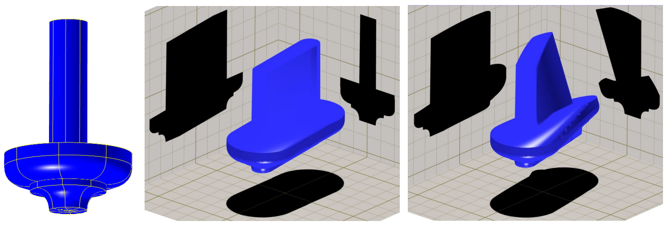


Figure 10: (a) A free-form tool and its swept volumes along a translational motion (b) and motion containing translation as well as rotation (c). The shadows of the swept volume are shown in black.

spectively. The relative running times for Algorithms 1, 3 and 4 for the flat-end tool along Sweep1 and Sweep2 are similar to those of the ball-end tool.

An example of a free-form tool is shown in Figure 10(a) which has one surface patch of order 4×4 having 13×14 control-points. The swept volumes of the tool shown in Figure 10(a) along Sweep1 and Sweep2 are shown in Figures 10(b) and 10(c), respectively. The relative running times of Algorithms 1, 3 and 4 for this tool are again similar to the ball-end tool. Clearly, Algorithm 3 is the algorithm of choice and is used in the rest of this section.

We now present four complete 5-axis CNC machining examples, simulated by the implementation of Algorithm 3. The first example performs the machining of the exterior surface of the body of the Utah teapot [34]. We use only a half of the teapot body for clarity of demonstration. The surface of the body is shown in top-left of Figure 11. A flat-end tool is used to perform a sequence of cuts starting from a rectangular stock. A few snapshots from the simulation, which took about a minute to compute (excluding rendering time), are shown in Figure 11. A dexel grid of size 300×300 is used. The tool-path is designed by adaptive-sampling a set of points along the iso-parametric curves of the surface of the teapot, and the surface normals are used to orient the tool. These positions as well as the normals to the surface at these positions lead to the sequence of SLERP motions as described at the beginning of this section, preserving a three orders of magnitude accuracy.

The second CNC simulation example performs the machining of the interior of the body of the Utah teapot. Again, we use only a half of the teapot surface. A ball-end tool is used for this purpose. The tool path is designed as follows. An offset of the teapot surface is taken at a distance of tool-radius, in the interior direction. A set of points is obtained by adaptive sampling along isoparametric curves of the offset surface, again preserving three orders of magnitude of accuracy. These points give the location of the tool, which is at the center of its hemisphere. The orientation of the tool is obtained by connecting each of these points with a common point at the top opening of the teapot (see Figure 12(b)). A sequence of SLERP motions is obtained from this sequence of positions and orientations of the tool. Hence, the machining is performed "through a point" [18]. A dexel grid size of 300×180 is used. A few frames from the simulation, which took about a minute to compute (excluding rendering time), are shown in Figure 12.

The third example simulates the carving of an impeller blade using a ball-end tool. The sequence of 2700 SLERP motions is pre-specified. A dexel grid size of 600×300 is used and a few frames from the simulation, which took about 8 minutes to compute (excluding rendering), are shown in Figure 13.

The fourth example simulates the 5-axis CNC machining of a wooden hand-railing using the router bit shown in Figure 10(a). A dexel grid of size 300×300 was used for this purpose. A few frames from the simulation, which took about a minute to compute, are shown in Figure 14.

The results shown in Figures 11, 12, 13 and 14 are produced by fitting triangles over the respective dexel grids of the stock being machined. Each square of the grid admits four triangles for each dexel interval, two at the start end-points of the dexel-intervals and two at the end. Further, the normals to the envelope computed at the dexel locations are employed to achieve a smooth rendering.

7. Conclusion

This paper presents an algebraic-based computational framework for 5-axis CNC machining simulation. This is done by a precise algebraic formulation of the envelope condition and solving the same using state-of-the-art constraint solvers. This has the advantage that the returned envelope is topologically accurate, up to a specified tolerance. At the same time, the algebraic approach readily generalizes to sweeping of free-form tools. The effectiveness of our algorithms is demonstrated on several real-life 5-axis machining simulation examples.

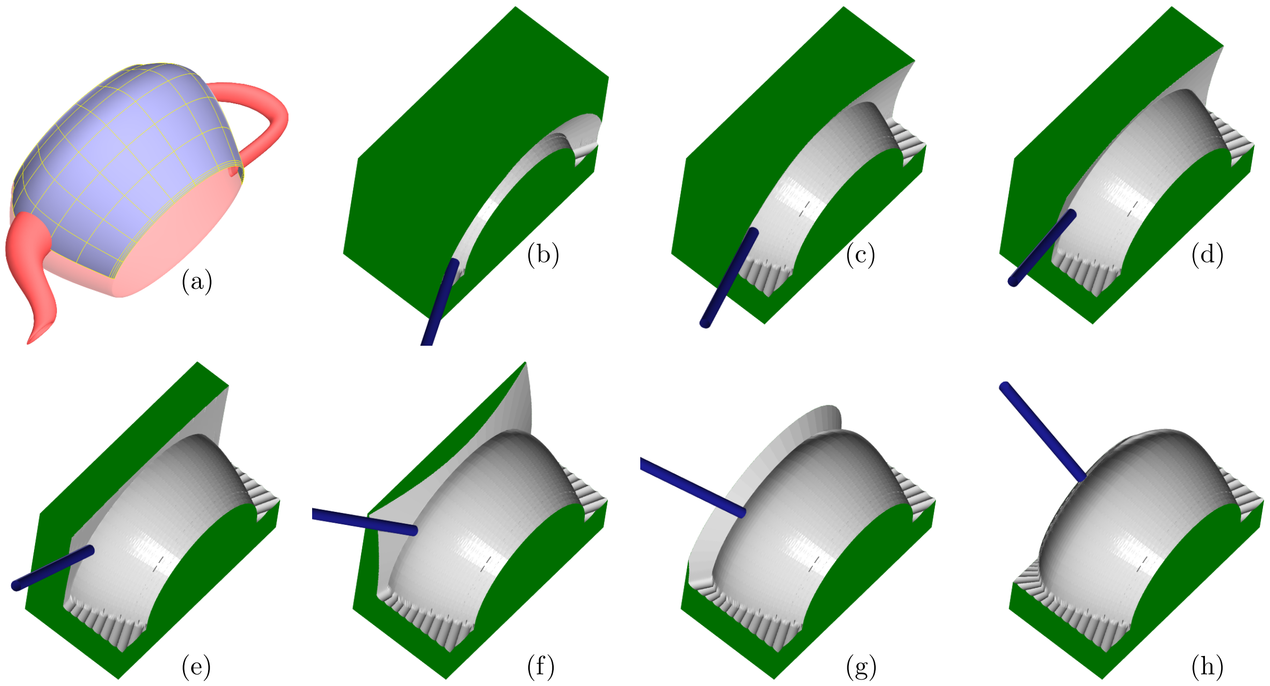


Figure 11: A few frames from the 5-axis CNC machining simulation of the exterior surface of the body of the Utah teapot, using a flat-end cutter. The original surface of the body (half) is shown in (a) by a blue highlight.

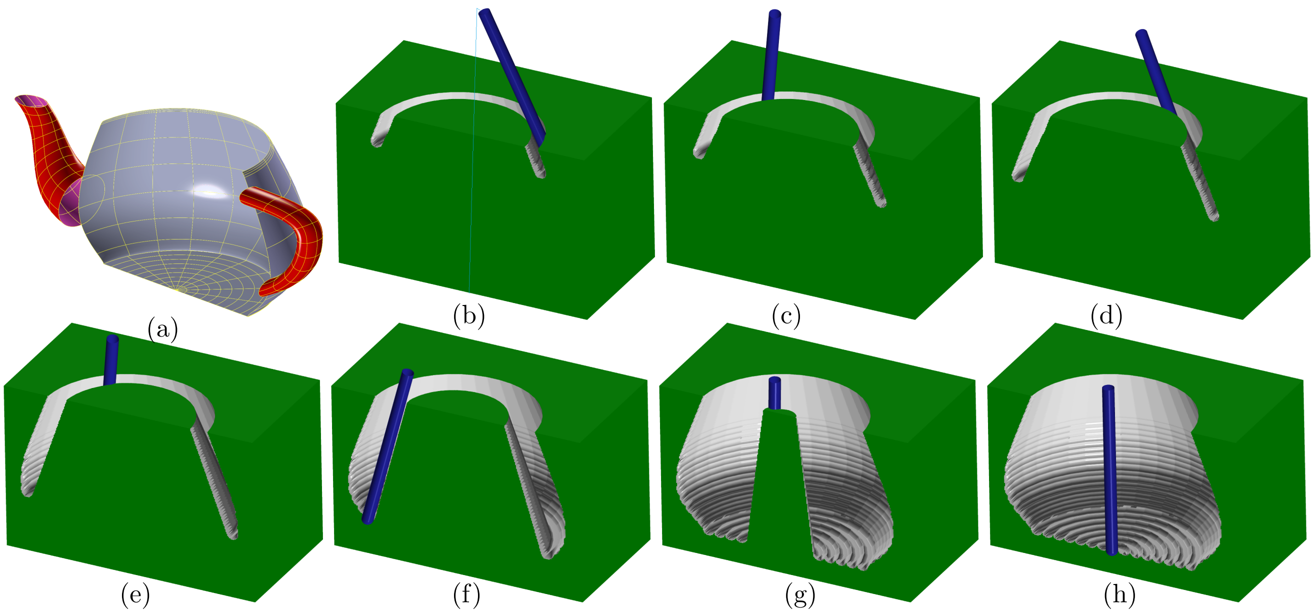


Figure 12: A few frames from the 5-axis CNC machining simulation of the interior surface of the body of the Utah teapot, using a ball-end cutter. The original surface of the body (half) is shown in (a) by a blue highlight. The central axis and the axis of the tool, for the "through point" machining, are showin in cyan in (b).

This work may be extended along several lines. One may choose to employ the solution of the algebraic constraints directly or employ an alternative representation, instead of dexels, which could better exploit the numerical precision inherent to our approach. One may explore the option of choosing the dexels' direction in a way to better approximate the swept volume. Also, dexels along multiple directions may be employed towards the same goal.

Being independent problems, the solution of each plane/line section may be computed in parallel with ease, in order to speed-up the execution. Such a parallel approach can make the presented algorithms usable with reasonable response rates.

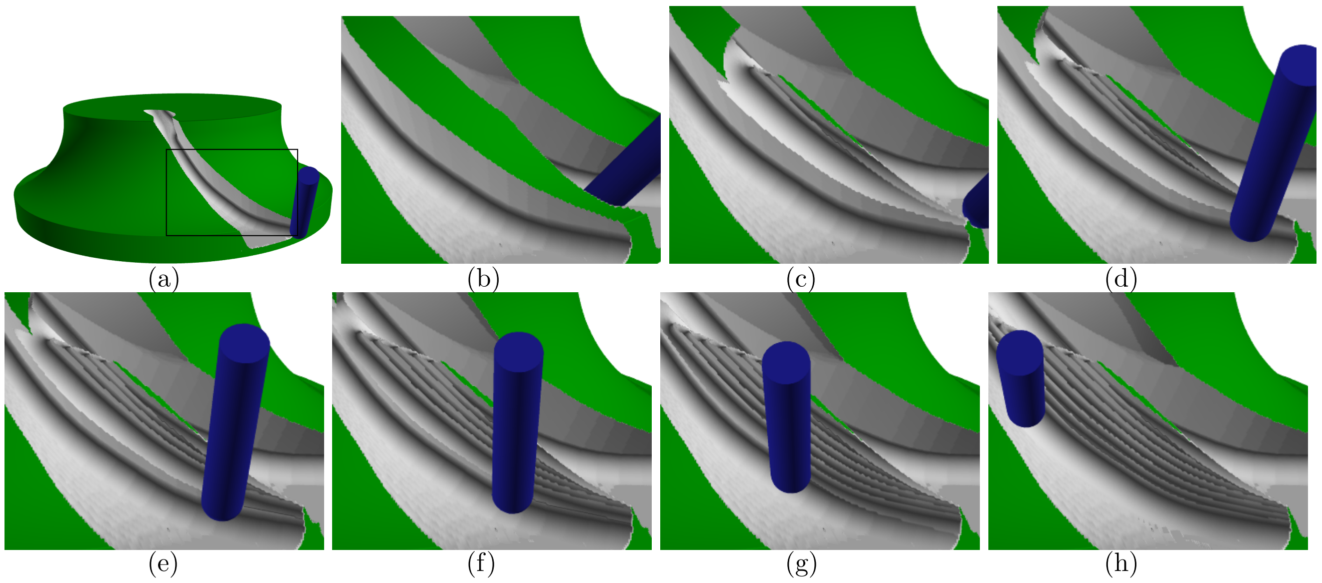


Figure 13: A few frames from 5-axis CNC machining simulation for carving of an impeller blade, using a ball-end cutter. The rectangular area marked in (a) is zoomed into, in the rest of the images.

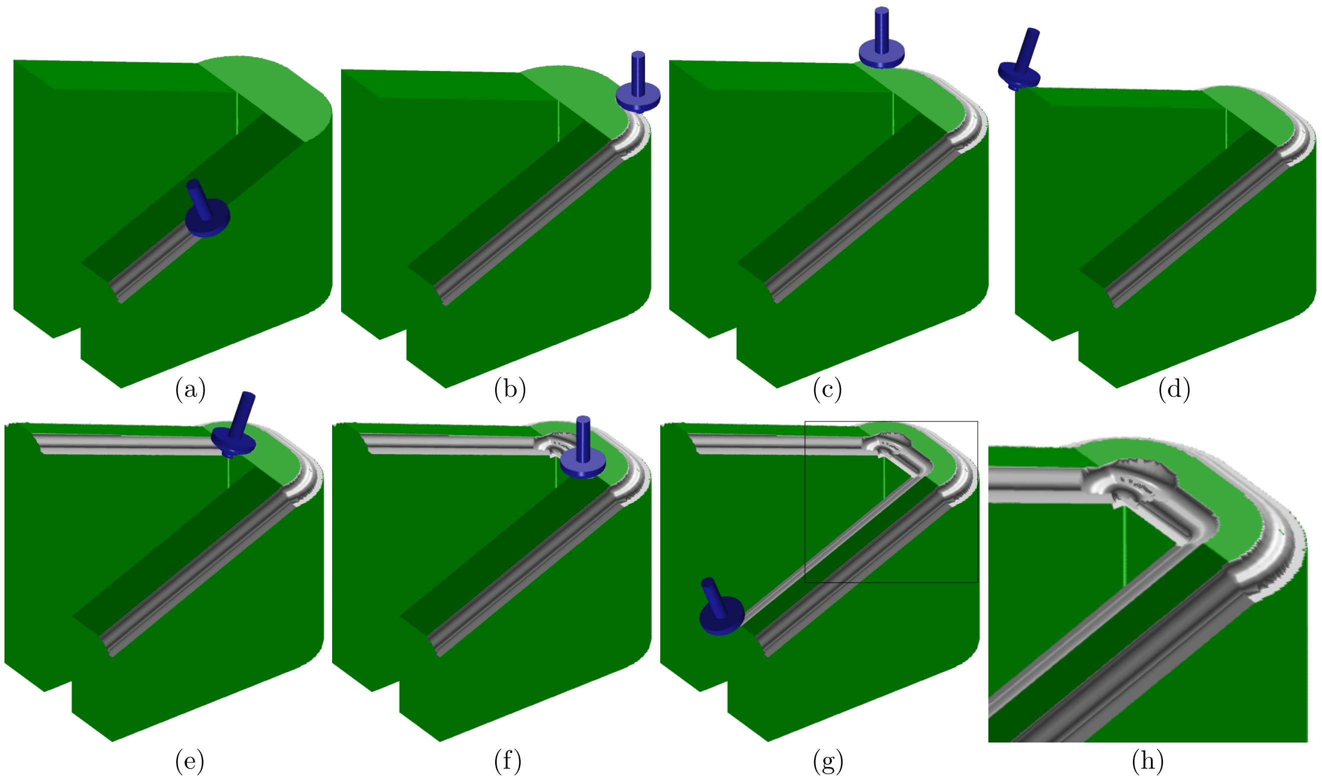


Figure 14: A few frames from 5-axis CNC machining simulation for carving of a wooden hand-railing, using the router bit shown in Figure 10(a). The concavity in the path causes gouging which can be seen in the black rectangle in (g) and is magnified in (h).

8. Acknowledgments

The example shown in Figure 13 was provided by ModuleWorks GmbH, Germany. This work was supported in part by DARPA, under contract HR0011-17-2-0028. All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies, nor should any endorsement be inferred.

References

[1] K. Abdel-Malek, W. Seaman, and H.-J. Yeh. An exact method for nc verification of up to 5-axis machining. In *Proceedings of DETC99: 1999 ASME Design Engineering Technical Conferences*, Las Vegas, 09 1999.

[2] K. Abdel-Malek, W. Seaman, and H.-J. Yeh. Nc verification of up to 5 axis machining processes using manifold stratification. *Journal of manufacturing science and engineering*, 123(1):99–109, 2001.

[3] K. Abdel-Malek and H.-J. Yeh. Geometric representation of the swept volume using jacobian rank-deficiency conditions. *Computer-Aided Design*, 29(6):457–468, 1997.

[4] B. Adsul, J. Machchhar, and M. Sohoni. Local and global analysis of parametric solid sweeps. *Computer Aided Geometric Design*, 31(6):294 – 316, 2014.

[5] J. Andrews and C. H. Squin. Generalized, basis-independent kinematic surface fitting. *Computer-Aided Design*, 45(3):615 – 620, 2013.

[6] E. Aras. Generating cutter swept envelopes in five-axis milling by two-parameter families of spheres. *Computer-Aided Design*, 41(2):95–105, 2009.

[7] M. Barton, G. Elber, and I. Hanniel. Topologically guaranteed univariate solutions of underconstrained polynomial systems via no-loop and single-component tests. *Computer-Aided Design*, 43(8):1035 – 1044, 2011.

[8] M. Barton, H. Pottmann, and J. Wallner. Detection and Reconstruction of Freeform Sweeps. *Computer Graphics Forum*, 2014.

[9] D. Blackmore, M. C. Leu, and L. Wang. The sweep-envelope differential equation algorithm and its application to nc machining verification. *Computer-Aided Design*, 29(9):629–637, 1997.

[10] P. Bo, M. Barto, D. Plakhotnik, and H. Pottmann. Towards efficient 5-axis flank {CNC} machining of free-form surfaces via fitting envelopes of surfaces of revolution. *Computer-Aided Design*, 79:1 – 11, 2016.

[11] C.-J. Chiou and Y.-S. Lee. Swept surface determination for five-axis numerical control machining. *International Journal of Machine Tools and Manufacture*, 42(14):1497 – 1507, 2002.

[12] Y. C. Chung, J. W. Park, H. Shin, and B. K. Choi. Modeling the surface swept by a generalized cutter for nc verification. *Computer-Aided Design*, 30(8):587–594, 1998.

[13] CMT. Industrial router bits. `"http://www.cmtutensili.com/show_items.asp?pars=RB~~2"`.

[14] E. Cohen, R. F. Riesenfeld, and G. Elber. *Geometric Modeling with Splines, An Introduction.* A K Peters, 2001.

[15] S. Du, T. Surmann, O. Webber, and K. Weinert. Formulating swept profiles for five-axis tool motions. *International Journal of Machine Tools and Manufacture*, 45(7):849–861, 2005.

[16] G. Elber. Irit modeling environment. `"http://www.cs.technion.ac.il/~irit/"`, January 2015.

[17] G. Elber and E. Cohen. *Hybrid Symbolic and Numeric Operators as Tools for Analysis of Freeform Surfaces*, pages 275–286. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[18] G. Elber and E. Cohen. *A Unified Approach to Accessibility in 5-axis Freeform Milling Environments*, pages 376–386. Springer US, Boston, MA, 1999.

[19] H. Erdim and H. T. Ilies. Classifying points for sweeping solids. *Computer-Aided Design*, 40(9):987–998, 2008.

[20] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture.* Halsted Press, 1979.

[21] G. Glaeser and E. Gröller. Efficient volume-generation during the simulation of nc-milling. In *Mathematical Visualization*, pages 89–106. Springer, 1998.

[22] Z. Hou, F.-L. Krause, F. Yan, and J. Wu. A new compressed voxel-based method for swept volume modeling of drum cutter and its application. In *2006 7th International Conference on Computer-Aided Industrial Design and Conceptual Design*, pages 1–5. IEEE, 2006.

[23] Y. Jung, B. So, K. Lee, and S. Hwang. Swept volume with self-intersection for five-axis ball-end milling. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 217(8):1173–1178, 2003.

[24] S. W. Lee and A. Nestler. Complete swept volume generation, part i: Swept volume of a piecewise c 1-continuous cutter at five-axis milling via gauss map. *Computer-Aided Design*, 43(4):427–441, 2011.

[25] S. W. Lee and A. Nestler. Complete swept volume generationpart ii: Nc simulation of self-penetration via comprehensive analysis of envelope profiles. *Computer-Aided Design*, 43(4):442–456, 2011.

[26] M. C. Leu, L. Wang, and D. Blackmore. A verification program for 5-axis nc machining with general apt tools. *CIRP Annals-Manufacturing Technology*, 46(1):419–424, 1997.

[27] S. Mann and S. Bedi. Generalization of the imprint method to general surfaces of revolution for nc machining. *Computer-aided design*, 34(5):373–378, 2002.

[28] S. Nelaturi and V. Shapiro. Configuration products and quotients in geometric modeling. *Computer-Aided Design*, 43(7):781–794, July 2011.

[29] M. Peternell, H. Pottmann, T. Steiner, and H. Zhao. Swept volumes. *Computer-Aided Design and Applications*, 2:599–608, 2005.

[30] L. Podshivalov and A. Fischer. Modeling an envelope generated by 3d volumetric nc tool motion. *The International Journal of Advanced Manufacturing Technology*, 38(9-10):949–957, 2008.

[31] D. Roth, S. Bedi, and F. Ismail. Generation of swept volumes of toroidal endmills in five-axis motion using space curves. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 306–311. ACM, 1999.

[32] K. Sheltami, S. Bedi, and F. Ismail. Swept volumes of toroidal cutters using generating curves. *International Journal of Machine Tools and Manufacture*, 38(7):855–870, 1998.

[33] A. Sullivan, H. Erdim, R. N. Perry, and S. F. Frisken. High accuracy nc milling simulation using composite adaptively sampled distance fields. *Computer-Aided Design*, 44(6):522–536, June 2012.

[34] A. Torrence. Martin newell's original teapot. In *ACM SIGGRAPH 2006 Teapot Copyright Restrictions Prevent ACM from Providing the Full Text for the Teapot Exhibits*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[35] T. Van Hook. Real-time shaded NC milling display. *SIGGRAPH Comput. Graph.*, 20(4):15–20, Aug. 1986.

[36] J. Wallner and Q. Yang. Swept volumes of many poses. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[37] L. Wang. An nˆ 2logn algorithm for generating swept solids in nc verification. *INTERNATIONAL JOURNAL OF FLEXIBLE AUTOMATION AND INTEGRATED MANUFACTURING*, 7(3/4):185–208, 1999.

[38] W. Wang and K. Wang. Geometric modeling for swept volume of moving solids. *IEEE Computer Graphics and Applications*, 12(6):8–17, 1986.

[39] K. Weinert, S. Du, P. Damm, and M. Stautner. Swept volume generation for the simulation of machining processes. *International Journal of Machine Tools and Manufacture*, 44(6):617–628, 2004.

[40] J. Xia and Q. Ge. Kinematic approximation of ruled surfaces using nurbs motions of a cylindrical cutter. In *Proc. 2000 ASME Design Automation Conference, Baltimore, MD, Paper No. DETC2000/DAC-14280*, 2000.

[41] J. Xia and Q. Ge. On the exact representation of the boundary surfaces of the swept volume of a cylinder undergoing rational bézier and b-spline motions. *Journal of Mechanical Design*, 123(2):261–265, 2001.

[42] J. Xia, Q. J. Ge, and A. Purwar. On the exact computation of the swept surface of a cylindrical surface undergoing two-parameter rational bézier motions. *International Journal of Mechatronics and Manufacturing Systems*, 4(3-4):356–369, 2000.

[43] J. Yang and K. Abdel-Malek. Verification of nc machining processes using swept volumes. *The International Journal of Advanced Manufacturing Technology*, 28(1-2):82–91, 2006.

[44] L. Zhu, H. Ding, and Y. Xiong. Simultaneous optimization of tool path and shape for five-axis flank milling. *Computer-Aided Design*, 44(12):1229 – 1234, 2012.

[45] L. Zhu, G. Zheng, and H. Ding. Formulating the swept envelope of rotary cutter undergoing general spatial motion for multi-axis nc machining. *International Journal of Machine Tools and Manufacture*, 49(2):199–202, 2009.