Using Curvature Bounds Towards Collision Free 5-Axis Tool-Paths

Ben Ezair, Gershon Elber

Computer Science Dept., Technion,Israel Institute of Technology, Haifa, Israel

Abstract

This paper presents how freeform surface properties can be estimated through bounding, rather than sampling. Bounding surface positions using the control mesh, and bounding surface normals using normal cones, are both well known procedures. In this paper, we add to these, a procedure to bound the normal curvature values of a surface.

We then show how collision free 5-axis tool-paths for CNC machining, using any convex shaped tool, can be generated using the normal curvature bounds of a surface, without calculating surface offsets. As the tool-paths are generated using conservative bounds on the surface, and not sampled points, the tool-paths can be both globally optimized and globally guaranteed to be collision free. Simulation results validating our approach are also presented.

Keywords: Collision Avoidance, Gouging, Tool orientation, Interference detection.

1. Introduction

Computer numerical control (CNC) machining, is a widely used form of subtractive manufacturing. In CNC machining, a target model, usually expressed as a set of boundary surfaces, is manufactured from some initial model (stock) by removing excess material using a machining tool. In this paper, we present methods that can be adapted to any convex machining tool, and use two types of tools as examples:

Definition 1.1. *A ball-end tool has a cylindrical shank, that ends in a* tip*: a hemisphere of the same radius. A flat-end tool is made of a cylindrical shank, with the bottom disc of the cylinder being the* tip*.*

An important aspect of CNC machining, is the generation of collision free (valid) tool-paths. Collision (or gouging) free tool-paths are those in which the tool does not remove material that should remain as part of the target model. Generating collision free tool-paths is especially challenging in 5-axis machining, where both the tool position and orientation change along the tool-path. Collisions can be divided into two main types:

1 Local, in which the tip of the tool gouges the target model (or the CNC machine, etc).

2 Global, when the shank of the tool gouges the target model (or the CNC machine, etc).

In this paper, we show how bounds, computed for the normal curvatures of a given model surface, can be used to generate collision free 5-axis tool-paths for convex tools. The main contributions of this paper are:

1 A method to generate tight bounds for the normal curvatures of a whole surface.

2 A method to generate globally verified valid 5-axis tool-paths for convex tools.

3 A collision avoidance strategy that enables global optimizations of the tool-path, without compromising its validity.

The rest of the paper is organized as follows: in Section 2, we discuss some of the relevant previous work. We lay down the theoretical background for bounding the normal curvature values of a surface in Section 3. Section 4 presents how we apply these normal curvature bounds in an algorithm that generates a globally collision free tool-path, for a flat-end or a ball-end tool. We present our experimental results, which include simulations validating our approach in Section 5. We mention some avenues for future research in Section 6, and conclude in Section 7.

2. Previous Work

The concepts introduced in this work relate to local collisions in CNC machining, and so in this section we focus on other research efforts that deal with local collision avoidance. A more thorough (but not very recent) review of CNC machining research, in general, can be found in [1].

Perhaps the greatest challenge in local collision avoidance stems from the fact that the tool has to make contact with the target model as part of the machining process. This means collision avoidance algorithms must allow tangential contact between the tool and the target model, but still ensure that the tool

does not gouge into the target model. Mathematically, we can say gouging occurs when the intersection of the tool and target model has a non-zero volume.

One way of addressing local collisions is by using offsets [2]. For ball-end tools, globally offsetting all model boundaries in their surface normal direction, by the tool radius, allows the tool to be replaced by a ray, and the tip of the tool with a singular point. However, applying this method to other types of tools (e.g. flat-end, toroidal), is difficult, as in this case every combination of tool orientation (and surface normal) would require a different offset surface. Additionally, the global offset operation can create models that contain singularities and self intersections.

Several local collision avoidance methods operate by matching the second order (curvature) or higher order differential behavior of the tool, to that of the machined surface. Matching the normal curvatures of the surface and the tool, would usually reduce gouging around the contact point, but would not eliminate it [3]. Curvature matching also optimizes the amount of removed material and resulting surface finish for flat-end tools. The principal axis method [4] as well as earlier work like [5], compute the optimal tilt for a given tool, that sets the normal curvature of the tool to best match that of the surface at the contact point. The rolling ball method [6], and the arc intersect method [7] operate similarly, but match the second order differential behavior of a sphere bounding the tip of the tool and an approximated second order differential behavior of the surface, based on a number of sampled points in the vicinity of the contact point. The method described in [3] matches higher order behavior, namely hyper-osculating circles, which indicate a greater level of compatibility between the surface and the tool. This method ensures no gouging occurs in some small neighborhood around the contact point.

In flank machining, the aim is for the tool and surface behavior to be matched along the entire length of the tool rather than individual points [8]. Flank machining is more limited, when compared to machining using the tip of the tool, in the areas of the model that it can be applied to. On the other hand, flank milling can achieve a better surface finish in these areas. This is because in flank milling it is possible to effectively change the shape of the contact profile between the tool and the surface [9], and achieve a much better matching between the surface and tool.

Another approach used for local collision avoidance is the multi-point method [10]. This approach operates on concave surface patches, trying to find tool configurations (position and orientation), that are as close as possible to having two tangential contact points between the tool and the surface. Using such configurations, the tool can be transitioned between cutter contact (CC) points while minimizing overall machining errors.

Apart from the method based on offsets, all of the above described methods rely on sampling of the tool-path and the model surface. Sampling based methods can only test for collisions at a finite set of CC points, and must operate under the assumption that no collisions occur when transitioning between sampled points. In this paper, we propose a curvature matching method based on bounding the surface normal curvatures rather than sampling them. By employing bounds on the normal curvatures of the surface (as well as normal cones and bounding boxes), we can conservatively test if a tool-path is collision free in a whole neighborhood (sub-surface). By testing for collisions in a sequence of adjacent neighborhoods, we can guarantee for an entire CC curve that the resulting tool-path, is globally collision free.

3. Calculation of Normal Curvature Bounds

Given a polynomial or rational regular C^2 surface $S(u,v) \in \mathbb{R}^3$, its normal surface $\bar{n}(u,v) = \frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$, and its *unit* normal surface $\hat{n}(u,v) = \frac{\bar{n}(u,v)}{\|\bar{n}(u,v)\|}$, we recall the following terms of the first and second fundamental forms (following [11] or similar):

$$\begin{aligned} E &= \left(\frac{\partial S}{\partial u}\right)^2, \quad F = \frac{\partial S}{\partial u}\frac{\partial S}{\partial v}, \quad G = \left(\frac{\partial S}{\partial v}\right)^2, \\ L &= \frac{\partial^2 S}{\partial u^2}\hat{n}, \quad M = \frac{\partial^2 S}{\partial u \partial v}\hat{n}, \quad N = \frac{\partial^2 S}{\partial v^2}\hat{n}. \end{aligned} \tag{1}$$

The principal curvatures at a point $p = S(u_0, v_0)$ would be the roots of the following quadratic equation for the variable κ, evaluated at (u_0, v_0):

$$\left(EG - F^2\right)\kappa^2 - (GL + EN - 2FM)\,\kappa - \left(LN - M^2\right) = 0\,. \tag{2}$$

In order to establish bounds on the maximal and minimal normal curvature values for the whole surface, we place bounds on all the above factors. For E, F, G, this is trivial as they are all products of (piecewise) polynomial or rational surface derivatives, and so are (piecewise) polynomial or rational scalar surfaces, with values that are bounded by the values of their control meshes. L, M, N, are also scalar surfaces, but are not (in general) polynomial or rational. L, M, N, are projections of the second partial derivatives of the surface on the *unit* normal, $\hat{n}$. As such, the values of L, M, N can not be bound in the same way.

To obtain maximum and minimum values for L, M, N, we must go one step back, and find the maximal and minimal values for the dot product of the unit normal of the surface with the second partial derivatives of the surface.

The unit normal (pointing outward in this paper) of the surface can be bound in a normal cone of the surface [12, 13]:

Definition 3.1. *Given a regular surface $S(u,v) \in \mathbb{R}^3$, all normal directions of S at all points $p \in S(u,v)$ can be bound inside a (circular) cone with a unit axis $\hat{d} \in S^2$ and angle $\alpha \in \mathbb{R}$. This cone, $C(\hat{d}, \alpha)$, is denoted the* normal cone *of S.*

$\bar{n}(u,v)$ is a rational surface as we assume $S(u,v)$ is a rational surface. Therefore, calculating the normal cone can be done following [14], using the control mesh of the normal surface $\bar{n}(u,v)$. Then, if $\bar{p}_{ij}$ is a control vector from the control mesh of a second partial derivative surface of S, and $C(\hat{d}, \alpha)$ is the normal cone of S, we can calculate the bounds on the dot product between $\bar{p}_{ij}$ and any unit direction $\hat{c} \in C$. We do this by placing

bounds on the maximal angle between $\bar{p}_{ij}$ and $\hat{c} \in C$. These bounds are based on the angle between $\hat{d}$ and $\hat{p}_{ij} = \frac{\bar{p}_{ij}}{\|\bar{p}_{ij}\|}$ as well as the angle α:

$$\begin{aligned} \langle \bar{p}_{ij}, \hat{c} \rangle &\geq \|\bar{p}_{ij}\| \cos\Big(min\Big(\arccos\langle \hat{p}_{ij}, \hat{d}\rangle + \alpha, \pi\Big)\Big), \\ \langle \bar{p}_{ij}, \hat{c} \rangle &\leq \|\bar{p}_{ij}\| \cos\Big(max\Big(\arccos\langle \hat{p}_{ij}, \hat{d}\rangle - \alpha, 0\Big)\Big). \end{aligned} \tag{3}$$

The second partial derivatives of the surface are all polynomial or rational vector surfaces, and their values are affine combinations of the control vectors in their control meshes. Given the nature of affine combinations, the dot product between a second partial derivative of S and a unit normal of S, is bounded by the maximal and minimal values in Equation (3) when applied to all control vectors in the relevant control mesh, generating bounds for L, M, N.

Given bounds on all factors in Equation (2), interval arithmetic [15] can be applied to bound any solution to Equation (2), κ. The result will typically be two intervals: one for the value of the maximal principal curvature, and one for the value of the minimal principal curvature. Note that the relation between the two intervals will depend on the values of the two principal curvatures of the surface. The two intervals may be disjoint for a surface with two very different principal curvature values, or overlapping, for a surface with principal curvature values that are close to one another.

4. Collision Avoidance in 5-Axis Machining

Let $\mathcal{M}$ be our our target model, with a closed boundary that is expressed as a set of regular C^2 surfaces, O. Further, let $\mathcal{S} = \{S_1, \ldots S_n\}$, be a subdivision of the boundary surfaces of $\mathcal{M}$ into smaller sub-surfaces. We wish to generate a collision free 5-axis tool-path for moving a given tool T along a given cutter contact (CC) curve $C(t)$, $t \in [t_0, t_1]$, on $\mathcal{S}$.

To generate a locally as well as globally collision free 5-axis tool-path we use a configuration space (or C-space) based method. For more information on the concept of C-space see [16]. The C-space we use is three dimensional: t parametrizes the motion of the contact point between the tool and the model, along $C(t)$, while Θ, and Φ parametrize the orientation of the tool (a direction on the unit sphere, S^2, along the axis of the shank and away from the tip). Each single configuration, a (t, Θ, Φ) point in the C-space, fully encodes the contact point and orientation of the tool. We divide the C-space into small 3D sub-domains of the form: $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, to make collision testing more manageable. Each sub-domain, that contains an infinite number of individual configurations, is tested to see whether or not all of the included configurations are collision free. The sub-domains that can be guaranteed to contain only collision free configurations are marked as valid. The required collision free tool-path is generated by finding the optimal path through the C-space (from plane $t = t_0$ to plane $t = t_1$) that only passes through valid sub-domains, and thus contains only collision free configurations.

In the following sections, we explain more thoroughly some of the steps in the above broad outline of our solution. In Sections 4.1, 4.2, and 4.3, we explain how to conservatively test a given sub-domain, $[t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, in the C-space for global and local collisions. Then, Section 4.4 explains how the C-space and then the optimal path through the C-space are generated.

4.1. Local Collision Detection

In this section, we aim to present a method for conservatively testing all surfaces in $\mathcal{S}$ for possible collisions with the tip of the tool, for a given sub-domain of the C-space, $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$. For surfaces in $\mathcal{S}$ that are far enough away from the tip, most collision detection tests [17], will be sufficient. As stated before, the main difficulty, is eliminating possible collisions with surfaces that are close to, or in contact with the tip. We denote by $\tilde{\mathcal{S}} \subset \mathcal{S}$ the set of surfaces in $\mathcal{S}$ that can not be eliminated as possible collisions using simple (for example bounding box) collision tests. If we can somehow guarantee none of them are penetrated by the tip of the tool, then we can ensure no local collisions occur in the sub-domain.

Intuitively, the method we use to achieve this guarantee, is based on ensuring that a sphere that bounds the tip of the tool, and is tangent to the surface at the contact point is more curved (has a higher normal curvature) than an entire surface neighborhood around the contact point. For example, as shown in Figure 1, if the blue sphere has a higher normal curvature than the surface in a large enough neighborhood around the contact point p, than anything inside the sphere does not penetrate the surface. This specifically includes the tip of the tool, as well as its circle shaped cutting profile.

More formally, the principles we use to achieve such guarantees are those presented in [18]. To keep this paper self contained we reiterate here the main result from [18]:

Theorem 4.1. *Let S be a surface, which is represented as the graph surface (or a height-field) $z = f(x, y)$ of a compactly supported, C^0, piecewise C^2 function $f : \mathbb{R}^2 \to \mathbb{R}$. If S is everywhere locally millable with a strictly convex C^0, piecewise C^2, tool T with z-parallel axis, then S is globally millable with T.*

Proof. See proof of Theorem 3 in [18]. □

In the above theorem, a surface S is locally millable with T at a contact point $p \in S \cap T$, if at p, the interior of the signed Dupin indicatrix of S (elliptic or hyperbolic) contains the signed Dupin indicatrix (see [18]) of a convex T (always elliptic) [11]. The term 'millable' as used above is equivalent to the term 'collision free', with respect to both the shank and the tip, when referring to a specific surface and tool configuration, defined by a contact point and tool orientation. Note that a point with a concave C^1 discontinuity (that has unbounded normal curvature) is not locally millable by any smooth tool.

We can now add the following:

Corollary 4.2. *Assume a ball-end tool $\mathring{T}$, aligned to the z axis, large enough to contain the tip of the original tool, T, and*

shares with T a tangential contact point with the model $\mathcal{M}$. Theorem 4.1 offers conditions that ensure $\mathring{T}$ does not collide with the surfaces in $\tilde{\mathcal{S}}$. If $\mathring{T}$ does not collide with $\tilde{\mathcal{S}}$, then the tip of the original tool T does not collide with $\tilde{\mathcal{S}}$ as well.

We can specify a containing tool $\mathring{T}$, as described in the above corollary, for a tip with any convex geometry that is in tangential contact with the model. The curvature of the tip of the tool, $\mathring{T}$, is defined as the *effective curvature* of the original tool T:

Definition 4.1. *Let S be a C^2 surface, and T a tool that has a tangential contact with S at a point $p \in S$. The* effective curvature *of the tool is the normal curvature of the smallest sphere that has a tangential contact with S at p, and fully contains the tip of the tool. Refer to Figure 1 for an illustration of the above bounding sphere for the tip of a flat-end tool.*

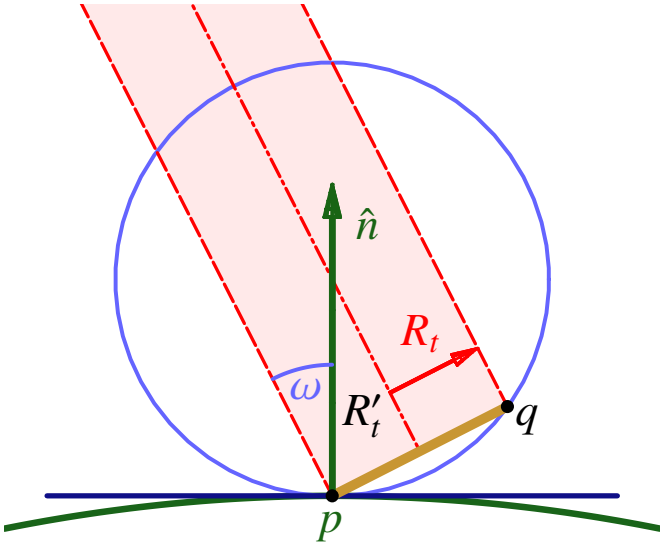

Figure 1: Calculating the effective curvature of flat-end tool, for a single specific orientation. The shank of the tool is shown in light red, the tip is the bottom disc. The normal, $\hat{n}$, at the contact point, p, is shown in green and the tangent plane at p is shown in dark blue. A sphere that bounds the tip of the tool and is tangent to the surface at p determines the effective curvature of the tool, and is shown in blue.

We will use the following Lemma to ensure a set of surface points can be represented as a graph surface (or a height-field) $z = f(x, y)$:

Lemma 4.3. *Let $\mathcal{S}$ be a set of C^2 surfaces, representing a closed 2-manifold boundary. V_c is a convex closed 2-manifold boundary enclosing a volume V_c°. $\mathcal{S}_I = \mathcal{S} \cap V_c^\circ$ is the set of all points on the surfaces $\mathcal{S}$ that are also inside V_c°. If the surface normals (that point outward) at all points in $\mathcal{S}_I$ have a positive dot product with the $\hat{z}$ direction, then $\mathcal{S}_I$ can be represented as a graph surface $z = f(x, y)$.*

Proof. Assume that $\mathcal{S}_I$ can not be represented as a graph. This means that there are two points $p_1, p_2 \in \mathcal{S}_I$, that have the same (x, y) coordinate. The line segment $\overline{p_1 p_2}$ is aligned with the direction $\hat{z}$. Without loss of generality assume that $\langle p_1, \hat{z} \rangle < \langle p_2, \hat{z} \rangle$ (an equality would indicate $\mathcal{S}$, the closed boundary of the model, is self intersecting).

Consider the volume enclosed by $\mathcal{S}$, $\mathcal{S}^\circ$, and V_c°. Their intersection, $V_I^\circ = \mathcal{S}^\circ \cap V_c^\circ$, is a (possibly disjoint) volume as well. By assumption the surface normal at p_1 has a positive dot product with $\hat{z}$, and there must be a small neighborhood of $\overline{p_1 p_2}$ around p_1 that is outside V_I°. Similarly, because of the relation between the surface normal at p_2 and $\hat{z}$, there must be a small neighborhood of $\overline{p_1 p_2}$ around p_2 that is inside V_I°. Therefore, $\overline{p_1 p_2}$ must transition from being outside V_I° to being inside V_I°.

Such a transition can only happen at a surface point on the boundary of V_I° (and inside V_c°) at which the dot product of the surface normal and $\hat{z}$ is negative. By assumption, no such point exists on $\mathcal{S}_I$. This means no such transition exist, and our assumption that $\mathcal{S}_I$ can not be represented as a graph, created a contradiction. □

Figure 2, illustrates the main argument behind the above proof: the existence of two points with upward ($\hat{z}$) pointing normals and the same xy location necessitates the existence of (at least one) point with a downward pointing normal on the line between them.

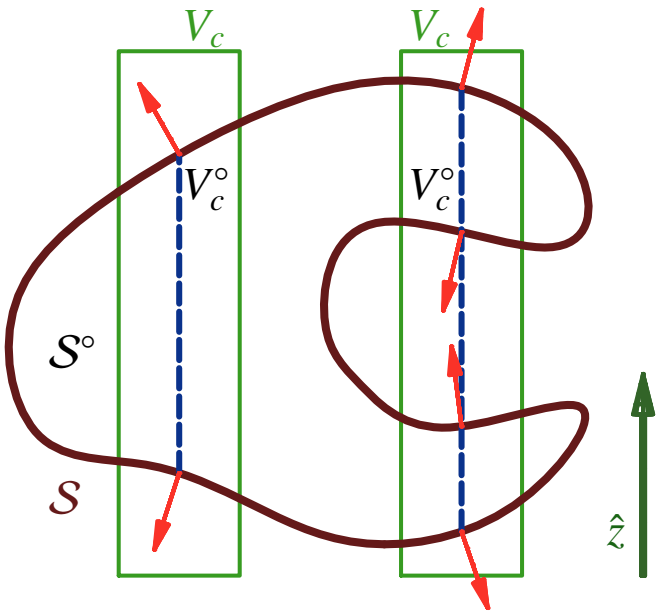

Figure 2: Illustration of the proof for Lemma 4.3. The volume $\mathcal{S}^\circ$ is enclosed inside the surface $\mathcal{S}$ (dark red outline). The intersection of a volume V_c° (for example the two volumes shown as green rectangles) and $\mathcal{S}^\circ$ cannot include two points on $\mathcal{S}$, that have the same x, y coordinate and a surface normal with a positive dot product with $\hat{z}$, without including a third point at the same x, y location with a normal that has a negative dot product with $\hat{z}$.

Algorithm 1 integrates the theoretical principles described above, into a set of steps that can be used to test if a given subdomain in the configuration space is free of local collisions. In other words, Algorithm 1 tests if the tip of a given tool T as it moves along a segment of the CC curve, $C([t_a, t_b])$, with an orientation in the range $[\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f] \in S^2$, gouges a closed

boundary made from a set of surfaces, $\mathcal{S}$, that we assume are all C^2. Algorithm 1 has two main stages. First a bounding volume, V_c° that bounds the tip of the tool as it moves in the given sub-domain is calculated, and surfaces in $\mathcal{S}$ that fall completely outside V_c° are eliminated as possibly colliding with the tip. In the second step the surfaces that do intersect V_c°, are eliminated as possibly colliding with the tip based on Corollary 4.2. Below is a more detailed explanation of Algorithm 1.

Algorithm 1 TestLocalCollsions

Input:

(1) $\mathcal{S} = \{S_1, \ldots S_n\}$, a set of C^2 surfaces;

(2) $C(t)$, a CC curve, a parametric curve on $\mathcal{S}$, $t \in [t_0, t_1]$;

(3) $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, a C-space sub-domain;

(4) T, a convex tool, of a given type and radius;

Output:

(1) $true/false$, no local collision in sub-domain $\mathcal{U}$;

Algorithm:

1: $T_{curvature} := ToolEffectiveCurvature(\mathcal{U}, C(t), \mathcal{S}, T)$;
2: $V_c^\circ := ToolTipConvexBoundingVolume(\mathcal{U}, C(t), \mathcal{S}, T)$;
3: $\tilde{\mathcal{S}} := \{\tilde{S}_1, \ldots, \tilde{S}_m\} = \left\{S_j \,\middle|\, \tilde{S}_j \in \mathcal{S} \textbf{ and } V_c^\circ \cap \tilde{S}_j \neq \varnothing\right\}$;
4: $C(\hat{d}_i, \alpha_i) := NormalCone(\tilde{S}_i)$;
5: $C(\hat{d}, \alpha) := BoundingCone(\bigcup_{i=1}^m C(\hat{d}_i, \alpha_i))$;
6: $Q := EmptyQueue()$;
7: $Visited := \{false, \ldots, false\}$; // A length m Boolean array
8: $Visited[1] := true$;
9: $QueueInsert(Q, \tilde{S}_1)$;
10: **while** $QueueNotEmpty(Q)$ **do**
11: $\quad \tilde{S}_i := QueueRemove(Q)$;
12: $\quad$ **if** $MaxNormalCurvature(\tilde{S}_i) \geq T_{curvature}$ **or** $\arccos\langle \hat{d}_i, \hat{d} \rangle + \alpha_i \geq \frac{\pi}{2}$ **then**
13: $\quad\quad$ return $false$;
14: $\quad$ **end if**
15: $\quad$ **for all** $\left\{\tilde{S}_j \in \tilde{\mathcal{S}} \,\middle|\, \tilde{S}_i \cap \tilde{S}_j \cap V_c^\circ \neq \varnothing\right\}$ **and** $!Visited[j]$ **do**
16: $\quad\quad Visited[j] := true$;
17: $\quad\quad QueueInsert(Q, j)$;
18: $\quad$ **end for**
19: **end while**
20: **if** $Visited[i] = true\,, \forall i \in 1, \ldots, m$ **then**
21: $\quad$ return $true$;
22: **end if**
23: return $false$;

The first step in Line 1 of Algorithm 1 is calculating the minimal *effective curvature* of the tool, for all configurations in $\mathcal{U}$, using $ToolEffectiveCurvature$. The details of this calculation can be found in Appendix A.1. The effective curvature depends on the radius and orientation of the tool, as well as the surface normals along the CC curve.

Lines 2 to 5 in Algorithm 1, initialize the set $\tilde{\mathcal{S}}$, and related information. We use $\tilde{\mathcal{S}}$, rather than the more accurate S_I, to avoid (as much as possible) handling trimmed surfaces in our actual implementation. The 3-manifold convex volume, V_c°, computed using $ToolTipConvexBoundingVolume$, includes the entire volume covered by the tool-tip for any of the configurations in the sub-domain. Full details of this computation can be found in Appendix A.2. This bounding volume depends not only on the radius and orientation of the tool, but also on the CC curve and the surface. The set of surfaces in $\tilde{\mathcal{S}}$, is then found by testing intersections between V_c° and $\mathcal{S}$.

The last operation, in lines 6 to 23 in Algorithm 1, tests if points in $\mathcal{S}_I = \tilde{\mathcal{S}} \cap V_c^\circ$, are collision free, based on the conditions of Theorem 4.1. If the conditions are met, then no collision occurs. Otherwise, we conservatively assume a collision does occur. These tests were all integrated into a breadth first search (BFS) [19] over $\mathcal{S}_I$.

The first requirement for Theorem 4.1, is that the surfaces in $\mathcal{S}_I$ are everywhere locally millable. This requirement is fulfilled because the test in line 12 will classify (conservatively) as a possible collision any situation in which the maximal normal curvature value of any relevant surface is greater than the minimal effective curvature value of the tool.

The second requirement in Theorem 4.1, is that the surfaces in $\mathcal{S}_I$ can be represented as a C^0 and piecewise C^2, graph surface $z = f(x, y)$. The piecewise C^2 requirement is an assumption (or precondition) of Algorithm 1. The C^0 continuity of $\mathcal{S}_I$ is tested by a BFS. The BFS ensures all surfaces in $\mathcal{S}_I$ are connected, and so form a single C^0 neighborhood inside V_c°. In general, $\mathcal{S}_I$ would not be a $z = f(x, y)$ graph for the original z axis. However, we can often select an (x, y, z) coordinate system, in which $\mathcal{S}_I$ can be represented as a graph. The direction $\hat{d}$, the axis of the global bounding cone for $\tilde{\mathcal{S}}$, computed in Line 5 in Algorithm 1, is a natural choice for the $\hat{z}$ direction in the new coordinate system. The x and y directions can be any two directions orthogonal to both $\hat{d}$ and each other. If the test in Line 12 indicates all surfaces in $\tilde{\mathcal{S}}$ face toward some direction $\hat{d}$, then in this new coordinate system (where $\hat{d} = \hat{z}$) the points in $\mathcal{S}_I$ can be represented by a graph $z = f(x, y)$, as shown in Lemma 4.3.

We implemented Algorithm 1 for ball-end and flat-end tools. However, it is applicable to other convex tool types like bull-nosed (toroidal) and various types of tools with conic shanks, with some adaptations. The adaptations needed are the implementation of the functions $ToolTipConvexBoundingVolume$, and $ToolEffectiveCurvature$ (described in Appendix A.1, and Appendix A.2) for these other types of tools.

4.2. Global Collision Detection

To confirm no global collisions occur for a given sub-domain of the C-space $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, we use an adaptation of the approach presented in [20]. Let V_m be a conical frustum that bounds the shank of the tool for all configurations in $\mathcal{U}$. For a cylindrical or conical shank, and sub-domain $\mathcal{U}$, no global collisions occur if no collisions between the model $\mathcal{M}$ and the shank occur inside V_m. The algorithm in [20] already shows how we can ensure no collisions occur inside a rectangular frustum between a tool's shank and a model. The adaptation to a conical frustum is straightforward. Note that any

global collision avoidance algorithm can be used to preform the above task. The only requirement is that it can ensure no global collisions occur in the given sub-domain.

4.3. Balancing Global and Local Collision Detection

For flat-end tools, the shank is in contact with the model during the machining process. This may force the shank to be close to surfaces that are back-facing in relation to the tool direction. Such a situation often occurs near sharp corners. Unfortunately, the method we use to eliminate global collisions (adapted from [20]) is overly conservative when the shank is close to surfaces that are back-facing with respect to the direction of the tool. Figure 3 illustrates such a situation.

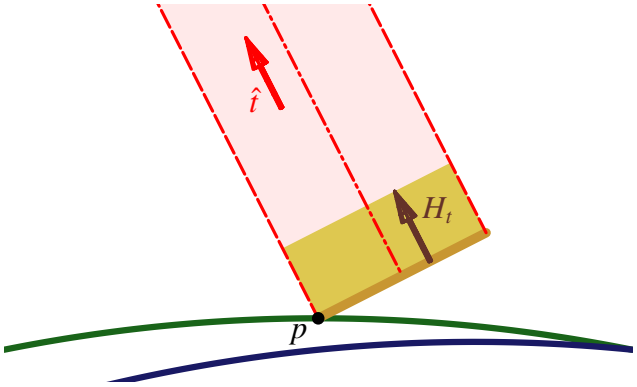

Figure 3: The tool is in contact with the surface in green, but is close to the surface in blue that is back-facing in relation to the tool direction ($\hat{t}$). The method we use to eliminate global collisions (adapted from [20]) is overly conservative in such situations. Extending the tip of the tool to include not only the bottom disk (marked as a thick brown line), but also an extended cylindrical tip (in yellow) will often solve the issue, as the remaining portion of the shank considered for global collisions will be moved sufficiently away from the blue surface.

We would like to avoid the overly conservative behavior described above. Fortunately, we can do so by taking advantage of the cylindrical geometry of flat-end tools. The division between the shank and the tip in flat-end tools can be relaxed, and a cylinder near the tip of the tool can be defined as the new tip. This allows us to define an *extended tip*:

Definition 4.2. *The union of the disc tip of a flat-end tool, with an adjacent small cylindrical section of the shank, is denoted as an* extended tip. *The height of the cylindrical addition to the tip, is $H_t \geq 0$. See also Figure 3.*

The larger the value of H_t the longer the extended tip is, shrinking the other portion covered by the shank. This has two effects: first, it distances the shank from the contact point, reducing the possibility of overly conservative behavior in our global collision detection. Second, it changes the geometry of the tip, meaning the approach presented in Section 4.1 must be applied to a larger geometry. While a larger tip geometry may produce more local collisions, in practice setting H_t to some small (relative to the tool radius) non-zero value, allows for more sub-domains to be recognized as valid. This, in turn, leads to overall better machining solutions.

4.4. C-Space Path Generation

Sections 4.1 and 4.2 explain how a given sub-domain of the C-space, $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, can be guaranteed to be collision free (for both local and global collisions). In this section, we present how the approaches presented in these sections can be combined to produce a tool-path that is globally collision free.

The first task we undertake is the subdivision of the boundary surfaces of the model, O, into as set of smaller sub-surfaces (patches), $\mathcal{S}$. Following the subdivision scheme presented in [20], we recursively subdivide the boundary surfaces of $\mathcal{M}$ until every surface has a bounding sphere with a radius smaller than some predetermined value, r_{max}, and a normal cone with an angular span smaller that some predetermined value, α_{max}. This ensures $\mathcal{S}$ only includes surfaces that are small and flat enough. While we subsequently only test collisions against the patches, these patches are a subdivision of the original model surface, and completely cover the model. This means that in terms of collisions, not colliding with any of the patches guarantees no collisions with the whole work-piece, globally.

This task also includes calculating the maximal normal curvature bound (as described in Section 3) and normal cones of the surfaces in $\mathcal{S}$. Following the requirements of the computations in Section 4.1, the surfaces in $\mathcal{S}$ must be C^2 surfaces. Any C^2 discontinuities in the original model must be located on the boundaries of surfaces in $\mathcal{S}$. However, these C^2 discontinuities should be taken into account when computing the maximal normal curvature bound.

In practice, calculating normal curvature bounds, as described in Section 3, even for the subdivided surfaces in $\mathcal{S}$, may be too inaccurate. To achieve better accuracy we first specify the accuracy goals using a range of normal curvature values $[\kappa_{min}, \kappa_{max}]$ for the tool (for a ball-end tool $\kappa_{min} = \kappa_{max}$), and a desired normal curvature accuracy κ_{res}: the maximal acceptable size of the resulting normal curvature intervals. If the calculated normal curvature intervals do not overlap $[\kappa_{min}, \kappa_{max}]$, or are smaller than κ_{res}, then our normal curvature bounds are accurate enough. Otherwise, we can recursively subdivide the surface (along alternating parametric axes) and calculate bounds for the sub-surfaces. The normal curvature bounds for a surface will be a union of the normal curvature bounds for its sub-surfaces. The recursion is terminated when the bounds are accurate enough (as defined before) or when a preset number of recursions occur. Setting a maximal number of recursions eliminates the possibility of an unbounded number of subdivisions that may occur at singularities (like cusp points).

The second task is the construction of the C-space. Recall that the C-space is three dimensional: t parametrizes the motion of the contact point between the tool and the model, along $C(t)$, while Θ, and Φ parametrize the orientation of the tool (a direction on the unit sphere, S^2). The subdivision of the C-space is achieved by subdividing the full C-space $[t_0, t_1] \times [0, 2\pi] \times [0, \pi]$, into smaller sub-domains. The rectangular orientation domain $[0, 2\pi] \times [0, \pi]$ is divided into a grid with a given bound on its angular resolution $Angle_{max}$. The CC curve $C[t_0, t_1]$ is subdivided at any points where it intersects the boundaries of surfaces in $\mathcal{S}$, while it can also, optionally, be further subdivided as desired.

Both of the above tasks are performed to make the bounds we often use in conservative computations tighter. The smaller the values of $Angle_{max}$, r_{max}, and α_{max} are, the tighter the bounds become. At the limit, when these values approach zero, the conservative computations become exhaustive tests, for every point along the CC curve, and every possible tool orientation.

The third task is recognizing which of the sub-domains of the C-space are collision free and thus valid. This is done by applying the results of Sections 4.1 and 4.2 to each of the sub-domains and marking those without global or local collisions as collision free (valid).

It is necessary, either before or after the above step, to examine sub-domains based on the angles between the surface normals and the tool directions. Performing these tests before the actual collision testing would improve performance, as it would eliminate more costly collision tests for some sub-domains.

For all tool types, it is desired to exclude sub-domains that include configurations in which the surface normal and tool direction have an angle between them that is too large (for example greater than 90°). Excluding these sub-domains would result in a more stable algorithm

For a flat-end tool we also exclude (mark as invalid) any sub-domain that includes a configuration in which the surface normal and tool direction perfectly align. In such a configuration, the flat-end tool's bottom disc is in the surface's tangent plane, and there is no longer a single tool position associated with every (three dimensional) point in the configuration space. Resolving this issue would require either a higher dimensional configuration space, or equivalently, to take the previous configuration into account. To simplify our configuration space traversal, we chose in our solution not to consider these configurations.

At this point, we have a C-space that is divided into sub-domains, each marked as valid or invalid. All that remains is finding an optimal path through the C-space (from the $t = t_0$ plane to the $t = t_1$ plane) that only passes through valid sub-domains, and thus contains only collision free configurations. We do so by converting the C-space into a graph, and employing the same graph search approach as in [20].

The optimal path obviously depends on the costs we assign to the edges of the graph. The costs should be set to optimize the machining process. The costs presented in [20], like penalizing for ball-end tools configurations in which the tool axis and surface normal are too close, to avoid a contact point with zero cutting speed, remain unchanged. We add to them costs specific for flat-end tools. These costs are meant to minimize the angle between the tool axis and surface normal direction (while still avoiding zero angles), which would minimize scallop heights. We would also set costs to make the tool axis remain, as much as possible, in the plane defined by the CC curve (tangent) direction, and the surface normal, at the contact point.

The valid sub-domains of configurations are given as a set of volumes in the C-space, and every configuration can be directly translated to a specific tool position (using $C(t)$) and orientation. We can use this to apply global continuous optimizations to the path, while ensuring the path remains collision free. For example, the path created by the graph algorithm can be smoothed using some global filter, as is shown in [20].

5. Experimental Results

We implemented the algorithm described in Section 4 as a C/C++ single threaded program. All tests ran on an Intel i7-4770 3.4 GHz windows 7 machine. To validate our results, using CNC simulations, we use the machining Verifier Application by ModuleWorks (https://www.moduleworks.com).

Throughout the following experiments, we use a tool with a unit (1) radius, regardless of type. For the normal curvature bounds calculation the maximum number of recursions is set to 8, and the accuracy value is set to $\kappa_{res} = 0.005$. Unless noted otherwise, r_{max} is set to 0.4, $\alpha_{max} = 0.05\pi$, and $Angle_{max} = 0.01\pi$. In all of our experiments we use an extended tip for flat-end tools. The default value for the length of the extended tip, H_t, is set to 0.4. Experimentally, the ratio we found $H_t = 0.4R_t$, gave a good balance between the need to avoid spurious global collision detections, and possibility of additional spurious local collision detection.

The most critical of the above parameters are r_{max} and α_{max}, as they determine the number of patches. Smaller values for r_{max} will create more, but smaller, patches while smaller values for α_{max} will create more, but flatter, patches. There is of course a certain overlap between these values as smaller patches will often be flatter and vice versa. In general, more patches require greater computation times, but also make the conservative bounds used in these computations much tighter, potentially exposing additional valid solutions. Another important parameters is $Angle_{max}$ that determines the number of orientation ranges that are considered. Again, smaller values would mean more orientation ranges, and longer computations times, but tighter bounds that potentially identify more valid tool-paths. Let n_o be the number of orientation ranges and let n_p the number of patches. Then, in our implementation, computation times would depend linearly on the number of orientation ranges, or $O(n_o)$, and will have $O(n_p \log n_p)$ complexity in the number of patches. The linear dependence on n_o is because we test for collisions for each orientation range, while the computation cost per orientation is fixed. $O(n_p \log n_p)$ is due to the requirement for the patches to be ordered in a BVH (bounding volume hierarchy [21]).

The first model we use is designed to test the quality of the achieved curvature matching for a flat-end tool. The model has a large developable area (shaped like a subtraction of a cylinder) with constant ($\kappa_{min} = 0, \kappa_{max} = 0.05$) parabolic normal curvature. The optimal curvature matching would occur if the angle between the surface normal and tool direction would be $\arcsin 0.05 \approx 0.05$ radians, resulting in an effective tool curvature that perfectly matches the normal curvature of the surface. In the generated tool-path, the angle between the tool direction and surface normal is about 0.078738 radians. The difference is smaller than the theoretical limit for the difference in this case (in radians): $Angle_{max} = 0.01\pi \approx 0.0314 > 0.028738 = 0.078738 - 0.05$. The normal curvature bounds calculated for the relevant area are very accurate (less than 0.01 between the

maximal and the minimal values). A still frame from the simulation can be seen in Figure 4. To simulate the machining process we offset the generated tool-path toward the surface (the direction opposite the surface normal) by 0.1. Using this offset a small amount of material would be removed and the machining result can be observed.

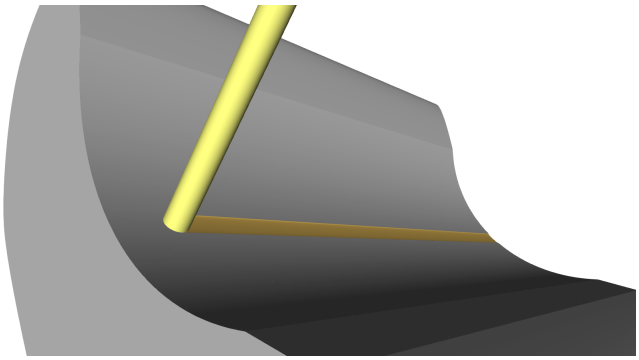

Figure 4: A model with a constant mean curvature (0.05) area being machined.

The second model is designed to examine the interplay between global and local considerations. This model, shown in Figure 5, is created by sweeping a closed curve (shown in green at the bottom of Figure 5) along a 3D knot curve. We will refer to this model as the *knot model*.

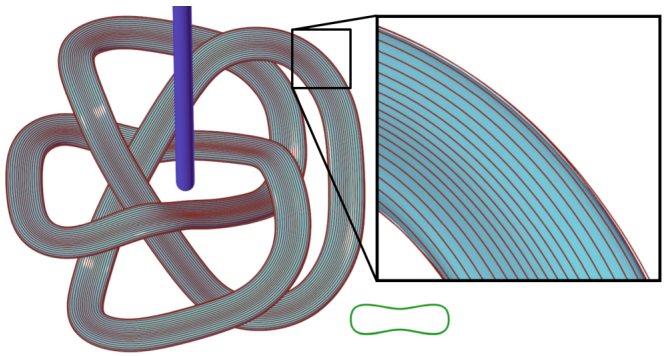

Figure 5: The knot model, with a radius one ball-end tool. The cross-section of the model is shown in green. The CC curve used in our experiments is also shown (in red).

In the following scenario, the entire surface of the knot model is machined in a simulation, ignoring fixture considerations. The CC curve we use goes along the length of the model several times, shifting a bit each time in the perpendicular direction, to cover the whole surface. We do not make modifications to the CC curve (or the traveling direction along it) in any way, even though such modifications may reduce the number of extractions and re-insertions, as this is not the focus of this research. Additionally for performance reasons, we only consider extractions and re-insertions of the tool when no other way to proceed exists, so our solution may not find the optimal extractions and re-insertion locations. In the parametric domain of the surface, this takes the form of straight line that cyclically loops around the domain 40 times. The stock (or initial model) used in our simulations is the knot model offset outward in the normal direction by 0.1.

Figures 6 and 7 show the results of this machining process. In Figure 6, the model was machined using a ball-end tool. Local collision detection in this case simply checks no gouging occurs along the tool-path, as orientation does not affect the local behavior of the tool. In Figure 7, a flat end tool was used to machine the model. In some places the surface finish achieved by the flat-end tool is better than that achieved by the ball-end tool, while in others the ball-end tool gives better performance. The difference stems from places where global accessibility considerations prevent optimal local placement of the flat-end tool, degrading performance. The local performance of a ball-end tool on the other hand is unaffected by global considerations, and its performance in terms of surface finish remains relatively constant.

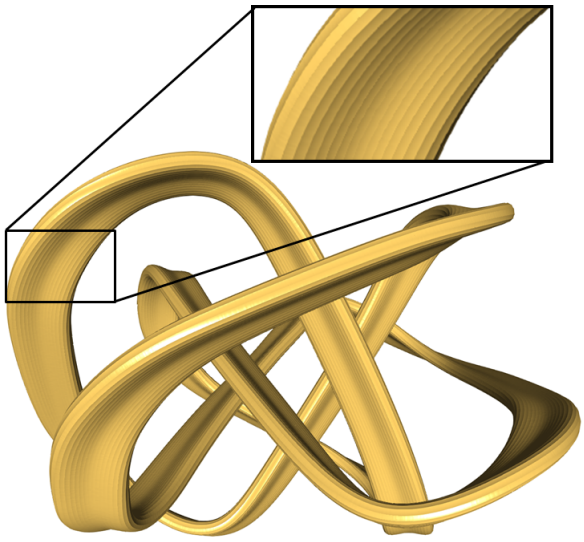

Figure 6: The knot model (for comparative tool size see Figure 5). Machined with a ball-end tool. A video of this machining process can be viewed at can be viewed in https://youtu.be/VuNdxDkJspk.

To get a quantitative measurement of the above (qualitative) result, we compare the simulation's result to the original knot model. To minimize the effect of the chosen CC curve (whose generation is not a part of the algorithms we present) errors are normalized using the performance of the ball-end tool, which is constant for a given CC curve. For mesh comparison we used the (approximated) Hausdorff distance computation in [22], based on 16.2 million sampled points on the surface of the original model and their distance to the simulation result. The measured distances are shown in Table 1. The best (lowest) maximal distance is achieved by the ball-end tool, whose local behavior is unaffected by global considerations, whereas the flat-end tool, with local behavior that is determined by the tool orientation, achieves better average (mean) performance.

Links to videos of the above machining tasks, can be found in the captions of Figures 6 and 7.

The next example uses a more realistic model: a bladed disk, possibly a turbine component. The model is shown in Figure 8. To accommodate the smaller scale of this model, for example

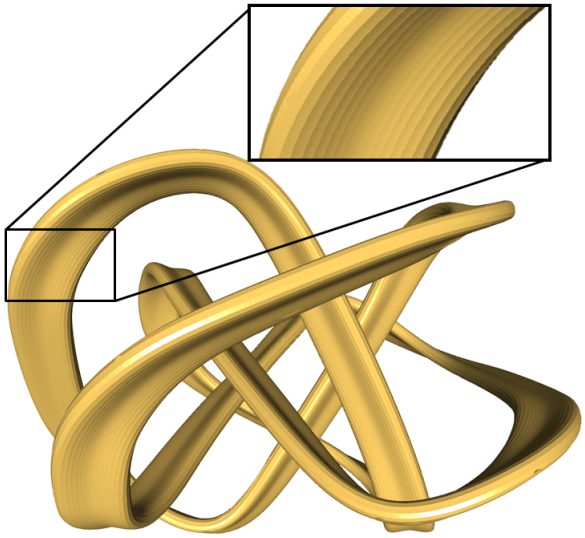

Figure 7: The knot model (for comparative tool size see Figure 5). Machined with a flat-end tool. A video of this machining process can be viewed at can be viewed in https://youtu.be/XE7Dy39_bHU.

Table 1: Distance, or error, between the simulation results and the original knot model. See Figures 6 and 7. The distances given relative to the mean error for a ball-end tool of $4.5e^{-3}$.

Simulation Scenario	Relative Max Distance	Relative Mean Distance
Figure 6, Ball-end tool	6.7	1.0
Figure 7, Flat-end tool	13.6	0.88

the thin trailing and leading edges of the blades, we changed r_{max} to 0.2. The CC curve we use goes back and forth along (both sides of) the blade 80 times, shifting a bit each time in the perpendicular direction to cover the whole surface. In the parametric domain of the surface, this takes the form of 80 cycles of a square wave. As the base of the blade actually includes normal curvature values that are much higher than the maximal effective curvature of the tools, we avoided a small area near the base of the blade. To create the stock model we added a ruled volume, between most of the machined surface area and a (normal) offset by 0.1 of the same surface, to the original model. Note that a small part of the machined area, near the base of the blade, was left uncovered by the added ruled volume so we can observe the effect of machining on the original surface as well. The machined area can be observed in Figure 9.

Figures 9 and 10 show the results of machining a blade using a ball-end and flat-end tools respectively. Figure 11 shows the result of machining a single blade model, without considering the rest of the model using a flat-end tool. Links to videos of the related simulations can be found in the captions of the figures.

As with the knot model, we use an approximated Hausdorff distance computation to evaluate machining performance. This time the comparison is based on 15 million sampled points on the target model: a single blade without the added ruled volume. Distance is measured from these points to the mesh of the simulation results, which are machined from the model with the

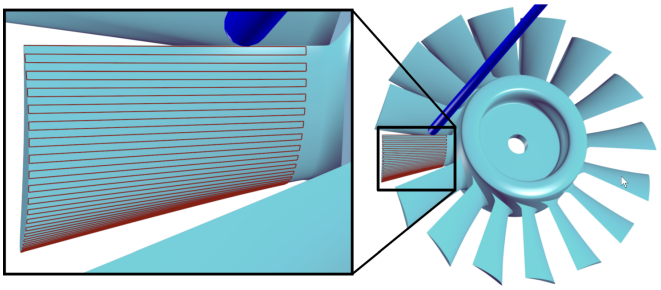

Figure 8: A model of a bladed disk, with a radius 1 ball-end tool. The CC curve used in our experiments is also shown (in red).

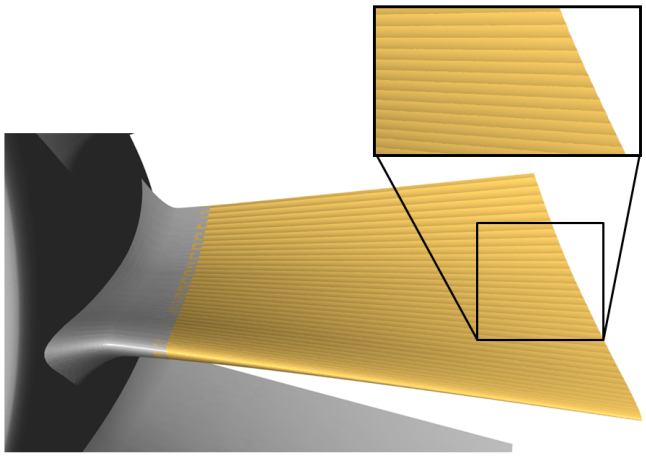

Figure 9: The model from Figure 8 after machining one of the blades, using a ball-end tool (for comparative tool size see Figure 8). A video of the related simulation can be viewed at https://youtu.be/HGtkAU2c3sA.

added ruled volume. The results are shown in Table 2. These results again confirm what Figures 9, 10, and 11 showed earlier: in areas where global considerations limit the placement of a flat-end tool ball-end performance (in terms of surface finish) is better, but in areas where optimal placement of the tool is possible, the flat-end outperforms the ball-end tool. This effect is especially apparent in the result shown in Figure 10: the area of rectangle (a), which is difficult to reach (due to the neighboring blade) has worse surface finish than the very accessible area of rectangle (b). In Figure 11, on the other hand, the surface finish is very good (smoother) everywhere. This is because global accessibility is no longer a factor, and the mild normal curvature in the machined area of the blade can be very well matched by the tool. On a related note relating to the discussion in Section 4.3, if H_t is set to 0, the sharp trailing edge of the blade can only be machined from a very limited set of directions. This, in turn, creates a noticeable degradation in the surface finish near the trailing edge when it is machined using a flat-end tool.

In the next examples, we take a closer look at the performance of our algorithms. We do so for three presented models and specified CC curves. The first model and CC curve are shown in Figure 12 (b). In Figure 12 (b), to generate the performance data only the hub and three blades were used: the blade with the CC curve and the two adjacent blades. The generated

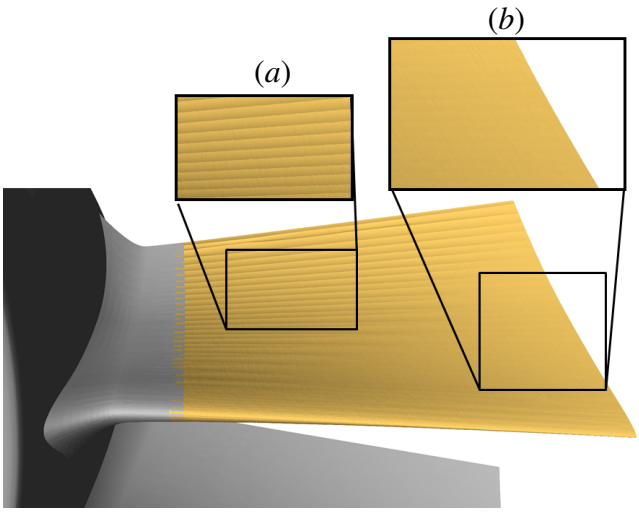

Figure 10: The model from Figure 8 after machining one of the blades, using a flat-end tool (for comparative tool size see Figure 8). Note the smoother finish in the rectangle (b), that is more accessible, relative to the rectangle (a) marking a less accessible location. A video of the related simulation can be viewed at https://youtu.be/vNCdcS3wyrY.

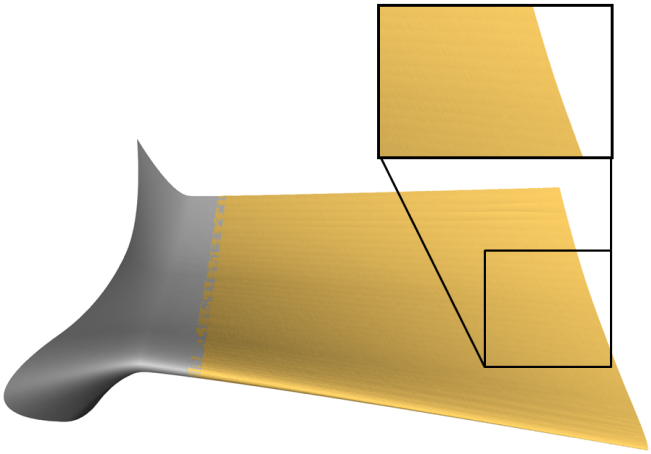

Figure 11: A single blade from the model in Figure 8 machined using flat-end tool (for comparative tool size see Figure 8). The tool-paths were generated without considering global collisions with the rest of the model, resulting in a smoother finish. A video of the related simulation can be viewed at https://youtu.be/yHve2Mgk9QU.

performance data does not account for the other blades. The second model is a single blade shown Figure 12 (a), and using the same CC curve. The third model is the knot model, with a single iso-parametric curve along the length of the knot as a CC curve, shown in Figure 13.

The first thing we evaluate is the computation time for the normal curvature bounds. The results for the various models can be seen in Table 3. As these results show, the computation time greatly depends on the geometry of the model. This result holds even if the times are normalized for the number of sub-surfaces in the model. Additionally, the computation times are consistently higher for a flat-end tool than a ball-end tool.

Table 2: Distance, or error, between the simulation results and the theoretical result. The distances given relative to the mean error for a ball-end tool of $2.4e^{-3}$.

Simulation Scenario	Relative Max Distance	Relative Mean Distance
Figure 9, ball-end tool	9.7	1.0
Figure 10, flat-end tool	13.9	0.63
Figure 11, flat-end tool	4.4	0.25

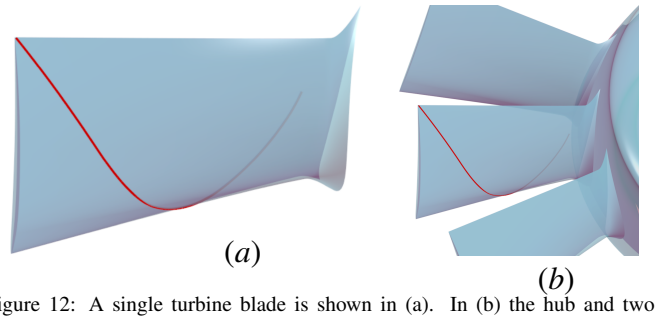

Figure 12: A single turbine blade is shown in (a). In (b) the hub and two adjacent blades are also shown. In both cases, a portion of the middle blade is rendered as semi-transparent to show the CC curve as it loops around the blade. Note that in (b) only the hub and three of the blades, the blade with the CC curve and the two adjacent blades, are accounted for in the accompanying performance data.

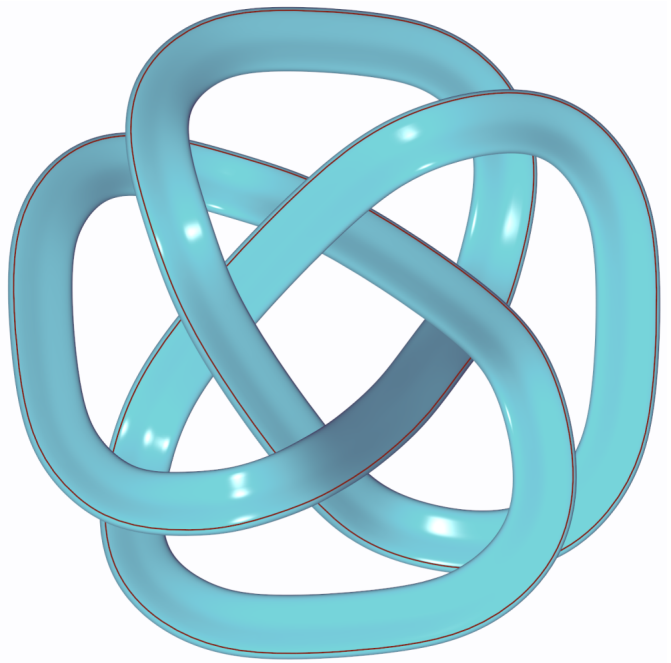

Figure 13: The knot model, with a single iso-parametric curve used in our experiments highlighted.

For a ball-end tool, it is enough to classify normal curvature bounds as greater or lesser than the *constant* effective curvature value of the tool. In this case, greater accuracy is only needed for curvature bounds (intervals) that contain the single effective curvature value of the tool. On the other hand, for a flat-end tool the effective curvature has a range of possible values (depending on the orientation), and so a longer computation time is needed to more accurately calculate normal curvature bounds

that overlap the given range. The number of sub-surfaces for which normal curvature bounds are needed may be relatively small compared to the total number of sub-surfaces. Hence, using a lazy evaluation strategy, for example, should improve the computation time of the whole turbine scenario (with about $50K$ sub-surfaces in the hub alone) to less than that of the single blade scenario.

Table 3: Computation times for the normal curvature bounds for several models, and the number of sub surfaces the models were divided to. The number in parenthesis in the computation time column is the number per sub-surface.

Scenario	Sub-Surface #	Normal Curvature Bounding Time [s]
Figure 12 (a), ball-end	6671	0.47 ($7.02e^{-5}$)
Figure 12 (a), flat-end		5.7 ($8.5e^{-4}$)
Figure 12 (b), ball-end	71117	86.1 ($1.2e^{-3}$)
Figure 12 (b), flat-end		375.1 ($5.3e^{-3}$)
Figure 13, ball-end	20128	575.4 ($2.82e^{-2}$)
Figure 13, flat-end		620.4 ($3.1e^{-2}$)

Table 4 lists computation times for producing valid (optimal) orientations for the tool-paths. These times include all steps except the initial subdivisions to sub-surfaces (which are negligible), and the normal curvature bounds computations presented earlier. Each column in Table 4 includes the times in subsequent columns. The Local Collision Time column lists computation times for local collision detection (for flat-end tools), while the Global & Local Collision Time column also includes global collision times. In addition to the time listed in the Global & Local Collision Time column, the Total C-space Time column also accounts for the time it takes to generate the C-space and find the optimal path through it. As these results show, for large models, the dominant factor in these times is global collision detection. This is not surprising as global collision detection must consider the entire model. On the other hand, local collision detection is more localized, involving a smaller number sub-surfaces near the tip (or extended tip). Comparing the results of the single blade, to those of the three bladed turbine, for the flat-end tool, illustrates this point: the local collision time remains more or less the same, as it involves the same CC curve on the same sub-surfaces. The global collision time however, increases dramatically, as the model with the hub and three blades contains more than ten times the number of sub-surfaces in a single blade model.

6. Future Work

One additional feature that can be added to the calculation presented in Section 3, for bounds on principal curvature values, is the computation of bounds on the principal curvature *directions* as well. Once the intervals for the principal curvatures are known, bounds on the principal curvature directions can also be calculated using interval arithmetic. The bounds for the two principal curvature directions will take the form of two

Table 4: Tool-path orientation computation times. Local collision times are only listed for flat-end tools. For ball-end tools these times do not depend on the tool-orientation, and are are negligible. Note that each column includes the times listed in subsequent columns (to the right).

Simulation Scenario	Total C-space Time [s]	Global & Local Collision Time [s]	Local Collision Time [s]
Figure 12 (a), ball-end,	15.8	13.4	N/A
Figure 12 (a), flat-end,	26.7	25.3	11.2
Figure 12 (b), ball-end	217.4	215.8	N/A
Figure 12 (b), flat-end	247.7	246.3	13.5
Figure 13, ball-end	75.05	58.7	N/A
Figure 13, flat-end	122.3	99.8	38.7

cones, with perpendicular axes, that will encompass all possible principal curvature directions for a given surface. Knowledge of the principal curvature directions can be used to optimally align tools: given a flat end tool, we can position the tool even in surface regions with very high normal curvature values by aligning the sharp corner of the tool with the maximal principal curvature direction. By using Euler's Theorem [11], knowledge of the principal curvature directions can also be used to optimally align tools in flank machining, or even possibly align hyperbolic cutters (tools).

Recall that higher values of H_t are more likely to cause failures in local gouging tests. In our experiments, H_t was set to a single constant value. It is possible however to dynamically (and optimally) set H_t for every sub-domain: starting with $H_t = 0$ for a given sub-domain, then incrementing it by some small value if Algorithm 1 indicates no local collision. The largest value of H_t where no local collision is detected can then be used when testing for global collisions, without potentially compromising the local collision tests.

Corollary 4.2 can be used in many different ways, not just the one presented in Section 4.1. For example, it is possible to apply Corollary 4.2 to both local and global collisions. This can be done in two steps: first finding all surfaces that are partially inside some bounding volume for the whole tool (a frustum), and are not completely back-facing with regard to the tool direction. The second step is to ensure all of these surfaces are in a single C^0, piecewise C^2, neighborhood of the contact point. Since a collision (global or local) can only occur if a collision with one of the surfaces found in the first step, eliminating such collisions using corollary 4.2 in step two, eliminates all collisions.

7. Conclusion

Bounding surface positions using the control mesh, and surface normals using normal cones, are both well known procedures. In this paper, we add to these, a procedure to bound the (normal) curvature values of a surface. In the context of machining, normal curvature bounds allow the second order behavior of a surface to be conservatively approximated by that

of the tool, without resorting to point sampling. The methods presented in this work show that collision free tool-paths can be generated efficiently using information based on bounding, rather than sampling of the model surface, ensuring the accessibility of the tool, globally.

Acknowledgements

This research was supported in part with funding from the Defense Advanced Research Projects Agency (DARPA), under contract HR0011-17-2-0028. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] A. Lasemi, D. Xue, P. Gu, Recent development in cnc machining of freeform surfaces: A state-of-the-art review, Computer-Aided Design 42 (7) (2010) 641–654.

[2] G. Elber, E. Cohen, A unified approach to accessibility in 5-axis freeform milling environments, in: Machining impossible shapes, Springer, 1999, pp. 376–386.

[3] Y.-J. Kim, G. Elber, M. Bartoň, H. Pottmann, Precise gouging-free tool orientations for 5-axis cnc machining, Computer-Aided Design 58 (2015) 220–229.

[4] N. Rao, F. Ismail, S. Bedi, Tool path planning for five-axis machining using the principal axis method, International Journal of Machine Tools and Manufacture 37 (7) (1997) 1025–1040.

[5] C. G. Jensen, Analysis and synthesis of multi-axis sculptured surface machining, Ph.D. thesis, Purdue University (1993).

[6] P. Gray, S. Bedi, F. Ismail, Rolling ball method for 5-axis surface machining, Computer-Aided Design 35 (4) (2003) 347–357.

[7] P. J. Gray, S. Bedi, F. Ismail, Arc-intersect method for 5-axis tool positioning, Computer-Aided Design 37 (7) (2005) 663–674.

[8] P. Bo, M. Bartoň, H. Pottmann, Automatic fitting of conical envelopes to free-form surfaces for flank cnc machining, Computer-Aided Design 91 (2017) 84–94.

[9] P. Bo, M. Bartoň, D. Plakhotnik, H. Pottmann, Towards efficient 5-axis flank cnc machining of free-form surfaces via fitting envelopes of surfaces of revolution, Computer-Aided Design 79 (2016) 1–11.

[10] A. Warkentin, F. Ismail, S. Bedi, Multi-point tool positioning strategy for 5-axis mashining of sculptured surfaces, Computer Aided Geometric Design 17 (1) (2000) 83–100.

[11] M. P. Do Carmo, Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition, Courier Dover Publications, 2016.

[12] T. W. Sederberg, R. J. Meyers, Loop detection in surface patch intersections, Computer Aided Geometric Design 5 (2) (1988) 161–171.

[13] T. W. Sederberg, A. K. Zundel, Pyramids that bound surface patches, Graphical Models and Image Processing 58 (1) (1996) 75–81.

[14] G. Barequet, G. Elber, Optimal bounding cones of vectors in three dimensions, Information Processing Letters 93 (2) (2005) 83–89.

[15] T. Hickey, Q. Ju, M. H. Van Emden, Interval arithmetic: From principles to implementation, Journal of the ACM (JACM) 48 (5) (2001) 1038–1068.

[16] T. Lozano-Perez, Spatial planning: A configuration space approach, IEEE transactions on computers c-32 (2) (1983) 108–120.

[17] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, R. Rowe, Collision detection: A survey, in: Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on, IEEE, 2007, pp. 4046–4051.

[18] H. Pottmann, J. Wallner, G. Glaeser, B. Ravani, Geometric criteria for gouge-free three-axis milling of sculptured surfaces, Journal of Mechanical Design 121 (2) (1999) 241–248.

[19] E. F. Moore, The shortest path through a maze, in: Proc. Int. Symp. Switching Theory, 1959, 1959, pp. 285–292.

[20] B. Ezair, G. Elber, Automatic generation of globally assured collision free orientations for 5-axis ball-end tool-paths, Computer-Aided Design 102 (2018) 171–181.

[21] H. J. Haverkort, Results on geometric networks and data structures, Ph.D. thesis, Utrecht University (2004).

[22] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, Meshlab: an open-source mesh processing tool., in: Eurographics Italian chapter conference, Vol. 2008, 2008, pp. 129–136.

[23] G. Elber, Free form surface analysis using a hybrid of symbolic and numerical computation, PhD Thesis, University of Utah, 1992.

[24] T. D. DeRose, R. N. Goldman, H. Hagen, S. Mann, Functional composition algorithms via blossoming, ACM Trans. Graph. 12 (2) (1993) 113–135.

[25] J. Machchhar, D. Plakhotnik, G. Elber, Precise algebraic-based swept volumes for arbitrary free-form shaped tools towards multi-axis cnc machining verification, Computer-Aided Design 90 (2017) 48–58.

Appendix A. Bounding the Tool Behavior

In this Appendix, we go into some of the details related to the computations of bounding volumes and effective curvature of the tool (as in Definition 4.1), for a given sub-domain of the configuration space. For a single configuration, the tool position and effective curvature depend on three factors: the contact point, p, the surface normal at that point, $\hat{n}$, and the tool axis direction, $\hat{t}$. These factors in turn are determined by the C-space coordinates (t, Θ, Φ), $C(t)$, and $\mathcal{S}$.

For a range of configurations (a sub-domain) $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, we can bound the behavior of p, $\hat{n}$, and $\hat{t}$. p is bound in the convex hull of the control polygon of $C([t_a, t_b])$. $\hat{t}$ is bound inside a cone, $C_t(\hat{d}_t, \alpha_t)$, which in turn is determined by $[\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$. Similarly, by performing a composition [23, 24], between the relevant segment of the CC curve in the domain of $\mathcal{S}$, $c([t_a, t_b])$, and the normal surfaces of $\mathcal{S}$, $\mathcal{S}_n$, the normal cone of $\bar{n}(t) = \mathcal{S}_n(c([t_a, t_b]))$, $C_n(\hat{d}_n, \alpha_n)$, that bounds all $\hat{n}$ (normal) directions can be found. In the following sections, we will make use of these bounds.

Appendix A.1. Calculating the Effective Curvature in a Sub-Domain

In this section, we discuss calculating, $T_{curvature}$, the effective curvature for a tool, as defined in Definition 4.1.

Algorithm 2 summarizes the steps needed to calculate the effective curvature for a given sub-domain. For a ball-end tool, the effective curvature is determined by the tool radius, R_t, and is equal to the curvature of the hemispherical tool-tip: $\frac{1}{R_t}$. For a flat-end tool, we can calculate the effective curvature according to Figure 1. As $T_{curvature}$ depends monotonically on ω, we can calculate the minimal effective curvature for a whole sub-domain based on the minimal value of ω (Line 6 in Algorithm 2). Recall that we a priori mark any sub-domain that contains a configuration with $\omega_{min} = 0$, as invalid.

For a flat-end tool, if $R'_t < \sqrt{\left(\frac{H_t}{2}\right)^2 + R_t^2}$ then the calculations for a normal tip, and an extended tip (as in Definition 4.2), diverge. In this case, the effective curvature is no longer independent of the chosen value of H_t. This is because the radius of the sphere that determines the effective curvature cannot be smaller than the radius of the minimal sphere bounding the extended tip. While the effective curvature can still be determined

Algorithm 2 ToolEffectiveCurvature

Input:

(1) $c(t)$, a CC curve, a parametric curve on $\mathcal{S}$, $t \in [t_a, t_b]$;

(2) $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, a C-space sub-domain;

(3) T, a convex tool, of a given type and radius, R_t. H_t, the extended tip length for a flat-end tool, should also be specified;

Output:

(1) $T_{curvature}$, the effective tool curvature in the sub-domain $\mathcal{U}$;

Algorithm:

1: **if** $BallEndTool(T)$ **then**
2: $T_{curvature} := \frac{1}{R_t}$;
3: **else if** $FlatEndTool(T)$ **then**
4: $C_n(\hat{d}_n, \alpha_n) := NormalCone(\mathcal{S}_n(c([t_a, t_b])))$;
5: $C_t(\hat{d}_t, \alpha_t) := BoundingCone(S^2([\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]))$;
6: $\omega := max\left(\arccos\langle \hat{d}_t, \hat{d}_n \rangle - \alpha_n - \alpha_t, 0\right)$;
7: $R'_t := \frac{R_t}{\sin\omega}$;
8: **if** $R'_t \geq \sqrt{\left(\frac{H_t}{2}\right)^2 + R_t^2}$ **then**
9: $T_{curvature} := \frac{1}{R'_t}$;
10: **else**
11: $T_{curvature} := -\infty$; // fail, no curvature matching
12: **end if**
13: **else**
14: $T_{curvature} := ConvexCurvature(C(t), \mathcal{U}, T)$;
15: **end if**
16: return $T_{curvature}$;

in these cases using Definition 4.1, it will no longer perform curvature matching. This situation is tested in Line 8 in Algorithm 2. For the $H_t = 0.4R_t$ value chosen in Section 5 this will only occur if ω is about 10 degrees away from being a right angle, which should occur rarely in practice.

Line 14 in Algorithm 2, which we did not implement, would find the minimal effective curvature in the sub-domain (according to Definition 4.1) for other types of convex tools (such as toroidal).

Appendix A.2. Finding The Convex Bounding Volume for the Tip of the Tool in a Sub-Domain

Here we discuss ways to find a convex bounding volume for the tip of the tool for all configurations in a sub-domain. The main consideration in picking the approach used to find the convex bounding volume, is the tightness of the bound, versus the required computational effort. While it is possible for very accurate sweeps to be generated (such as in [25]), we chose to use a less computationally intensive approach that still produces results that are quite tight, using a bounding sphere.

Algorithm 3 summarizes the steps needed to calculate a bounding sphere for the tip of the tool in a given sub-domain $\mathcal{U}$. Line 8 in Algorithm 3 performs the main calculation for

Algorithm 3 ToolTipConvexBoundingVolume

Input:

(1) $c(t)$, a CC curve, a parametric curve on $\mathcal{S}$, $t \in [t_a, t_b]$;

(2) $\mathcal{U} = [t_a, t_b] \times [\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]$, a C-space sub-domain;

(3) T, a convex tool, of a given type and radius, R_t;

Output:

(1) B, a sphere bounding the tip of the tool in $\mathcal{U}$;

Algorithm:

1: $C_n(\hat{d}_n, \alpha_n) := NormalCone(\mathcal{S}_n(c([t_a, t_b])))$;
2: $C_t(\hat{d}_t, \alpha_t) := BoundingCone(S^2([\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f]))$;
3: **if** $BallEndTool(T)$ **then**
4: $p_m := C(\frac{t_a+t_b}{2})$; // center point of $C(t)$
5: $R_p := \max_{p \in C([t_a, t_b])} \|p - p_m\|$; // bounding radius of $C(t)$
6: $R_n := \max_{\hat{n} \in C_n} R_t \|\hat{n} - \hat{d}_n\|$; // bounding radius of $\hat{n} \in C_n$
7: $B.center := p_m + \hat{d}_n R_t$;
8: $B.radius := R_t + R_p + R_n$;
9: **else if** $FlatEndTool(T)$ **then**
10: $[\hat{n}] := BoundingInterval(\hat{n} \in C_n(\hat{d}_n, \alpha_n))$;
11: $[\hat{t}] := BoundingInterval(\hat{t} \in C_n(\hat{d}_t, \alpha_t))$;
12: $[p] := BoundingInterval(p \in C([t_a, t_b]))$;
13: $[c] := [p] + \left(\frac{[\hat{n}] \times [\hat{t}]}{\|[\hat{n}] \times [\hat{t}]\|}\right) \times [\hat{t}] R_t + \left(\frac{H_t}{2}\right)[\hat{t}]$;
14: $B1 := BoundingSphere([c])$;
15: $B.center := B1.center$;
16: $B.radius := \sqrt{\left(\frac{H_t}{2}\right)^2 + R_t^2}) + B1.radius$;
17: **else**
18: $B := BoundingVolume(C(t), \mathcal{U}, T)$;
19: **end if**
20: return B;

a ball-end tool, adding together, R_t, the radius of the tool, R_p, the radius for the bounding sphere of the contact point, and R_n, the radius of the bounding sphere for the surface normal at the contact point, multiplied by R_t. In Lines 10 to 13 in Algorithm 3, the main calculation for a flat-end tool is performed, based on interval arithmetic. The calculation starts by setting, $[\hat{n}]$, $[\hat{t}]$, and $[p]$, each a vector interval that bounds (like a bounding box) the possible values of the respective vector ($\hat{n}$, $\hat{t}$, and p) in the sub-domain $\mathcal{U}$. In Line 13, $[c]$, the vector interval for the center of the extend tip of the tool, is similarly calculated. The calculation is essentially the same as calculating the center of the extended tip, given specific $\hat{n}$, $\hat{t}$, and p, only performed for the intervals of these values: starting with the possible locations of the contact point, $[p]$, we add the possible distances to the bottom disc of the tool, $\left(\frac{[\hat{n}] \times [\hat{t}]}{\|[\hat{n}] \times [\hat{t}]\|}\right) \times [\hat{t}] R_t$, and then the possible distances to the center of the extended tip, $\left(\frac{H_t}{2}\right)\hat{t}$.

Finally, Line 18 in Algorithm 2, which we did not implement, would find the bounding sphere in the sub-domain for other types of tools (such as toroidal).

Figures A.14 and A.15 illustrates some of the elements involved in Algorithm 3. Note that for a flat end tool we account for an extended tip with its larger bounding sphere radius.

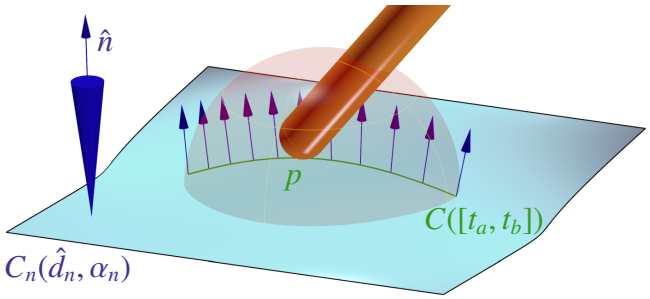

Figure A.14: The calculation of the bounding volume for the tip of a ball-end tool, in a sub-domain of the C-space, is based on bounding all contact points $p \in C([t_a, t_b])$, and the surface normal direction at all contact points $\hat{n} \in C_n(\hat{d}_n, \alpha_n)$. In this case the tool orientation is irrelevant.

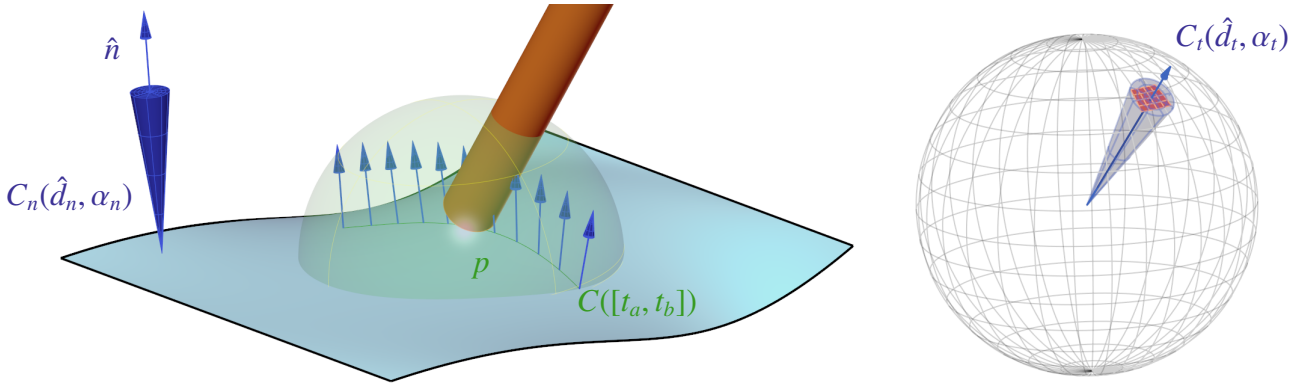

Figure A.15: The calculation of the bounding volume of the extended tip for a flat-end tool, in a sub-domain of the C-space, is based on bounding all contact points $p \in C([t_a, t_b])$, the surface normal direction at all contact points, $\hat{n} \in C_n(\hat{d}_n, \alpha_n)$, and the orientation of the tool $\hat{t} \in C_t(\hat{d}_t, \alpha_t)$. The tool orientation is bounded based on based on $S^2([\Theta_c, \Theta_d] \times [\Phi_e, \Phi_f])$.