Accessibility for Line-Cutting in Freeform Surfaces

Boris van Sosin[1,*], Michael Bartoň[2], Gershon Elber[1]

Abstract

Manufacturing techniques such as hot-wire cutting, wire-EDM, wire-saw cutting, and flank CNC machining all belong to a class of processes called line-cutting where the cutting tool moves tangentially along the reference geometry. From a geometric point of view, line-cutting brings a unique set of challenges in guaranteeing that the process is collision-free. In this work, given a set of cut-paths on a freeform geometry as the input, we propose a conservative algorithm for finding collision-free tangential cutting directions. These directions, if they exist, are guaranteed to be globally accessible for fabricating the geometry by line-cutting. We then demonstrate how this information can be used to generate globally collision-free cut-paths. We apply our algorithm to freeform models of varying complexity.

Keywords: Accessibility analysis, Subtractive manufacturing, Hot-wire cutting, wire-EDM

1. Introduction & Motivation

Geometric modeling of a curved object represented as a set of NURBS surfaces is a well known and common task, using contemporary geometric CAD software. In contrast, physical realization (or manufacturing) of curved objects is far more challenging. Given a free-form object, its *realization* as a tangible object has attracted a considerable research attention, for many years. See [1] for a recent survey. The geometric challenges can be mostly found in multi-axis machining, where the tool can cut at its tip (i.e. [2]) or on it side (shank, i.e [3]).

Modeling of manufacturing processes requires one to consider many factors: starting from geometrical issues like local and global accessibility, collision detection, and manufacturing technology, through related issues such as the choice of tool's shapes and sizes (for example in the context of 5-axis flank CNC machining), to physical parameters like material properties. For certain type of materials such as foam or polystyrene, (hot-wire) line-cutting is the most common subtractive manufacturing (SM) methodology. Other types of line-cutting processes also include wire electrical-discharge machining (wire EDM), and even more traditional tools, such as the band-saw. In hot-wire cutting, for example, the cutting tool is typically a thin straight conductive wire, stretched between two points on a rigid frame. During the manufacturing process, an electric current passes through the wire, heats it up, and the wire consequently cuts a ruled surface off the material, as it moves in space, forming the desired shape.

The development of hot-wire cutting machines started with 2-axis machines. Having the wire fixed vertically, the material block moves on a 2D platform, yielding two degrees of freedom. The state-of-the-art technology offers hot-wire machines with 5 degrees (or more) of freedom which can remove large amounts of material in a single cut, making it suitable for manufacturing of general 3D objects. Alternatively, the hot wire can be tensioned on a bracket, mounted on a robotic arm, (e.g. see Figure 1), to yield a multi-axis control. Most recently, a robotic hot-blade prototype that consists of 18 degrees of freedom has been introduced [4, 5]. The extra degrees of freedom stem from the fact that the cutting curve is not a straight line, but a bent 3D curve. The tool-path planning algorithm uses the bending energy of the wire to adapt its shape dynamically to the surface during the cutting process. While having that many of degrees of freedom is an advantage from the point of view of approximation quality, curved wire is a fragile tool and a straight wire dominates the industry.

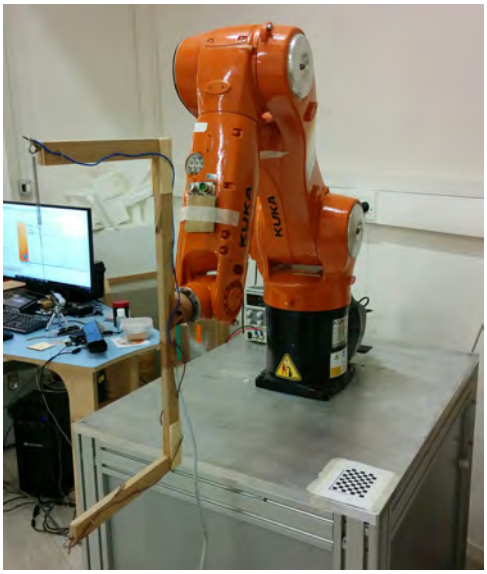


Figure 1: A hot-wire cutting configuration based on a 6-DoF robot arm.

Another work inspired by how wire cutting deals with approximation of free-form surfaces by 3D motions of planar curves [6]. The 3-parametric space of planes intersecting the input sur-

*Corresponding author. E-mail address: sosin@cs.technion.ac.il
[1]Computer Science Department, Technion - Israel Institute of Technology, Haifa, Israel
[2]BCAM - Basque Center for Applied Mathematics, Bilbao, Basque Country, Spain

face is explored to reveal segments of the planar cuts that admit tangential movability along the free-form surface. Partial shape matching is sequentially applied to find congruent profiles that are used to initialize the sweep approximation. Finally, an optimization loop is performed to optimize the planar profile and its motion. However, global collision detection between a handler of the curved wire and the surface is not considered.

In this work, we consider hot-wire cutting with a *straight wire* in a multi-axis cutting context. We focus solely on the geometric aspects of line-cutting, most notably the accessibility questions, which can be relevant also to any physical implementation of the line-cutting process.

1.1. Previous work

In the computer aided geometric design literature, quite a few results are devoted to free-form surface approximation by ruled surfaces [7, 8, 9, 10, 11, 12, 13, 14, 15, 16].

Rationalization of complex surfaces appearing in free-form architecture is studied by Flöry and Pottmann [7]. The input free-form surface is approximated by ruled surface strips. The rulings of the strips are initialized by the asymptotic curvature network and the initial planar strips are further optimized to become curvature continuous across strip boundaries.

Sprott and Ravani approximate a ruled surface by a motion of a cylindrical tool [10]. They conceptualize a ruled surface as a curve on Plücker's quadric and use the striction curve of the ruled surface to initialize the position of the tool as the axis of the cylinder is required to intersect surface normals along a single ruling. Senatore et al. look for the maximum size of a cylindrical tool while guaranteeing bounds on under- and over-cutting errors[12].

Li et al. consider approximation of a ruled surface by a motion of a conical tool and propose an optimization base approach [8]. The conical tool moves along two guiding rails and position of the tool's axis is sequentially optimized to minimize the error between the tool and the input ruled surface. Another work that considers cylindrical cutters and uses optimization approach is [11]. The offset of the input ruled surface is computed and three curves on it are selected. A position of the tool's axis is computed to minimize, in the least square sense, the distance from these three curves. This formulation is linear and therefore only linear system of equations needs to be solved to compute the position. However, the algorithm requires three guiding curves and therefore the initialization is hardly automatic.

Elber and Fish use subdivision approach to approximate a general free-form surface by ruled surfaces [9]. The reference surface is subdivided along parametric directions to get a set of bilinear patches as the final approximation. Each patch is guaranteed to approximate the reference surface within a user-defined threshold and, due bilinearity, two possible solutions of ruling's motion are found for each patch. To meet high tolerances, however, a large number of subdivision steps is needed.

An automated ruled-surface approximation algorithm is introduced in [15]. This algorithm starts with a point cloud as the input (even thought conceptually it can also work on a discrete sampling of parametric surfaces), which is first partitioned into elliptic and hyperbolic regions. For hyperbolic regions (parabolic being treated as a special case of hyperbolic), the asymptotic curves on the sampled input data are computed and used as for initialization of the rulings. The initial ruled surface is further optimized towards smoothness and approximation quality.

Another attempt to optimize the piecewise ruled surface approximation was proposed in [14] that employed multi-dimensional dynamic programming between opposite rail curves on the input general freeform surfaces. By optimally matching the sampled points on the rails curves, a set of piecewise ruled surface approximation with minimal error can be computed.

A similar approach that uses asymptotic curve network is proposed in [16]. The segmentation into strips is formulated as a level-set problem, in which discretely sampled points $\mathbf{x}$ on the input surface are assigned values $f(\mathbf{x})$ and f is optimized to choose strip boundaries with properties which are desirable, both for the ruled surface fitting and for the overall aesthetic quality of the end-result. Specifically the algorithm seeks conoidal ruled surfaces that are preferable for fabrication of wooden panels, as all the panels can be placed parallel to the same plane.

In the context of 5-axis CNC machining, physical experiments of machined ruled surfaces using cylindrical milling tools are presented in [13]. The key ingredient is the feedrate adjustment that increases the machining quality, especially in highly curved regions.

Our work tackles a problem of *global accessibility*. While global collision detection algorithms for flat-end and ball-end CNC milling are quite well studied, see e.g. [1] and the references therein, to the best of our knowledge, there is very little research on global accessibility in the context of flank (aka side or peripheral) CNC machining, see discussion [3, Section 3]. Most of the research on flank milling deals only with *local* collisions [17, 18, 19]. Typically, locally gouging-free position of the tool is computed via optimization, followed by global collision test as a post-process that eliminates poses that collide globally [20, 21]. In contrast, in the proposed work we aim to design the motion of the tool (hot-wire), considering global collision directly in the motion-planning stage. Moreover, since most of the above discussed works were inspired by CNC machining, only collision detection of a ruled surface with *finite* rulings is considered. In this work, our assumptions are different as they come from the constellation of the hot-wire cutting machine, namely that material is posed inside a large frame that holds the hot wire, see Figure 1, and therefore the design surface has to be accessible by ruled surfaces with *infinite* rulings.

Recently, hot-wire cutting was studied for a special class of input surfaces, namely minimal surfaces [22]. These surfaces are in fact manufactured as thin shells and therefore the material has to be removed from both sides of the surface. The manufacturing technology is 5-axis hot wire cutting, however, the path-planing is realized on both sides of the surface. Weierstrass parametrization is used to map a complex planar domain to a minimal surface. Hot-wire cutting lines follow the principal curvature lines of the surface, which are claimed as positions that minimize global collisions. However, these arguments are based on experimental observations which is in contrast to asymptotic lines approach [16] that offers locally the best fit of a line to a surface. When compared to our approach, [22] consider only a special class of (minimal) surfaces while our algorithm handles any (at least C^2 continuous) free-form surface.

Another category of research dealing with tangential contact of infinite lines and curved surfaces belongs to computer graphics, namely to computing silhouettes of 3D objects [23]. While for a 3D quadric, its silhouette from an arbitrary viewpoint is known to be a planar curve (conic section), for surfaces of higher degrees and more complicated shapes, the silhouettes are 3D curves [24]. The silhouette is a contact curve between the input surface and the silhouette cone. Given a model $\mathcal{M}$, its outer visual hull, $VH(\mathcal{M})$, is the maximal water-tight model which has identical silhouettes as $\mathcal{M}$ in all viewing directions outside the convex hull of $\mathcal{M}$ [23]. The visual hull is then the intersection of all possible silhouette cones, considering all possible viewpoints

lying outside the convex hull of the input surface. The hot-wire cutting problem is therefore linked to the visual hull problem by the fact that the visual hull is the smallest super-set (in terms of inclusion) of $\mathcal{M}$ that can be manufactured by infinite lines.

The visual hull problem have been introduced and extensively investigated by A. Laurentini, in the context of reconstructing 3D shape from multiple photos [23]. The 2D visual hull is first defined on planar configurations of multiple polygons P, and an algorithm is proposed for computing the visual hull of such scenes. Visual hull computation of 2D scenes is done by partitioning the plane into regions by the number of lines passing through these regions that do not pass through the interior of any polygon in P. The regions for which this number is zero form the interior of the visual hull of P. This approach is generalized to polyhedra in 3D (which, of course needs to partition 3D with planes, rather than lines). In a series of papers, this approach has been further generalized to computing the visual hull of shapes of increasing complexity: solids of revolution [25], smooth curved solids [26], and piecewise-smooth solids [27].

A different approach to line-accessibility of smooth 3D objects (represented as B-rep with parametric surfaces) has been proposed in [24], which forgoes the partitioning of the entire 3D Euclidean space, and instead computes the boundaries between the accessible and inaccessible directions. This approach can also be possibly applied to computing accessibility with respect to external obstacles, while it does not enumerate the tangential line-accessible directions. While [23, 25, 26, 27, 24] focus on the problem of finding which *regions* of the geometry that are line-accessible, our work aims to classify the *tangent directions* that are line-accessible in each region.

In the context of flat-end cutting, the global collision detection is performed by bounding the tilt angle that guarantees globally collision free pose [28]. Having such a cone that bounds the admissible directions of the tool axis at every contact point, so called *iso-conic* partitioning of the input surface is applied and the contact curves are computed to minimize the variation of the bounding cone.

On a conceptual level, our paper follows [2] where the accessibility of a free-form surface is studied, in the context of 5-axis ball end milling. Motion of the tool axis is navigated by exploring a three dimensional search space. The first parameter is the parameter value of the input contact curve over the free-form surface, and the other two are a pair of angles determining locally the orientation of the tool's axis (inclination and tilt angles). The search space is being subdivided until a conservative estimate of admissible positions of the milling axis is found. This approach guarantees globally accessible and gouging-free configurations of the milling tool.

Unlike previous related work on line-cutting, this work offers global accessibility assurances for given cut-path lines. Herein and unlike [2], only tangential line contacts are of interest and considered.

The rest of the paper is organized as follows. Section 2 provides some necessary background on bounding differential properties of surfaces. Section 3 presents the accessibility algorithm itself, Section 4 employs the previous results toward motion planning in line-cutting while Section 5 portrays some examples. Finally, we conclude in Section 6 and discuss possible future work.

2. Background

Let $\mathcal{S}$ be a C^1, and sometimes C^2 as necessary, regular parametric surface in model $\mathcal{M}$, $\mathcal{S} \subset \mathcal{M}$, and let $\mathcal{S}$ be subdivided into small-enough patches, $\{S_i\}$ (The term "small-enough" is quantified by means of the patch size and the angular span of the normal directions later in Section 3). Given a single such patch, denoted S_0, in this section we consider how to bound the tangents and normals of S_0, in Section 2.1, and S_0's asymptotic directions, in Section 2.2.

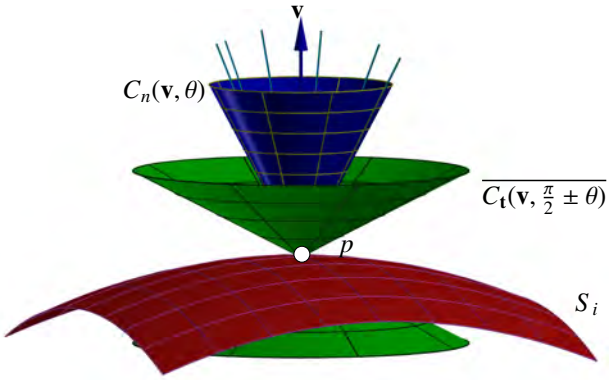


Figure 2: An illustration of the bounding normal cone $C_n(\mathbf{v}, \theta)$ (blue), and complementary tangent cone $\overline{C_{\mathbf{t}}(\mathbf{v}, \frac{\pi}{2} \pm \theta)}$ (the space between the green cones) of a surface patch S_i. Several arbitrarily chosen normal directions are shown inside the normal cone, as well as its axis $\mathbf{v}$.

2.1. Bounding the normals and tangents

Hence-after, we assume surfaces are oriented with their normal vectors pointing outside $\mathcal{M}$. The normal field of C^1 surface S_0 can be computed as

$$\mathbf{n}_{S_0}(u, v) = \frac{\partial S_0(u, v)}{\partial u} \times \frac{\partial S_0(u, v)}{\partial v}, \tag{1}$$

and it never vanishes, as S_0 is regular. Following [29], a tight bounding cones for $\mathbf{n}_{S_0}$ can be derived as $C_{\mathbf{n}}(\mathbf{v}, \theta)$, with axis $\mathbf{v}$ and angular span θ. See also Figure 2. S_0 is considered small enough if its bounding box is below some prescribed size d_{max} and θ is below some prescribed angular span θ_{max}.

Given $C_{\mathbf{n}}(\mathbf{v}, \theta)$, the normal cone of S_0, the tangent directions of S_0 are all bounded inside the double *complementary cone* $\overline{C_{\mathbf{t}}(\mathbf{v}, \frac{\pi}{2} \pm \theta)}$ (See also Figure 2):

Lemma 1. *Let S_0 be a regular, C^1-continuous, surface patch with a normal cone $C_{\mathbf{n}}(\mathbf{v}, \theta)$. Then, the tangent directions in S_0 are all bounded inside the double complementary cone $\overline{C_{\mathbf{t}}(\mathbf{v}, \frac{\pi}{2} \pm \theta)}$.*

Proof: *Let $\mathbf{u}$ be a tangent line in S_0 at $p \in S_0$, and let $\mathbf{n}$ be the normal vector of S_0 at p. $\mathbf{n} \in C_{\mathbf{n}}(\mathbf{v}, \theta)$ and hence $\angle(\mathbf{n}, \mathbf{v}) \leq \theta$. But $\angle(\mathbf{n}, \mathbf{u}) = \frac{\pi}{2}$, or $\frac{\pi}{2} - \theta \leq \angle(\mathbf{v}, \mathbf{u}) \leq \frac{\pi}{2} + \theta$, and $\mathbf{u} \in \overline{C_{\mathbf{t}}(\mathbf{v}, \frac{\pi}{2} \pm \theta)}$.* ■

2.2. Bounding the asymptotic directions

Given a hyperbolic patch S_0, we aim to construct geometric bounds on all tangents to S_0. Such a bound of S_0 is clearly going to intersect the neighboring sub-patches, in particular the 1-ring (8-connected) neighborhood of S_0. This 1-ring neighborhood consists of 8 surrounding patches that arise from the subdivision process and share with S_0 either a corner point or a boundary curve. See Figure 3.

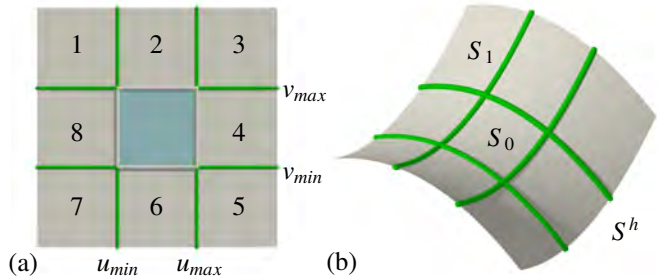


Figure 3: An 8-connected (1-ring) neighborhood. (a) The parameter domain $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$ (blue) of S_0 is surrounded by 8 neighboring parameter domains that are pre-images of S_i, $i = 1, \ldots, 8$. (b) The 3-space analogue. Patch S_0 and all its neighboring patches are assumed to be hyperbolic, and all together form S^h.

To remedy the difficulty of intersecting neighboring patches, we treat this 1-ring neighborhood differently. Consider a hyperbolic patch S_0, and let

$$S^h = S_0 \cup \{S_i \mid S_i \textit{ is hyperbolic} \text{ and } S_i \text{ 8-connected neighbor of } S_0\},$$

be the 1-ring hyperbolic neighborhood of S_0, forming a larger-patch, S^h. We seek to bound the ranges of directions in which S^h is guaranteed to have only negative normal curvature. That is, the surface curves "down" with respect to the tangent plane (See Figure 4 or, for example, Chapter 3.3 in [30]). Toward this end, we must bound all the asymptotic directions of S^h. For a fixed point $S^h(u, v)$, a tangent direction $aS^h_u(u, v) + bS^h_v(u, v)$ is asymptotic if

$$0 = a^2 L(u, v) + 2abM(u, v) + b^2 N(u, v), \tag{2}$$

where S^h_u denotes the first derivative of S^h with respect to u, $L(u, v), M(u, v), N(u, v)$ are the coefficients of the second fundamental form of S^h at (u, v) and $a, b \in \mathbb{R}$ (see Chapter 3.6 in [30]). We also impose a normalization constraint

$$1 = a^2 + b^2, \tag{3}$$

as (2) is a homogeneous quadratic form in (a, b) with a trivial solution $(0, 0)$. We further write:

$$\begin{aligned} L(u, v) &= S^h_{uu}(u, v) \cdot \bar{\mathbf{n}}^h(u, v) = S^h_{uu}(u, v) \cdot \tfrac{\mathbf{n}^h(u,v)}{|\mathbf{n}^h(u,v)|}, \\ M(u, v) &= S^h_{uv}(u, v) \cdot \bar{\mathbf{n}}^h(u, v) = S^h_{uv}(u, v) \cdot \tfrac{\mathbf{n}^h(u,v)}{|\mathbf{n}^h(u,v)|}, \\ N(u, v) &= S^h_{vv}(u, v) \cdot \bar{\mathbf{n}}^h(u, v) = S^h_{vv}(u, v) \cdot \tfrac{\mathbf{n}^h(u,v)}{|\mathbf{n}^h(u,v)|}, . \end{aligned} \tag{4}$$

where again S^h_{uu} denotes the second derivative of S^h with respect to u, etc., and $\cdot$ denotes the inner product. Further, $\bar{\mathbf{n}}^h(u, v)$ denotes the unit normal of S^h that can be computed as $\frac{\mathbf{n}^h(u,v)}{|\mathbf{n}^h(u,v)|}$, where $\mathbf{n}^h(u, v) = S^h_u \times S^h_v$. Assuming S^h is regular, $|\mathbf{n}^h(u, v)| \neq 0$ throughout the patch. Then, we can simplify Equations (4) by factoring out the (non zero) magnitude of the normal, $|\mathbf{n}^h|$. If S^h is a piecewise polynomial or rational surface, then this step results in the piecewise polynomial or rational constraint:

$$\begin{aligned} 0 &= a^2 L(u, v) + 2abM(u, v) + b^2 N(u, v) \\ &= a^2 S^h_{uu} \cdot \frac{\mathbf{n}^h}{|\mathbf{n}^h|} + 2abS^h_{uv} \cdot \frac{\mathbf{n}^h}{|\mathbf{n}^h|} + b^2 S^h_{vv} \cdot \frac{\mathbf{n}^h}{|\mathbf{n}^h|} \Rightarrow \\ 0 &= a^2 S^h_{uu} \cdot \mathbf{n}^h + 2abS^h_{uv} \cdot \mathbf{n}^h + b^2 S^h_{vv} \cdot \mathbf{n}^h. \end{aligned} \tag{5}$$

To further simplify the notation, let:

$$\begin{aligned} \hat{L}(u, v) &= S^h_{uu}(u, v) \cdot \mathbf{n}^h(u, v), \\ \hat{M}(u, v) &= S^h_{uv}(u, v) \cdot \mathbf{n}^h(u, v), \\ \hat{N}(u, v) &= S^h_{vv}(u, v) \cdot \mathbf{n}^h(u, v), \end{aligned}$$

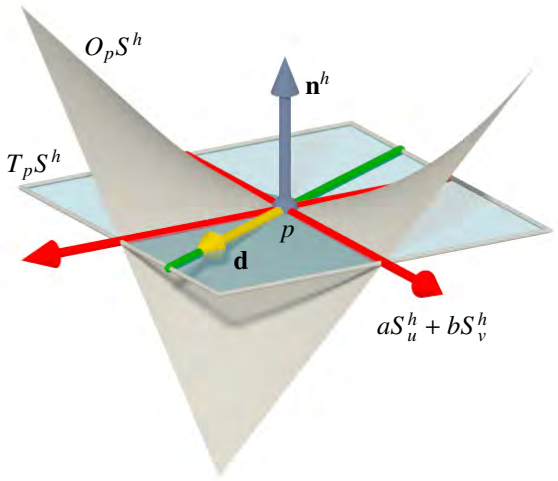


Figure 4: Second order analysis at a point. The osculating paraboloid O_pS^h of S^h at p is intersected with the tangent plane T_pS^h to define the two asymptotic directions (red). For a fixed point p, the coefficients $a, b \in \mathbb{R}$ are computed from Equation (6). A direction $\mathbf{d}$ (yellow) in which $\partial^2_{\mathbf{d}} S^h \cdot \mathbf{n}^h < 0$ defines a line (green) that is locally penetration-free with S^h.

then also (5) becomes

$$0 = a^2 \hat{L}(u, v) + 2ab\hat{M}(u, v) + b^2 \hat{N}(u, v). \tag{6}$$

In order to bound the asymptotic directions of S^h, we need to bound all the pairs (a, b) in the patch which satisfy Equation (6). We accomplish this in two steps:

1. First, we reduce the problem to finding the range of values of a single variable, rather than (a, b). This is explained in Section 2.3
2. Then, we solve the equation using a mix of symbolic computations over piecewise polynomial/rational B-spline functions and interval arithmetic. This is discussed in Section 2.4

Remark 1. So far we used the terms appearing in the second fundamental form ($\hat{L}(u, v)$, $\hat{M}(u, v)$, $\hat{N}(u, v)$) evaluated at a specific point (u, v), i.e., as *real numbers*. From now on, unless stated otherwise, we will use these terms as *bivariate functions* in (u, v) as our aim is to bound these entities for whole surface patches, not just as values at a point. We also omit the parameters and will write $\hat{M}$, etc.

2.3. A reduction to a single variable problem

Due to homogeneity, if (a, b) is a solution to Equation (6), then so is $(\lambda a, \lambda b)$, for $\lambda \in \mathbb{R}$. We seek non trivial solutions and hence assume $\lambda \neq 0$. Solving (6) satisfying the normalization constraint (3) gives two non-linear equations that need to be simultaneously solved. However, the problem is essentially a single-variable problem as the family of (normalized) tangent directions is one-parametric. Therefore, if possible, we solve only a univariate case, in particular we do:

1. Remember that $\hat{N}$ is a B-spline function if S^h is. If all the coefficients of $\hat{N}$ are all positive (or all negative), then $\hat{N} > 0$ (or $\hat{N} < 0$), by the convex hull property. Inspecting Equation (6), if $\hat{N} > 0$, then $a \neq 0$. If (a, b) is a solution and $a \neq 0$, then $(1, \frac{b}{a})$ is also a solution. We can then restrict a to $a = 1$, and solve for b in Equation (6).

2. Otherwise, if $\hat{N}(u,v) = 0$ for some (u,v) in the domain, we examine if $\hat{L}$ can be zero. If $\hat{L} \neq 0$, we restrict b to $b = 1$, and solve for a, in Equation (6).
3. Finally if $\hat{N} = \hat{L} = 0$ for some (in general different) pairs of parameters, we solve a bivariate system consisting of Equations (6) and (3) and unknowns a and b, for example, using [31].

2.4. Solving the single variable problem

We now present the complete formulations only for the $a = 1$ case, while the other one follows a similar path. We emphasize that the patch S^h, see Figure 3, is assumed hyperbolic. For case 1 ($a = 1$), we obtain

$$b = \frac{-2\hat{M} \pm \sqrt{4\hat{M}^2 - 4\hat{L}\hat{N}}}{2\hat{N}} = \frac{-\hat{M}}{\hat{N}} \pm \sqrt{\frac{\hat{M}^2 - \hat{L}\hat{N}}{\hat{N}^2}}, \qquad (7)$$

where the term under the square root is guaranteed to be positive everywhere (due to S^h being hyperbolic everywhere). The rational expressions $\frac{-\hat{M}}{\hat{N}}$ and $\frac{\hat{M}^2-\hat{L}\hat{N}}{\hat{N}^2}$ are computed using symbolic (product and difference) operations on rational B-spline functions [32], which are performed with machine-precision. Since the square root in $\sqrt{\frac{\hat{M}^2-\hat{L}\hat{N}}{\hat{N}^2}}$ cannot be represented as a rational B-spline function, we continue solving for the intervals where the asymptotic directions may reside using interval arithmetic [33]. Exploiting the convex hull property of B-spline function, we find the minimum and maximum values of those rational expressions, $[\frac{-\hat{M}}{\hat{N}}_{min}, \frac{-\hat{M}}{\hat{N}}_{max}]$ and $[\frac{\hat{M}^2-\hat{L}\hat{N}}{\hat{N}^2}_{min}, \frac{\hat{M}^2-\hat{L}\hat{N}}{\hat{N}^2}_{max}]$, and represent them as intervals. The rest of the formula is computed by interval arithmetic, resulting in a solution for b: an interval $[b_{min}, b_{max}]$.

The asymptotic directions of S^h in Euclidean space all have a representation of the form: $aS^h_u + bS^h_v$. Herein, however, a and/or b represent ranges of values, and they are parameterized:

1. $a(r) = 1,\ b(r) = r \cdot b_{min} + (1-r)b_{max}$, for case 1.
2. $a(r) = r \cdot a_{min} + (1-r)a_{max},\ b(r) = 1$, for case 2.
3. $a(r)$ and $b(r)$ as univariate solutions for the two Equations (3) and (6) and two unknowns, a and b.

For each univariate solution, we obtain the trivariate:

$$T(u,v,r) = a(r)S^h_u(u,v) + b(r)S^h_v(u,v),$$

for each computed bound on the asymptotic directions of S^h. Since (7) returns, in general, two solutions (i.e. two interval ranges), the trivariate $T(u,v,r)$ typically consists of two disconnected components. Consequently, the bounding region of the (trivariate) asymptotic directions in 3-space typically consist of two bounding double-cones (see Figure 5 (a)). Finally, the bounding cones are projected onto a plane, $\mathcal{T}_{S^h}$, orthogonal to $\mathbf{v}$, the axis of the normal cone of S^h, $C_{\mathbf{n}}(\mathbf{v},\theta)$. The 2D situation in $\mathcal{T}_{S^h}$ is shown in Figure 5 (b).

The 2D angular spans between the projected cones on $\mathcal{T}_{S_0}$ represent the ranges of directions in which the normal curvature is either entirely positive, or entirely negative. All that remains is to sample each of those angular spans at a single point, to find the spans with the negative normal curvature, or the spans that are locally line-accessible in S_0.

3. The tangent line-accessibility analysis algorithm

Consider a C^1 regular parametric surface $\mathcal{S}(u,v)$, representing some (part of) 3D model $\mathcal{M}$, and let $\mathbf{n}_{\mathcal{S}}(u,v)$ be the (unnormalized) normal field of S. Given some tangent line to $\mathcal{S}$, L_t, we define the global *tangent line-accessibility problem*, or simply *line-accessibility problem*, as follows:

> **Definition 1.** *Given an infinite line tangent to surface $\mathcal{S}$, L_t, at $p \in \mathcal{S}$, L_t is globally* line-accessible *if it intersects model $\mathcal{M}$ at no location other than p.*

In order to line-cut $\mathcal{M}$, two major tasks must be fulfilled:

1. First, a set of covering cut-paths must be formulated for $\mathcal{M}$ so that line-cutting $\mathcal{M}$ by sweeping L_t along these cut-paths will yield a good approximation to $\mathcal{M}$.
2. The second task is: given a cut-path curve, C_p, find the best globally accessible tangential direction, if any, to line-cut with, while moving along C_p. The focus of this work is about this second accessibility task.

Given a C^2 regular parametric surface $\mathcal{S}(u,v)$, one can determine, if it is elliptic (convex or concave) or hyperbolic, or a mix of the above, by computing the parabolic curves in $\mathcal{S}$ [34]. Following [35], one can compute the Gaussian curvature field of B-spline surface $\mathcal{S}(u,v)$, as a scalar B-spline function $K(u,v)$. The zero set of K, $K(u,v) = 0$, delineates the parabolic curves of S, if any, and these parabolic curves, in turn, divide $\mathcal{S}(u,v)$ into hyperbolic and elliptic zones.

Consider a prescribed cut-path curve, $C_p \subset \mathcal{S}$, along which the tangential cut-line moves on $\mathcal{S}$. We seek to detect, if exists, tangential direction(s) to $\mathcal{S}$ that are line-accessible, for all $p \in C_p$. Further, we seek to employ optimal tangential directions, as much as possible, directions where the cutting line best fit $\mathcal{S}$. Given a hyperbolic location $p \in C_p$, the optimal tangential directions are typically the asymptotic directions [16]. Given a convex elliptical region, the best tangential directions are the principal directions with the minimal (in absolute value) normal curvature. Finally, for parabolic locations, the direction with zero normal curvature is to be preferred while concave elliptical regions are locally (and globally) line-inaccessible. Herein, we seek to verify that these preferred directions are indeed line-accessible, and if not, we aim to select those that are as close as possible.

Finding all the valid (tangential) cut-lines for all points on $\mathcal{S}$ is an intractable task. Instead, we employ a conservative discretization of the problem (See Figure 6):

1. As discussed in Section 2, $\mathcal{S}$ is subdivided into small yet finite patches, as set $\{S_i\}$, until the size and normal deviations in all S_i (if not singular) are small, and below d_{max} and θ_{max}, respectively. In addition, to ensure that the subdivision process is finite, we stop the subdivision for patches smaller than d_{min}. Some examples will be presented in Section 5.
2. For each such small patch S_i:
   (a) Classify S_i as a convex, concave, hyperbolic, or a mixed patch.
   (b) Define a set of n tangent vectors $\mathbf{u}_{ij},\ j = 1...n$, forming an angular division of a (central) tangent plane of S_i. The tangent plane is defined by the axis of the cone $C_{\mathbf{n}}(\mathbf{v},\theta)$, as described in Section 2.1. For an illustration of the angular division, see Figure 6 (a).

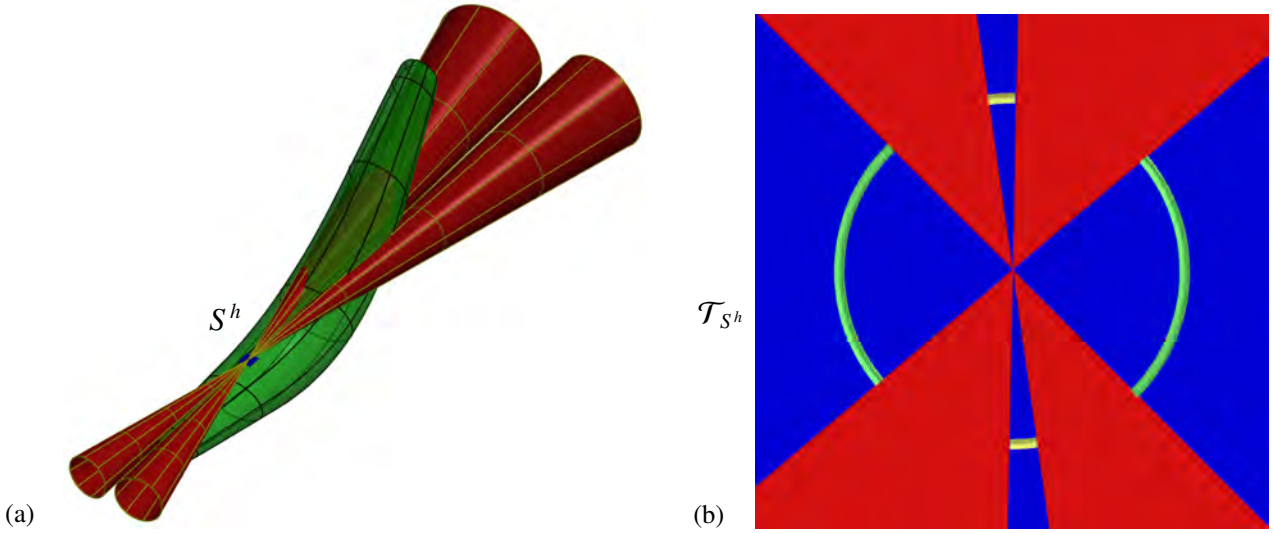


Figure 5: (a) An illustration of the bounding cones (in red) of the asymptotic direction of a small hyperbolic region S^h (in blue). (b) A zoomed-in orthogonal projection of (a) in the direction of the axis of the bounding normal cone of S^h onto the tangent plane $\mathcal{T}_{S^h}$ is shown. The range of locally (and conservatively) accessible directions is marked in green, presenting negative normal curvatures. The range of directions marked in yellow is also outside the asymptotic directions bounding cones, however, the normal curvature in these directions is positive and hence these directions are locally inaccessible. The directions in the green area are locally accessible; global accessibility test must still be conducted.

(c) Compute n bounding frustums, $\mathcal{F} = \left\{F_{S_i,\mathbf{u}_{ij}}\right\}_{j=1}^{n}$, of tangent lines of $\{S_i\}$ in approximate direction $\mathbf{u}_{ij}$. These frustums overlap and are large enough to (together) include all possible tangent lines of S_i. See Figure 6 (b)-(d).

(d) If frustum $F_{S_i,\mathbf{u}_{ij}}$ intersects with some portion of surface patch $S_k \subset \mathcal{M}$, $k \neq i$, more rigorous tests are applied to determine if any tangents of S_i inside $F_{S_i,\mathbf{u}_{ij}}$ are indeed obstructed by S_k. If such an obstructing patch is detected, even after rigorous tests, we conservatively declare all tangent lines in $F_{S_i,\mathbf{u}_{ij}}$ as inaccessible. Otherwise, $F_{S_i,\mathbf{u}_{ij}}$, and all the tangent lines of S_i it contains, is defined as line-accessible. This task is, by far, the most challenging task.

In the rest of this section, we will describe our proposed obstruction tests in details, and demonstrate how to achieve conservative accessibility classifications for each of the frustums. Hence-after and without loss of generality, we consider the line-accessibility of one surface patch in $\{S_i\}$, that is denoted S_0. Note that the set $\{S_i\}$ covers all surfaces of model $\mathcal{M}$ and hence accessibility with respect to $\{S_i\}$ reflects on accessibility with respect to $\mathcal{M}$.

3.1. The main algorithm

After the initial steps discussed above, we have, for S_0, its normal and tangent cones, $C_n(\mathbf{v}, \theta)$ and $\overline{C_{\mathbf{t}}(\mathbf{v}, \frac{\pi}{2} - \theta)}$, respectively. The tangent space of S_0, bounded by $\overline{C_{\mathbf{t}}(\mathbf{n}, \frac{\pi}{2} - \theta)}$, is discretized by constructing the set of frustums $\left\{F_{S_0,\mathbf{u}_{0j}}\right\}_{j=1}^{n}$, using the following construction rules (Recall Figure 6):

1. The axis of $F_{S_0,\mathbf{u}_{0j}}$ is $\mathbf{u}_{0j}$.
2. The smaller base of $F_{S_0,\mathbf{u}_{0j}}$ is aligned orthogonally to $\mathbf{u}_{0j}$, behind S_0. The size of the smaller base of $F_{S_0,\mathbf{u}_{0j}}$ is chosen so that $F_{S_0,\mathbf{u}_{0j}}$ is a bounding volume of S_0.
3. The vertical opening angle of $F_{S_0,\mathbf{u}_{0j}}$, in $\mathbf{v}$, is set to θ, following the computed vertical opening angle of the complementary tangent cone.

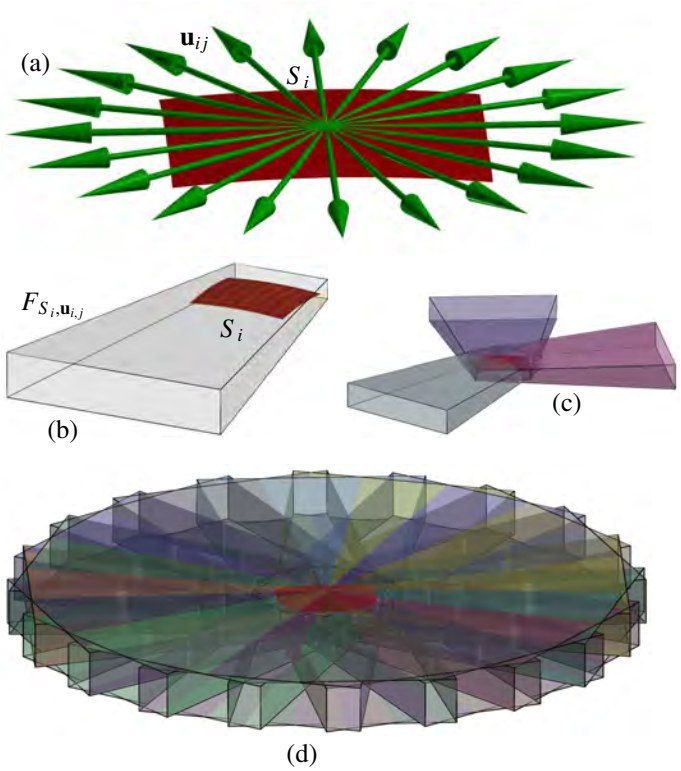


Figure 6: (a) A division of the tangent plane of surface S_i (in red) to n tangents $\mathbf{u}_{i,j}$, $j = 1, ..., n$. (b) An example of one frustum $F_{S_i,\mathbf{u}_{i,j}}$, for one surface patch S_i and one tangent $\mathbf{u}_{i,j}$. (c) Three samples of frustums of the tangential space of surface patch S_i. (d) a covering set of frustums of the tangent space of S_i, using all $\mathbf{u}_{i,j}$, $j = 1, ..., n$. Note frustums must overlap to guarantee a full coverage of the tangent space.

4. The radial opening angle of $F_{S_0,\mathbf{u}_{0j}}$ is $2\pi/n$, where n is the number of frustums.
5. The depth of frustum $F_{S_0,\mathbf{u}_{0j}}$ is chosen so the frustum will extend beyond the bounding box of $\mathcal{M}$.

Definition 2. *Hence after, we use the term* locally accessibly direction *to denote a tangent direction in a hyperbolic patch S_0, that (following the asymptotic directions analysis in Section 2.2) was found to have only negative normal curvature. Note that for a convex patch S_0 all directions are locally accessible.*

Assumption 1. *From Section 2.2, conservative bounds on the asymptotic directions of S_0 are also available. That said, consider a tangent line L_t at point p to S_0, in a direction that was detected to be locally accessible. We assume L_t intersects S_0 at no location except p. While for a convex patch this assumption holds, for a hyperbolic patch, it is not difficult to synthesize an example in which L_t will penetrate S_0 in another location q. That said, this penetration error is minute and clearly depends on the size of the (small) patch and the differential properties' variations in S_0 that we already bound. Hence, this penetration error is also bounded and a bound for this error is computed and discussed in Appendix A.*

Further, for simplicity of the forthcoming algorithms, we also assume tangent line L_t to S_0 will not penetrate the 1-ring (8-connected) neighborhood of S_0. Again, for similar reasons, the error can be bounded.

While in convex S_0 patches, all tangent lines are guaranteed not to penetrate S_0, and for hyperbolic cases, Section 2.2 provides a way of eliminating directions with non-negative normal curvature, we have no such guarantee for mixed patches. Our algorithm will therefore aim to determine which tangent directions of mixed S_0 patches are globally line-accessible, while due to assumption 1, we can bound the potential error for S_0 itself and its immediate 1-ring neighborhood.

All that said, and excluding S_0 itself and its 1-ring neighborhood, we are now ready to verify the tangent directions in S_0, that are globally accessible, against all other patches in $\mathcal{M}$.

The accessible directions of a surface patch S_0 are found by the main algorithm, Algorithm 1. Given all patches in $\mathcal{M}$, the algorithm, in Line 1 (using $NoObstructGlbl(S_0)$, Algorithm 2), first filters out, other patches that are trivially irrelevant to the question of line-accessibility to S_0. Then, for each sampled direction $\mathbf{u}_{0j}$, Algorithm 1 constructs a frustum $F_{S_0,\mathbf{u}_{0j}}$, and checks, in Line 4, if this direction is locally accessible and if not purges this frustum (in Line 5).

If $F_{S_0,\mathbf{u}_{0j}}$ is locally accessible, we find (in Line 7) the subset of patches, $\mathcal{P}_{S_0,\mathbf{u}_{0j}}$, that intersect with $F_{S_0,\mathbf{u}_{0j}}$, and hence potentially obstruct tangent lines in $F_{S_0,\mathbf{u}_{0j}}$. We use a bounding volume hierarchy (BVH), that contains all surface patches $S_i \subset \mathcal{M}$, to efficiently find the subset of patches that intersect with $F_{S_0,\mathbf{u}_{0j}}$.

Then, for all $S_i \in \mathcal{P}_{S_0,\mathbf{u}_{0j}}$, Algorithm 1 invokes Algorithm 3 to determine if a patch S_i, actually obstructs line-accessibility to S_0 in frustum $\mathcal{F}_{S_0,\mathbf{u}_{0j}}$. If Algorithm 3 finds at least one patch S_i which obstructs accessibility to a tangent line in $F_{S_0,\mathbf{u}_{0j}}$, frustum $F_{S_0,\mathbf{u}_{0j}}$, with all its (tangential) directions, is conservatively decreed inaccessible.

Algorithm 2 performs some simple filtering of patches that are immediately identified as non-obstructing for the line-accessibility of patch S_0. Let $\hat{C}_t(S_0)$ be a bounding volume of the Minkowski sum of the complementary tangent cone of S_0, $\overline{C_t}(S_0)$, with S_0 (See Line 1). Then, in Line 2 of the algorithm,

Algorithm 1 The main line-accessibility algorithm: Finding the subset of frustums of $S_0 \subset \mathcal{M}$ with line-accessibility.

Input:
S_0: a surface patch from a surface $\mathcal{S} \subset \mathcal{M}$;
$\{S_k\}$: a set of all surface patches from $\mathcal{M}$, excluding S_0;
n: the number of frustums to discretize tangent space of S_0.
R: number of refinements iterations to apply - See Algorithm 3;

Output:
A subset of frustums, $\mathcal{F}^A = \left\{F_{S_0,\mathbf{u}_{0j}}\right\}$, in which S_0 is line-accessible, if any;

Algorithm:

1: $\mathcal{I} := NoObstructGlbl(S_0, \{S_i\})$; // Initial set of patches that tangent line-obstruct S_0 in no frustum. See Algorithm 2.
2: $\mathcal{F}^A := \left\{F_{S_0,\mathbf{u}_{0j}}\right\}_{j=1}^{n}$; // All frustums around S_0. Their construction depends on S_0, as well as the parameter n.
3: **for all** $F_{S_0,\mathbf{u}_{0j}} \in \mathcal{F}$ **do**
4: **if** $F_{S_0,\mathbf{u}_{0j}}$ is locally inaccessible **then** // according to asymptotic directions analysis in Section 2.2.
5: $\mathcal{F}^A := \mathcal{F}^A - \{F_{S_0,\mathbf{u}_{0j}}\}$;
6: **else**
7: $\mathcal{P}_{S_0,\mathbf{u}_{0j}} := \{S_k |\, F_{S_0,\mathbf{u}_{0j}} \cap S_k \neq \phi\} - \mathcal{I}$; // Set of potentially obstructing patches that intersect frustum $F_{S_0,\mathbf{u}_{0j}}$.
8: **for all** $S_i \in \mathcal{P}_{S_0,\mathbf{u}_{0j}}$ **do**
9: **if** $TestObstructInFrustum(S_0, F_{S_0,\mathbf{u}_{0j}}, S_i, R)$ **then** // see Algorithm 3
10: $\mathcal{F}^A := \mathcal{F}^A - \{F_{S_0,\mathbf{u}_{0j}}\}$;
11: **break**; // Once detected as obstructed, $F_{S_0,\mathbf{u}_{0j}}$ can conservatively be purged.
12: **end if**
13: **end for**
14: **end if**
15: **end for**
16: **return** $\mathcal{F}^A$; // All the accessible frustums, if any.

every other surface patch S_k is tested for inclusion in $\hat{C}_t(S_0)$. Every patch S_k that does not intersect with $\hat{C}_t(S_0)$ is immediately marked as non-obstructing.

Algorithm 2 then performs two other tests to mark additional cases as irrelevant. In Lines 5 and 6 of the algorithm, patches in a convex neighborhood are also marked as non-obstructing. Finally, in Lines 7 and 8, patches that are in the 1-ring neighborhood of S_0 are also marked as non-obstructing, following assumption 1.

Algorithm 3 does a more precise (and more expensive) test for the possibility of S_k obstructing line-accessibility from S_0. Toward this end, we present the following:

Lemma 2. *Consider parametric patch $S_0(u, v)$ and another patch $S_k(r, t)$, and let L_t be some line tangent to S_0. S_k can line-occlude L_t only if $\exists(u, v, r, s)$ such that,*

$$O(u, v, r, s) = (S_0(u, v) - S_k(r, t)) \cdot N_0(u, v) = 0, \quad (8)$$

where N_0 is the normal field of $S_0(u, v)$.

Algorithm 2 NoObstructGlbl(S_0): Computes a set of patches that obstruct line-accessibility to S_0 in *no* tangent direction.

Input:
S_0: a surface patch from the input surface $\mathcal{S} \subset \mathcal{M}$;
$\{S_k\}$: a set of all surface patches from $\mathcal{M}$, excluding S_0;

Output:
$\mathcal{I}$: a set of patches from $\mathcal{M}$ which are guaranteed to obstruct no line-accessibility to S_0;

Algorithm:

1: $\hat{C}_{\mathbf{t}}(S_0) := MinkowskiSum(\overline{C_{\mathbf{t}}}(S_0), S_0)$;
2: $\mathcal{I} := \{S_k \mid \hat{C}_{\mathbf{t}}(S_0) \cap S_k = \phi\}$; // Initial set of no-obstruction patches - no-overlap with expanded (via Minkowski sum) tangent complementary cone of S_0;
3: $C :=$ if S_0 is convex, the set of convex adjacent patches to S_0, forming an angular span of up to 180 degrees (preventing self-occlusions);
4: **for all** $S_i \in \{S_k\} - \mathcal{I}$ **do**
5: **if** S_0 is convex and $S_i \in C$ **then**
6: $\mathcal{I} := \mathcal{I} \cup \{S_i\}$;
7: **else if** S_0 and S_i are adjacent **then**
8: $\mathcal{I} := \mathcal{I} \cup \{S_i\}$; // By Assumption 1
9: **end if**
10: **end for**
11: **return** $\mathcal{I}$;

Proof: By contradiction. Assume S_k obstructs line-accessibility from S_0 and yet Equation (8) never vanish. Then there exist a point p on S_0 such that the line L_t tangent to S_0 at p intersects with S_k at point q. But then S_k is intersecting with the tangent plane of S_0 at p that contains L_t. $L = S_0(u_p, v_p) - S_k(r_q, t_q)$ but because L_t is tangent to S_0 we have $(S_0(u_p, v_p) - S_k(r_q, t_q), N_0(u_p, v_p) = 0$, a contradiction. ■

Lemma 2 presents a way to precisely determine if patch S_k can line obstruct S_0. Given (regular) B-spline surfaces S_0 and S_k, and the unnormalized normal field N_0 that is also a B-spline vector function, symbolically compute [32] the 4-variate function $O(u, v, r, s)$ in Equation (8) as a Bspline function. If all coefficients of O are positive or all coefficients are negative we are ensured line-accessibility from S_0 is not occluded by S_k.

We apply additional inequality constraints to Constraint (8) to further rule out cutting directions which are irrelevant for deciding whether S_0 is occluded by S_k in frustum $F_{S_0,\mathbf{u}_{0j}}$. Let P_L, P_R be the two normals of the side planes of $F_{S_0,\mathbf{u}_{0j}}$ (oriented to point inside the frustum). A cutting direction $L = S_0(u_p, v_p) - S_k(r_q, t_q)$ is inside $F_{S_0,\mathbf{u}_{0j}}$ if $Q_L(u, v, r, s) = L(u, v, r, s) \cdot P_L \geq 0$ and $Q_R(u, v, r, s) = L(u, v, r, s) \cdot P_R \geq 0$. Therefore, S_0 is occluded by S_k if for some parameters u, v, r, s in the common domain of S_0, S_k all following conditions hold:

1. $O(u, v, r, s) = 0$.
2. $Q_L(u, v, r, s) \geq 0$.
3. $Q_R(u, v, r, s) \geq 0$.

If one can precisely determine if such u, v, r, s exist, one can precisely determine if S_k can line-occlude S_0. While methods to determine minima and zeros of spline do exists they are expensive. Instead, we reexamine the signs of all the coefficients of $O(u, v, r, s), Q_L(u, v, r, s), Q_R(u, v, r, s)$ and apply several successive subdivisions the common domain of S_0, S_k to improve the accuracy as necessary.

Algorithm 3 summarizes this process. Given S_0 and S_k, the relevant frustum $F_{S_0,\mathbf{u}_{0j}}$ and the number of allowed refinement iterations over O, the algorithm first finds the minimal tensor product sub-region of S_k that is in frustum $F_{S_0,\mathbf{u}_{0j}}$. It can happen that the full S_k might line-obstruct S_0 but the portion of S_k in frustum $F_{S_0,\mathbf{u}_{0j}}$ might not. See also Figure 7. Then, Algorithm 3 goes and compute O for S_k and S_k^c and examines its coefficients. If all coefficients positive or all negative, the algorithms terminates with the recognition that no obstruction can occur. Otherwise, R successive refinement attempts are made over O to achieve a tighter and tighter bound.

Algorithm 3 TestObstructInFrustum($S_0, F_{S_0,\mathbf{u}_{0j}}, S_i, R$): Determines if S_i obstructs line-accessibility to S_0 in frustum $F_{S_0,\mathbf{u}_{0j}}$.

Input:
S_0: a surface patch from input surface $\mathcal{S} \subset \mathcal{M}$;
$F_{S_0,\mathbf{u}_{0j}}$: a frustum of S_0 in $\overline{C_{\mathbf{t}}(\mathbf{n}, \frac{\pi}{2} \pm \theta)}$;
S_k: another surface patch from $\mathcal{M}$;
R: number of refinements iterations to apply;

Output:
True if S_i obstructs line-accessibility to S_0 within $F_{S_0,\mathbf{u}_{0j}}$,
False otherwise;

Algorithm:

1: $S_k^c := Clip(S_k, F_{S_0,\mathbf{u}_{0j}}) :=$ Minimal tensor product subdomain of S_k in $F_{S_0,\mathbf{u}_{0j}}$;
2: $\mathcal{R} := \{(O(u, v, r, s), Q_L(u, v, r, s), Q_R(u, v, r, s))\}$ // triplets of:
$O(u, v, r, s) = \left\{(S_0(u, v) - S_k^c(r, t)) \cdot N_0(u, v)\right\}$,
$Q_L(u, v, r, s) = (S_0(u, v) - S_k^c(r, t)) \cdot P_L$,
$Q_R(u, v, r, s) = (S_0(u, v) - S_k^c(r, t)) \cdot P_R$.
3: **for** $i = 1$ to R **do** // Subdivision cycles over u, v, r, s
4: $IsObstructed$:=**False**;
5: **for** $(O, Q_L, Q_R) \in \mathcal{R}$ **do**
6: **if** $0 \in BBox(O)$ and $max(Q_L) \geq 0$ and $max(Q_R) \geq 0$ **then** // Examining the coefficients of $O, Q_L Q_R$
7: $IsObstructed$:=**True**;
8: **end if**
9: **end for**
10: **if** $IsObstructed$ **then**
11: **return True**;
12: **end if**
13: $\mathcal{R}$:= Subdivide all triplets $(O, Q_L, Q_R) \in \mathcal{R}$;
14: **end for**
15: **return** $IsObstructed$;

4. Using the line-accessibility information for motion planning

To demonstrate how the line-accessibility information, produced by the presented algorithm, can be used in motion planning, we developed a simple cutting scheme which takes line-accessibility into account, and used it in the fabrication process. Because the cutting line is considered infinite, opposite frustums

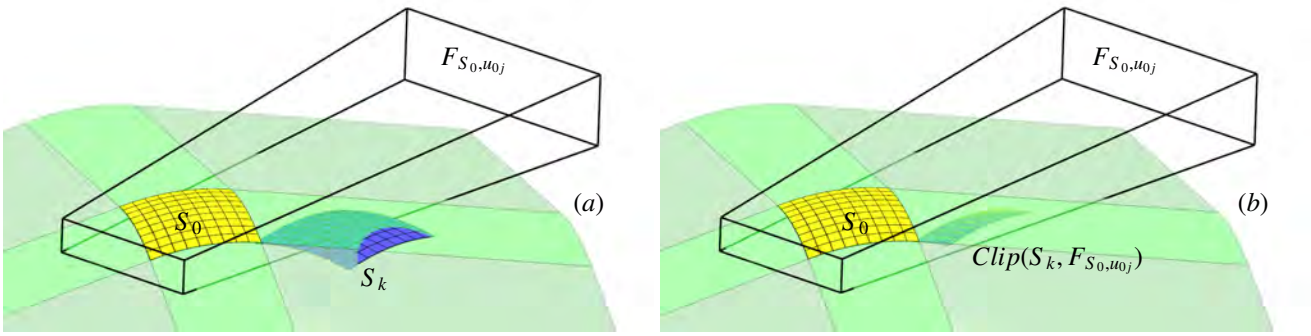


Figure 7: In (a), a patch S_k can intersect with some frustum $F_{S_0,u_{0j}}$ and also line-obstruct S_0. Yet its intersection portion with $F_{S_0,u_{0j}}$ might not line-obstructing S_0. By clipping S_k to the minimal tensor product sub-region that is in $F_{S_0,u_{0j}}$, in (b) as $Clip(S_k, F_{S_0,u_{0j}})$, $Clip(S_k, F_{S_0,u_{0j}})$ is detected as not line-obstructing and hence is not occluding for frustum $F_{S_0,u_{0j}}$. The green extensions depicts the tangent line-accessibility from S_0.

are paired together: $F_{S_i,\mathbf{u}_{ij}}, F_{S_i,\mathbf{u}_{ik}}$, such that $\mathbf{u}_{ij} = -\mathbf{u}_{ik}$. Given n frustums around the tangent plane, we have $n/2$ such paired frustums.

Consider a cut-path parametric curve $C_p(t)$ on model $\mathcal{M}$. In our experiments, we used a covering algorithm of the model $\mathcal{M}$ by isoparametric curves, following [**?**]. However, the line-accessibility analysis described in the previous sections is applicable for any sets of cut-paths. The following succession of operations is performed:

1. Find the sequence of surface patches $S_0, S_1, ..., S_{m-1}$ through which $C_p(t)$ passes. Recall that for each patch, we know which frustums of tangent directions $F_{S_i,\mathbf{u}_{ij}}$ are accessible, as determined by Algorithm 1.
2. Create a 2D table with $n/2$ rows and m columns. The i-th column represents the accessible tangent directions of S_i, and the j-th cell in each column represents the paired frustums, $F_{S_i,\mathbf{u}_{ij}}$ and its opposite frustum. Note each column is periodic, as the angular domain is.
3. Recall that for hyperbolic regions, directions of cuts close to the asymptotic directions, and for convex regions directions of cuts close to the smaller principal curvature (in absolute value) are preferred. Hence, assign each (i, j) cell in the 2D table with a cost that relates to the preferred minimal (in absolute value) normal curvature, in the direction of the axis $\mathbf{u}_{ij}$ of $F_{S_i,\mathbf{u}_{ij}}$.
4. Find a minimum-cost path from the first column (of S_0) of the table to the last column (of S_{m-1}), using a BFS (breadth-first search), and under the following rules:
   (a) The path can traverse only cells for which there exist paired frustums that are line-accessible.
   (b) The path can progress from cell (i, j) only to $(i, j + 1)$, $(i, j - 1)$, or $(i + 1, j)$. That is, to the next or previous frustum (within the periodicity of the columns), or to the next surface patch, along $C_p(t)$.

   If no such path exists, the entire cutting curve is decreed inaccessible.
5. In the computed minimum-cost path, each surface patch S_i along $C_p(t)$ (represented by a column) is traversed via one or more cells. Let t_{i0} and t_{i1} be the parameter values of $C_p(t)$ where it enters and exits S_i, respectively. Set the line-cutting directions at t_{i0} and t_{i1} to be the best tangent direction in the frustums corresponding to the cell through which the path enters and exits (respectively) the column. The rest of the cutting tangents for each $t \in [t_{i0}, t_{i1}]$ will be assigned by computing a spherical-linear interpolation (Slerp [36]) of t_{i0}, t_{i1}, and projecting it onto the surface normal at parameter value t.

Remark 2. While adjacent surface patch do not share the same tangent plane, the tangent planes are very similar. Further, since the crossing point in $C_p(t)$ on the boundary shared by S_i and S_{i+1} is in both accessible frustums and the geometry is assumed C^1, this transition is always feasible.

5. Examples

We now demonstrate the different algorithms presented as part of this work. Figure 8 classifies the different, flat enough and small enough, patches of four different models into elliptic convex (green), hyperbolic (yellow) and mixed (magenta), by computing the Gaussian curvature field of these four spline surfaces, as a spline scalar field.

Given a classified patch, frustums which discretize the direction for line-accessibility are constructed. Figure 9 shows two models and the whole set of parallel (that do not overlap, for clarity) frustums of two patches in these models. Green colored tips in the frustums denote an accessible direction whereas red inaccessible. Note also that for an elliptic patch all the 360 degrees is populated with frustums where as in a hyperbolic patch on the valid regions between the asymptotic directions.

As we stated, treatment of adjacent patches is quite challenging. Given patch S_0, if we assume no tangent line penetrate S_0 or its 1-ring 8 neighbors in any other location, all patches of the Banana and Vase model in Figure 10 right are found accessible (green). Otherwise, as can be seen in Figure 10 left, some patches that are accessible are not found as such (red).

In more complex models, such as the duck in Figure 11, our algorithm fails to guarantee accessible cutting directions in some of the patches. This occurs most often at or around mixed patches (patches which contain both elliptic and hyperbolic regions). However, one should note that while red patches are marked as inaccessible to wire-cutting, in practice, we expect that infinite-wire cutting of patches in their close proximity, or immediate adjacent patches, is likely to cut above the red patches as well. If the red patches are fairly isolated, the result is likely to be a complete part with only small gaps (but not gouging) between the final cut artifact and these red patches.

In our final example, we used a rough covering of our input vase surface with isoparametric curves. Along with the tangent line-accessibility analysis in Section 3 and the motion planning described in Section 4, we create the line-accessible

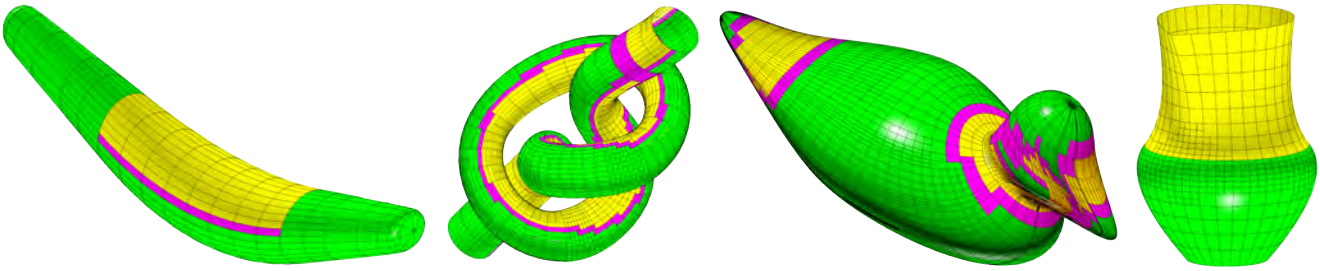


Figure 8: Four different freeform spline models, divided into small enough and flat enough patches, only to be classified as elliptic convex (green), hyperbolic (yellow) and mixed (magenta) patches.

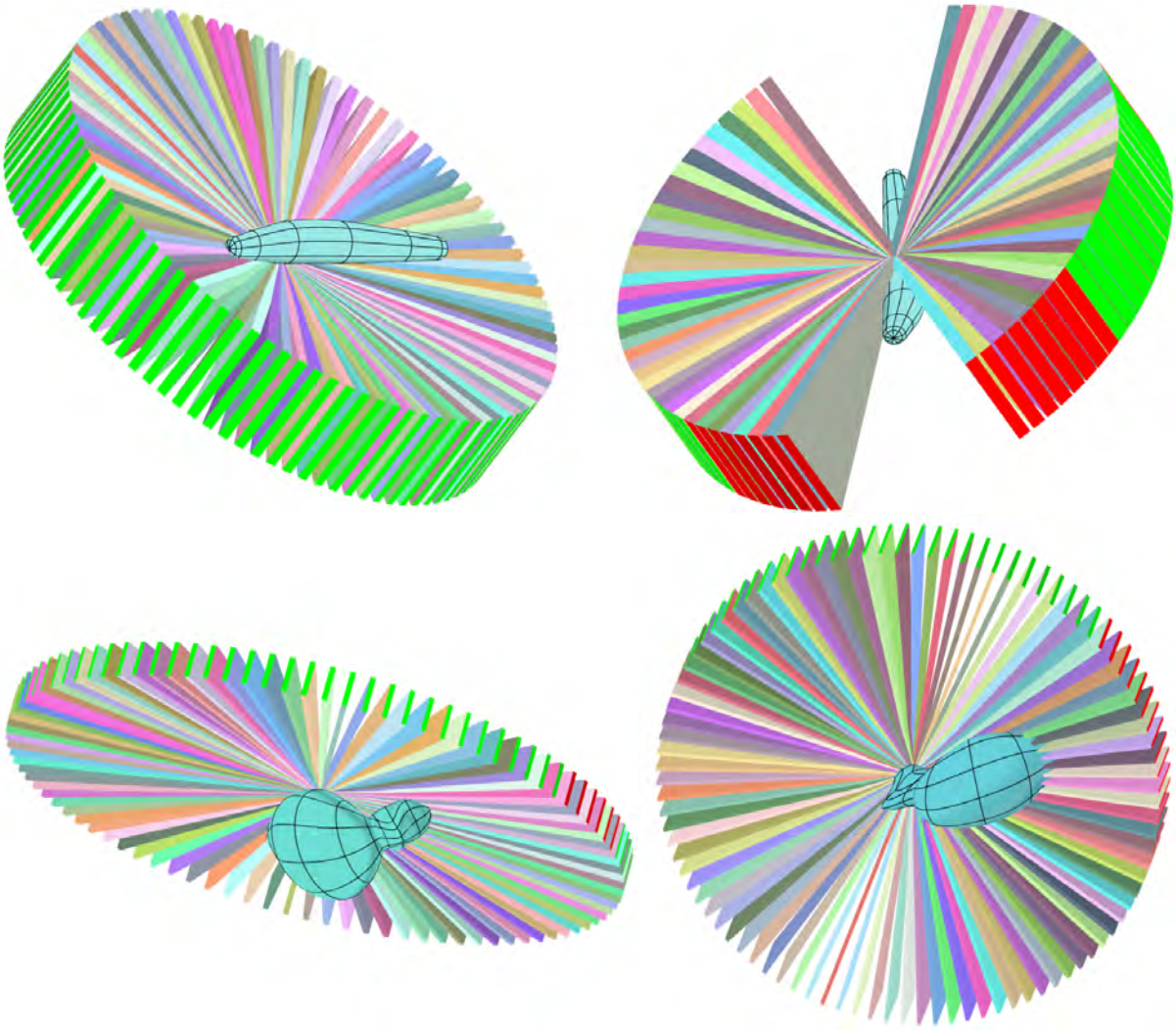


Figure 9: Two models of a banana (top) and a duck (bottom) are presented (in cyan), along with two patches with all their frustums. Note some frustums are found inaccessible (red tip) due to global inaccessibility. Accessible frustums are marked with green tips.

cut-paths that is shown in Figure 12. This cutting was used to drive the robot shown in Figure 1 (b) yielding the final cut part, from Styrofoam, shown in Figure 1 (c). A video presenting the robot cutting the Styrofoam is available in `https://youtu.be/THwKUoJmEA8`

6. Discussion and future work

This work has presented a complete set of algorithms that enable one to not only detect line-accessible zones in a given model $\mathcal{M}$ but also to conservatively compute all globally accessible directions, and from that derive conservatively optimal line-cut paths over $\mathcal{M}$.

In this work, we focused our discussion on the problem of the accessibility of infinite tangent lines over $\mathcal{M}$. However, there are quite a few directions this proposed set of algorithms can extend to:

- While we ensured global accessibility, we allowed for minute errors in the 1-ring neighborhood of the examined hyperbolic patch and the patch itself. This allowed local

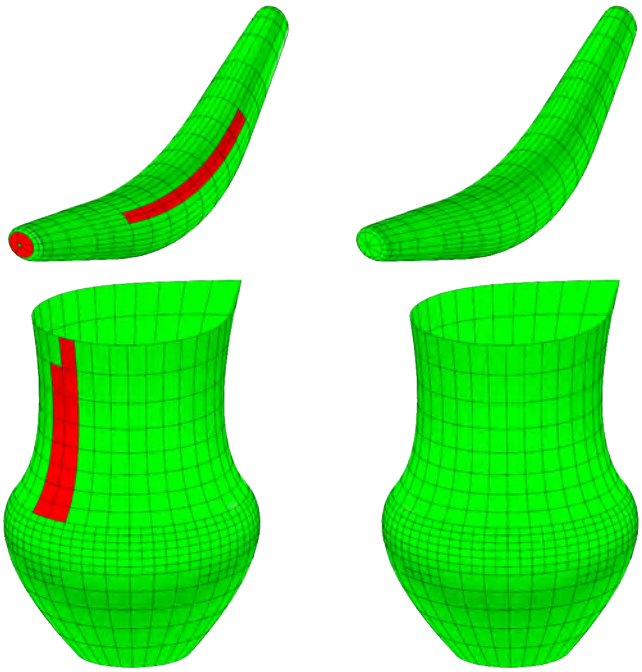


Figure 10: Given the classified patches of these Banana and Vase models into elliptic convex, hyperbolic and mixed patches, our assumption that tangent lines do not penetrate their own patches or the patches immediate neighbors renders all patches accessible (green on the right). Without this assumption, some patches that are accessible are detected as inaccessible (red on the left).

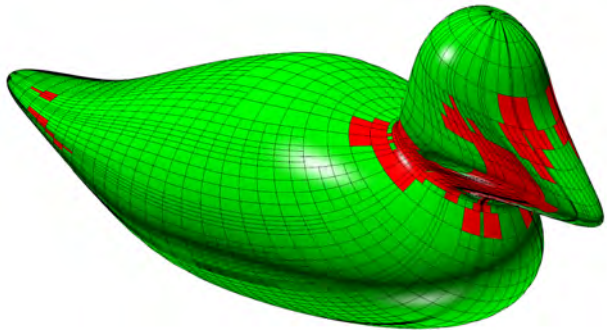


Figure 11: The accessible patches in the duck model. Note that our algorithm fails to detect accessible directions to some of the patches. These are typically mixed patches, or close to mixed patches (recall also Figure 8).

minute gouging clearly deserves some further research.

- Instead of using an infinite line-shaped tool (e.g. a heated wire) for fabrication, it is possible to use a rod-shaped tool, which extends from the cutting wire machine into the stock and ends at the contact point with the model. To modify our algorithm to solve the accessibility problem for such a ray-tool, the motion planning component simply need to seek a single accessible frustum, rather than a pair of opposite accessible frustums.

- Some wire-cutting configurations use a short wire, rather than a long (i.e. infinite) one which always extends outside the boundaries of the model. To solve the accessibility problem for this configuration, it is possible to limit the depth of the frustums to the length of the wire, while accessibility of the arc-frame that holds the finite wire needs to be verified as well.

- The presented algorithm can easily be adapted to detect obstructions that are caused by additional objects other than model $\mathcal{M}$ itself, similarly to [24]. This can be done by applying the same subdivision scheme to the additional objects, and considering the resulting surface patches as potentially obstructing patches. This could also be useful for handling parts of the manufacturing environment that are not the model, but can still interfere with the cutting process, such as the fixture that holds in place the work piece.

- Since the line-accessibility analysis of each surface patch S_i does not depend on the analysis results of any of the other patches, Algorithm 1 can be executed in parallel, virtually unmodified. This can greatly improve the performance of the algorithm and make it applicable for more complex models.

- Another, more complicated improvement to the presented algorithm is support of line-accessibility analysis for models with trimmed-surface. Most industrial-grade models consist of multiple trimmed surfaces, rather than tensor-products. This challenge can possible be faced by using an untrimmed algorithm [38] that converts trimmed geometry to tensor products.

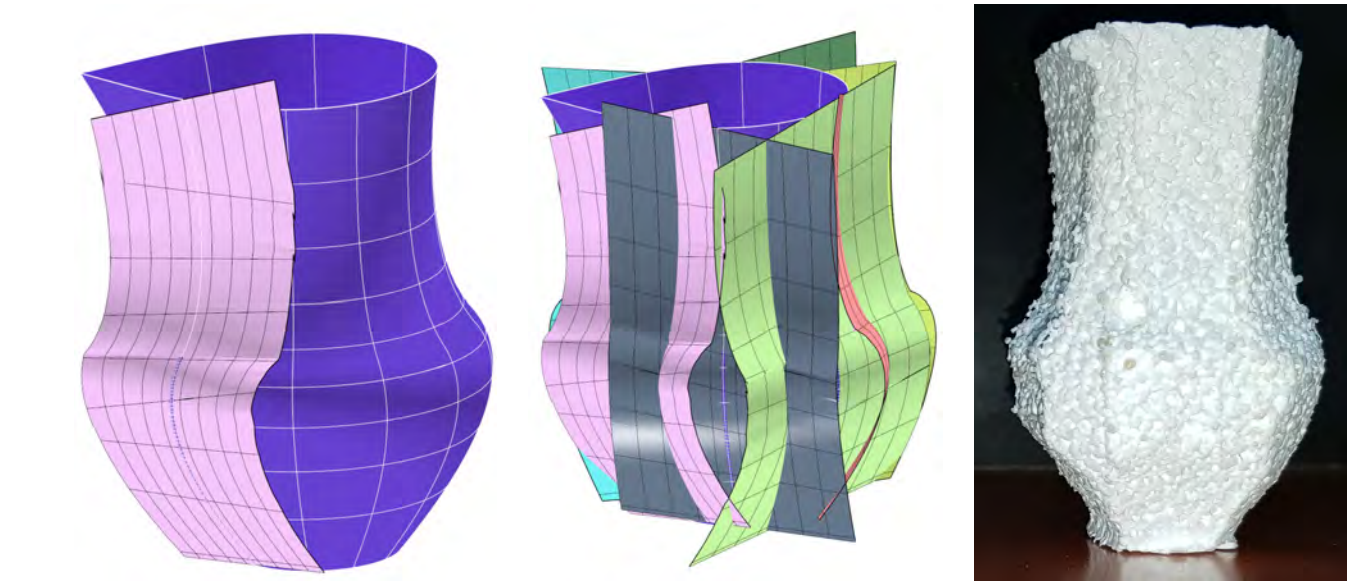
- Another emerging manufacturing technology is cutting with flexible (non-linear) blades [4, 5]. The flexible blade allows for cuts that better approximate the curvature and hence the shape of the model, while it can also access regions which are inaccessible for line-shaped cutting tools. The accessibility analysis for cutting with a flexible blade is a highly challenging problem, which has not yet been studied to the best of our knowledge. Yet, it has the potential of greatly improving the usability of this technology.

- Yet another important direction for future research is combining algorithms for optimizing the approximation of surfaces by line-cutting with the presented ability to analyze the geometry for line-accessibility. For example, methods for approximating models with ruled surfaces, such as [7, 15, 16], can be used to generate cut-paths with better coverage and approximation accuracy of the input model. Then, a cutting scheme which takes into account both the ruled surface approximation and the presented line-accessibility analysis will be able to generate high-quality, collision-free cutting paths.

Acknowledgments

This research was supported in part by the ISRAEL SCIENCE FOUNDATION (grant No. 597/18), in part the Defense Advanced Research Projects Agency (DARPA), under contract HR0011-17-2-0028, and in part by the Spanish State Research Agency (Spanish Ministry of Science, Innovation and Universities): BCAM Severo Ochoa excellence accreditation SEV-2017-0718 and by Ramón y Cajal with reference RYC-2017-22649. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

The authors would also like to thank Tom Shaked from the Faculty of Architecture and Town Planning, Technion, IIT, for his help in programming the robot shown in Figure 1, in our initial experiments in line cutting Styrofoam (Figure 12 C).



(a) (b) (c)

Figure 12: In (a), a vase with one ruling cut-path is presented. in (b), eight such ruling cut-paths are presented around the vase. Note the changes in the ruling direction as the paths are traversed. In (c), the actual cut part is presented, from Styrofoam. See also the video in `https://youtu.be/THwKUoJmEA8`.

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) 641–657.

[2] 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.

[3] R. F. Harik, H. Gong, A. Bernard, 5-axis flank milling: A state-of-the-art review, Computer-Aided Design 45 (3) (2013) 796–808.

[4] D. Brander, J. A. Bærentzen, K. Clausen, A.-S. Fisker, J. Gravesen, M. N. Lund, T. B. Nørbjerg, K. H. Steenstrup, A. Søndergaard, Designing for hot-blade cutting: geometric approaches for high-speed manufacturing of doubly-curved architectural surfaces, in: Advances in Architectural Geometry (AAG 2016), vdf Hochschulverlag AG an der ETH Zürich, 2016, pp. 306–327.

[5] D. Brander, J. A. Bærentzen, A.-S. Fisker, J. Gravesen, Designing interactively with elastic splines, Computer Aided Geometric Design 62 (2018) 181–191.

[6] M. Bartoň, H. Pottmann, J. Wallner, Detection and reconstruction of freeform sweeps, Computer Graphics Forum 33 (2) (2014) 23–32.

[7] S. Flöry, H. Pottmann, Ruled surfaces for rationalization and design in architecture, in: LIFE in:formation. On Responsive Information and Variations in Architecture, 2010, pp. 103–109.

[8] C. Li, S. Bedi, S. Mann, Flank milling of a ruled surface with conical tools – an optimization approach, Int. J. Adv. Manuf. Technol. 29 (2006) 1115i–1124.

[9] G. Elber, R. Fish, 5-axis freeform surface milling using piecewise rule surface approximation, ASME Journal of Manufacturing Science and Engineering 119 (3) (1997) 383–387.

[10] K. Sprott, B. Ravani, Cylindrical milling of ruled surfaces, Int. J. Adv. Manuf. Technol. 38 (2008) 649–656.

[11] H. Gong, C. Li-Xin, L. Jian, Improved positioning of cylindrical cutter for flank milling ruled surfaces, Computer-Aided Design 37 (2005) 1205–1213.

[12] J. Senatore, Y. Landon, W. Rubio, Analytical estimation of error in flank milling of ruled surfaces, Computer-Aided Design 40 (2008) 595–603.

[13] C. Chu, W. Huang, Y. Hsu, Machining accuracy improvement in five-axis flank milling of ruled surfaces, International Journal of Machine Tools and Manufacture 48 (7) (2008) 914–921.

[14] C. Wang, G. Elber, Multi-dimensional dynamic programming in ruled surface fitting, Computer-Aided Design 51 (2014) 39–49.

[15] S. Flöry, H. Pottmann, Ruled surfaces for rationalization and design in architecture, LIFE in: formation. On responsive information and variations in architecture (2010) 103–109.

[16] S. Flöry, Y. Nagai, F. Isvoranu, H. Pottmann, J. Wallner, Ruled Free Forms, Springer, 2013.

[17] J. Senatore, F. Monies, Y. Landon, W. Rubio, Optimising positioning of the axis of a milling cutter on an offset surface by geometric error minimisation, Int. J. Adv. Manuf. Technol. 37 (2008) 861–871.

[18] S. Bedi, S. Mann, C. Menzel, Flank milling with flat end milling cutters, CAD 35 (3) (2003) 293–300.

[19] L. Zhu, G. Zheng, H. Ding, Y. Xiong, Global optimization of tool path for five-axis flank milling with a conical cutter, Computer-Aided Design 42 (10) (2010) 903–910.

[20] A. Calleja, P. Bo, H. González, M. Bartoň, L. N. López de Lacalle, Highly accurate 5-axis flank cnc machining with conical tools, The International Journal of Advanced Manufacturing Technology (2018) 1–11.

[21] 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.

[22] H. Hua, T. Jia, Wire cut of double-sided minimal surfaces, The Visual Computer 34 (6-8) (2018) 985–995.

[23] A. Laurentini, The visual hull concept for silhouette-based image understanding, IEEE Transactions on pattern analysis and machine intelligence 16 (2) (1994) 150–162.

[24] A. Segall, J. Mizrahi, Y. J. Kim, G. Elber, Line accessibility of free form surfaces, Graphical Models 76 (5) (2014) 301–311.

[25] A. Laurentini, Computing the visual hull of solids of revolution, Pattern Recognition 32 (3) (1999) 377–388.

[26] A. Bottino, A. Laurentini, The visual hull of smooth curved objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (12) (2004) 1622–1632.

[27] A. Bottino, A. Laurentini, The visual hull of piecewise smooth objects, Computer vision and image understanding 110 (1) (2008) 7–18.

[28] N. Wang, K. Tang, Five-axis tool path generation for a flat-end tool based on iso-conic partitioning, Computer-Aided Design 40 (12) (2008) 1067–1079.

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

[30] N. M. Patrikalakis, T. Maekawa, Shape interrogation for computer aided design and manufacturing, Springer Science & Business Media, 2009.

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

[32] G. Elber, Free form surface analysis using a hybrid of symbolic and numeric computation, Ph.D. thesis, CS, The University of Utah (1992).

[33] R. E. Moore, Methods and applications of interval analysis, Vol. 2, Siam, 1979.

[34] M. P. Do Carmo, Differential Geometry of Curves and Surfaces: Revised

and Updated Second Edition, Courier Dover Publications, 2016.
[35] G. Elber, E. Cohen, Second-order surface analysis using hybrid symbolic and numeric operators, ACM Transactions on Graphics (TOG) 12 (2) (1993) 160–178.
[36] K. Shoemake, Animating rotation with quaternion curves, in: ACM SIGGRAPH computer graphics, Vol. 19, ACM, 1985, pp. 245–254.
[37] G. Elber, E. Cohen, Adaptive isocurve-based rendering for freeform surfaces, ACM Transactions on Graphics (TOG) 15 (3) (1996) 249–263.
[38] F. Massarwi, B. van Sosin, G. Elber, Untrimming: Precise conversion of trimmed-surfaces to tensor-product surfaces, Computers & Graphics 70 (2018) 80–91.

Appendix A. Bounds on penetration error in local neighborhhod

According to Assumption 1, our algorithm does not test adjacent patches for obstruction. Therefore, cases of tangent lines penetrating the interior of the model $\mathcal{M}$ may go undetected if they are limited to the patch S_0 and its 1-ring neighborhood, denoted together as patch S. In this appendix, we compute a bound on the error which may occur from ignoring obstructions within a surface patch S. Obstructions extending beyond S will be detected by our algorithm, causing the entire frustum of cutting directions to be inaccessible. Hence, we only consider cases for which both the entry and exit points are within S. The error is defined in terms of the 'depth' of the obstructed tangent line below the surface patch S.

Our subdivision process guarantees either one of the following:

1. The size of S (in terms of the diagonal of the bounding box) is bounded from above by $\tilde{d}_{max}$ (computed as $\tilde{d}_{max} = \sqrt{3}d_{max}$, where d_{max} is the longest edge of the bounding box of S), and the angular span of the normal directions of S is bounded from above by θ_{max}, or:
2. The size of S is bounded from above by $\tilde{d}_{min} = \sqrt{3}d_{min}$ (where d_{min} is the longest edge of the bounding box of S), regardless of the angular span of the surface, for cases of extremely curved geometry.

In the first case, we bound the error as follows:

Lemma 3. *Let S be a patch created from S_0 and its 1-ring neighborhood. Let $\tilde{d}_{max}$ be the* upper *bound on the size of S, and θ_{max} be the upper bound on the angular span of the normal directions of S. Then, the maximum depth of the obstruction below S is $\tilde{d}_{max} sin(2\theta_{max})$.*

Proof: *Let A, B be the points where the tangent enters and exits S. Position the complimentary tangent cone, $\overline{C_{\mathbf{t}}(\mathbf{n}, \frac{\pi}{2} \pm \theta)}$, of S, at A and B. Since S is fully contained inside each of the two complimentary tangent cones, B is inside the tangent cone constructed at A, and A is inside the tangent cone constructed at B (see Figure A.13).*
Let P be the plane which contains the line AB and the vector $\mathbf{n}$, and Let C be the intersection point of the two tangent cones and the plane P. Further, let a, b, c be the angles of triangle ABC, at the vertices A, B, C, respectively. Then, the depth of line segment AB under S is bounded by the altitude of triangle ABC at C, denoted h_C. Using the Law of Sines, this altitude is in turn given by: $h_C = |A - B|\frac{sin(a)sin(b)}{sin(c)}$. The angle $c = \pi - 2\theta$, and the angles a, b are each smaller than 2θ. Further, the length of the line segment AB is less than $\tilde{d}_{max}$. Therefore: $h_C < \tilde{d}_{max}\frac{sin^2(2\theta)}{sin(\pi-2\theta)} = \tilde{d}_{max} sin(2\theta) < \tilde{d}_{max} sin(2\theta_{max})$. ■

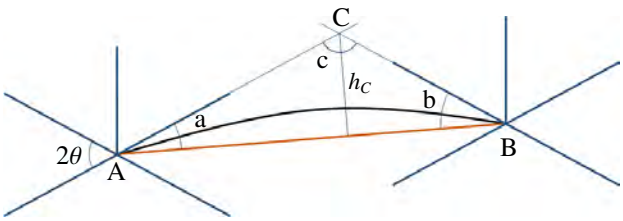


Figure A.13: A diagram of the error bound computation for patches with bounded normal cone angles, as described in Lemma 3.

The second case is fairly rare, and as stated already, occurs only in regions of very high curvature, and hence, no bound on θ exists. In this case, the error can be simply bounded by $\tilde{d}_{min}$.