Computation of Voxel Maps Containing Tool Access Directions for Machining Free-form Shapes

J.W.H. Tangelder,
J.S.M. Vergeest,
and M.H. Overmars

UU-CS-1996-23
June 1996

ISSN: 0924–3275

Proceedings of
The 1996 ASME Design Engineering Technical Conference and
Computers in Engineering Conference
August 18-22, 1996, Irvine, California

COMPUTATION OF VOXEL MAPS CONTAINING TOOL ACCESS DIRECTIONS FOR MACHINING FREE-FORM SHAPES

Johan W.H. Tangelder, Joris S.M. Vergeest
Faculty of Industrial Design Engineering
Delft University of Technology
Jaffalaan 9, NL-2628 BX Delft
The Netherlands
Phone +31 15 2781061, Fax +31 15 2787316
j.w.h.tangelder@io.tudelft.nl, j.s.m.vergeest@io.tudelft.nl

Mark H. Overmars*
Department of Computer Science
Utrecht University
P.O. Box 80089, NL-3508 TB Utrecht
The Netherlands
Phone +31 30 2533736, Fax +31 30 2513791
markov@cs.ruu.nl

ABSTRACT

An algorithm that derives tool access directions for machining free-form shapes is presented. A free-form shape to be machined is given by a preliminary B-spline model. We allow that the B-spline surface data are as inaccurate as the user-selected geometric accuracy of the prototype to be machined. Using surface sampling a visibility voxel map is obtained. From this map a voxel map is derived that contains per voxel a set of tool access directions. From the obtained voxel map regions can be selected that can be machined with a fixed tool access direction per region.

Keywords: Accessibility, voxel map, visibility, free-form shapes, manufacturing.

1 INTRODUCTION

Design and manufacturing can be integrated by either imposing constraints on the designed product geometry, or by extending the capabilities of the manufacturing process. We support preliminary design using the latter approach. Our goal is fully automatic NC machining of foam prototypes of large complex preliminary CAD models.

For this task we have installed a 7-axis Sculpturing Robot system consisting of an industrial Manutec R15 robot with 6 rotational degrees of freedom and a turntable that can rotate around its z-axis, see figure 1. The robot holds a milling tool and a stock of foam is mounted on the turntable. The machining process shrinks the initial foam stock to a foam prototype of a CAD model. We previously developed and implemented the SRPLAN1 robot motion planning and foam machining algorithm that uses only 5 predefined tool directions [Tangelder and Vergeest, 1994]. This, though, limits the range of possible objects to be machined. For example, figure 2 shows two models which cannot be machined with SRPLAN1. We are implementing a new motion planner, that derives a limited number of suitable tool access directions from the CAD model geometry, in this way extending the range of possible objects.

In fast, preliminary, design the product geometry may be only roughly specified. We consider the approach in which, with help of a B-spline modeler, a free-form shape M is defined as the volume enclosed by a B-spline surface shell, i.e., a set of B-spline surfaces. For this preliminary B-spline model we allow that the B-spline surface data are as inaccurate as the user-selected geometric accuracy of the prototype. The surfaces may be selfintersecting or intersect each other, and parts of the surfaces may be no part of the boundary of M. If a surface is part of the boundary, it need not be specified which side of the boundary is the outside of the object. Also surfaces that are intended to be adjacent, are allowed to be separated by narrow openings as long as these are smaller than the user-selected geometric accuracy.

A model checker [Tangelder, 1995] has been implemented, that checks the validity of the preliminary model M by a geometric procedure using a user-selected spatial accuracy of the prototype as a parameter as follows:

*This author was partially supported by the Netherlands Organization for Scientific Research (N.W.O.)

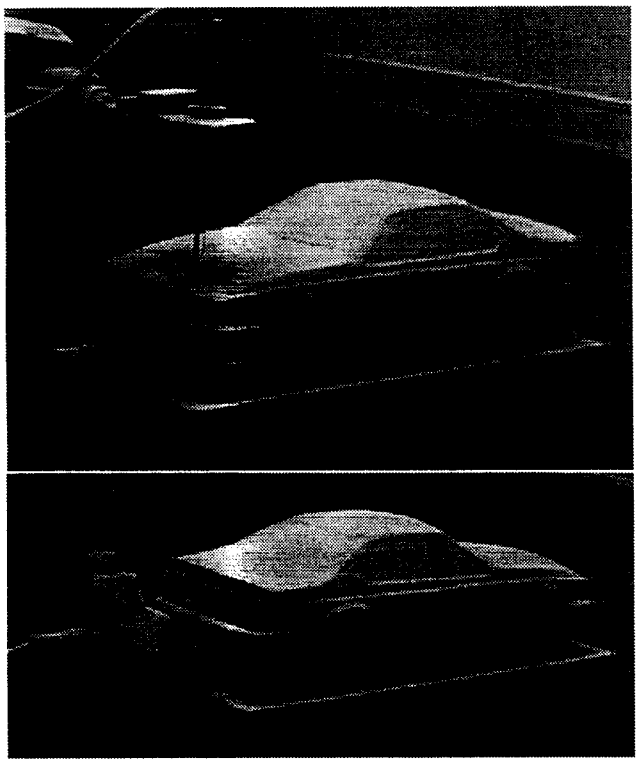

Figure 1. The Sculpturing Robot system performs rapid prototyping of CAD-defined objects.

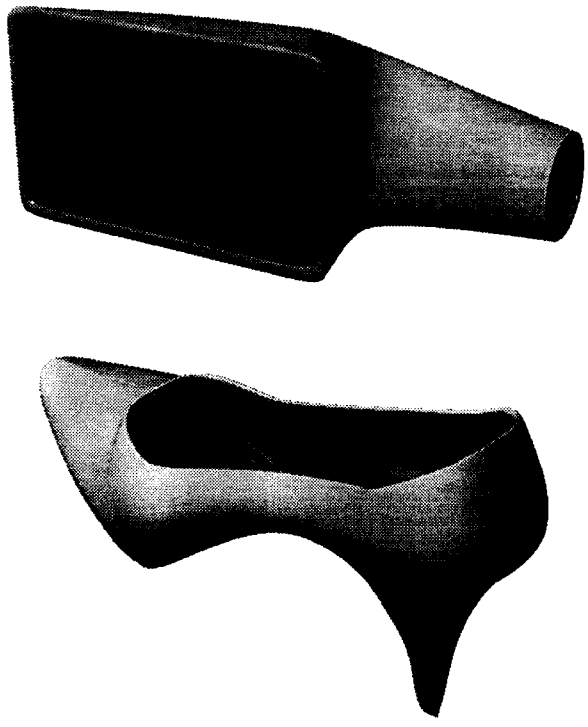

Figure 2. Examples of a car mirror (top), and a pump (bottom) that cannot be milled with the SRPLAN1 strategy, because additional tool access directions are required.

Surface sampling is used to obtain a voxel representation of the boundary ∂M of M. The voxel that are not boundary voxels are either exterior voxels or interior voxels. M encloses the interior voxels. The user-selected geometric accuracy of the prototype is taken as the voxel resolution d (i.e., the voxel edge length). We accept a model as valid if

- The voxel representation of ∂M is edge-connected, i.e. for each pair of voxels (v_0, v_m) in ∂M there is a sequence of voxels $(v_0, v_1, \ldots v_{m-1}, v_m)$ in ∂M, where v_i and v_{i+1} share at least an edge, $i = 0, \ldots, m-1$.
- For each B-spline surface the side of the surface that is (partially) the outside of the object can now be determined as follows. The normal vectors on a surface $S(u,v)$ are given by $\mathbf{n}(u,v) = SIGN(u,v)\frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$, where either $SIGN(u,v) = -1$ or $SIGN(u,v) = 1$, such that $\mathbf{n}(u,v)$ point outwards. One side of a surface defines the outside if for all (u,v) either $SIGN(u,v) = -1$ or $SIGN(u,v) = 1$. Otherwise both sides define a part of the outside of the object. In this case we do not accept the model as valid. We check this condition by applying Dijkstra's paint spreading algorithm [Dijkstra, 1959] to distinguish the interior and exterior voxels. Next we use surface sampling to compute $\frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$ at a number of sample points. If all these vectors point to an exterior voxel or a boundary voxel we set $SIGN(u,v) = 1$. If all these vectors point to an interior voxel or a boundary voxel we set $SIGN(u,v) = -1$. Otherwise we do not accept the model as valid.

Figure 3 shows a preliminary B-spline model and its voxel representation.

The ability of a tool to access a free-form shape M depends also on the shape of the tool. The axis of a flat-end tool cannot be tilted from the normal to ∂M at a given point. The axis of a ball-end tool can be tilted from the normal up to 90° degrees. Clearly, ball-end cutters offer the greatest range of movement. In this paper we ignore further limitations on accessibility that may be the consequence of collision avoidance between the tool holder and M or between the robot links and M.

In this paper we describe an algorithm that, given a free-form shape M, builds a voxel map that partitions the boundary ∂M of M. Given a ball-end milling tool, the algorithm computes for each voxel v in ∂M a set of global tool access directions for v. The obtained voxel map will be used to select regions that can be machined with a fixed

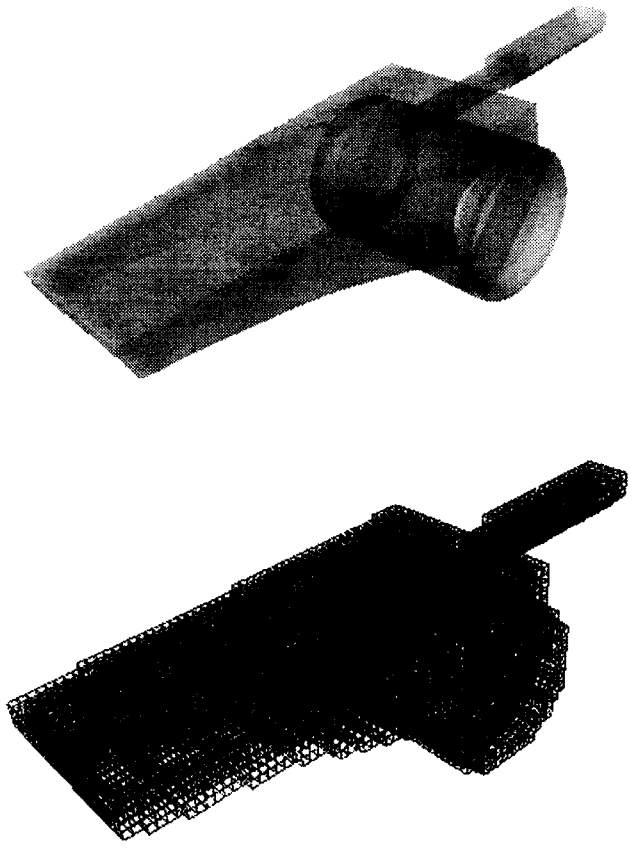

Figure 3. A preliminary B-spline model of a vacuum cleaner (top) and its voxel representation (bottom) with a low resolution. (Note that the surfaces in the B-spline model overlap each other)

tool access direction per region.

In section 2 an overview of research on determining tool access directions is given. In section 3 the shape M, that can be obtained by the milling process, is described as a minimal enclosing hull of M. In section 4 we present our algorithm. Conclusions are drawn in section 5.

2 RELATED RESEARCH

In [Gan, 1992] visibility analysis is applied to determine a range of set-up directions, that allows a given surface to be manufactured by a three-axis NC machine without having to redirect the stock-in-progress. Such a range of directions can be represented by the (global) visibility map, abbreviated to V-map, of the surface. Any point in a V-map represents a direction such that the entire surface is visible to its exterior at infinity. A two-stage approach is proposed for finding the V-map of a surface. An initial range of directions, termed the local V-map, is first obtained by considering only the local geometry of the surface. The (global) V-map is then obtained by removing from the local V-map every direction that causes interference with the tool. In [Spyridi and Requicha, 1990] local and global visibility cones and a two-stage approach as just described are used to determine accessibility of surface features for coordinate measuring machines. In [Woo, 1994] the concept of V-maps is applied to a wide range of manufacturing processes. An example of machining a model with 12 surfaces on a 4-axis milling machine is given. By analysing the 12 V-maps of the surfaces it is determined that only two set-ups are necessary. In this example these surfaces have no intersections except at their boundaries. Also each surface has a non-empty V-map. In [Chen et al., 1993, Gupta et al., 1995, Tang et al., 1992] geometric algorithms are presented that can be used to select a direction maximizing the number of surfaces that are visible with that direction. In [Elber and Cohen, 1995] a method to compute the local V-map of a surface exactly, is presented. In [Elber, 1994] an approach is presented that, given a convex surface S to be milled with 5-axis milling using a flat-end tool with direction perpendicular to S and other surfaces that limit the accessibility of S, reduces the milling accessibility problem to a 3-axis milling accessibility problem.

In all references above the V-maps are stored per surface. Hence, accessibility for a given tool access direction can be determined only per surface or for a surface set and not for regions, whose boundaries in general do not coincide with surface boundaries. For our new motion planner we need a data structure from which regions with a suitable tool access direction can be extracted. Hence, we represent the spatial coherence of M explicitely using a voxel map containing tool access directions instead of a V-map per surface.

3 THE RESULT OF THE MILLING PROCESS

A ball-end milling tool T can be modelled as a cylinder holding a spherical eraser, both with radius α. In this section we assume that if the spherical eraser is positioned such that it does not intersect M, also a global tool access direction exists such that the cylinder holding the spherical eraser does not intersect M. In the next section we will deal with finding global tool access directions. Due to the shape of T, in general it will be impossible to create the model M exactly. So the goal of the milling process is to obtain a minimal enclosing hull of M. If we assume that the spherical eraser can be placed anywhere outside the CAD model volume without intersecting M, this hull is the α-hull of M described in [Edelsbrunner and Mücke, 1994] as follows: Assume that the spherical eraser is omnipresent in the sense that it carves out foam at all positions where it does not in-

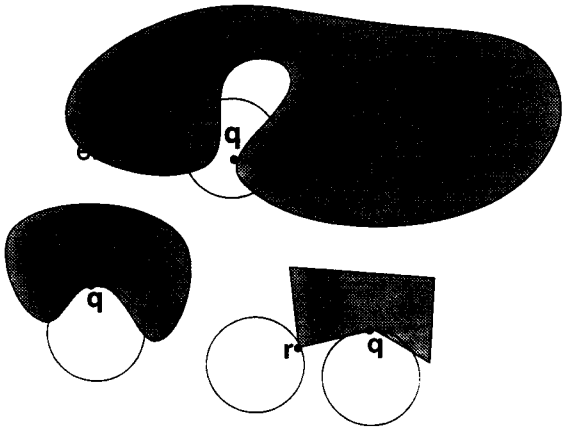

Figure 4. An example of a violation of accessibility condition 1. (top) and two examples (bottom) of a violation condition 2. In the right bottom example the curvature at q is ∞, at r it is $-\infty$.

tersect M. The resulting object is called the α-hull $\mathcal{H}_\alpha(M)$ of M. $\mathcal{H}_\alpha(M)$ is the minimal volume enclosing M that can be obtained with a free-flying eraser with radius α.

$M = \mathcal{H}_\alpha(M)$ if the spherical can touch every point on ∂M without intersecting M, i.e., if the following two accessibility conditions hold.

1. The spherical eraser intersects the interior of M, while the intersection of ∂M and the spherical eraser is homeomorphic to an open disk in $\mathbb{R}^2$.
2. The curvature of ∂M is at most $1/\alpha$.

Figure 4 illustrates cases for which $M \neq \mathcal{H}_\alpha(M)$.

M can be machined fully only if $M = \mathcal{H}_\alpha(M)$ and ∂M is accessible by the milling tool, i.e., the milling tool can touch every point on ∂M. Otherwise the final result of the milling process contains $\mathcal{H}_\alpha(M)$. We investigate accessibility in the next section.

4 DETERMINING ACCESSIBILITY

We consider the following problem. Given a complex three-dimensional shape $M \subset \mathbb{R}^3$ to be machined, provide a set of tool access directions such that M is fully accessible by a milling tool from these directions (i.e., every point $\mathbf{p}$ on the boundary ∂M of M can be touched by the tool end with one of these tool access directions). With these tool access directions the tool paths can be generated in a number of stages with a fixed tool access direction per stage using a depth buffer milling strategy (see [Saito and Takahashi, 1991, Tangelder and Vergeest, 1995]).

We present an algorithm to determine tool access directions for the milling process, using visibility analysis of M. The algorithm uses a novel spatial 3D visibility data structure, that integrates a visibility map of a volume with a low resolution voxel representation.

4.1 Definitions

Let $M \subset \mathbb{R}^3$ be a three-dimensional shape defined by a surface shell $\mathcal{M}$ as described in section 1. For our presentation it is convenient to assume that M emits rays. Further, we consider in this paper only normal vectors on ∂M that point outwards M.

Figure 5 illustrates the visibility definitions given below. A global visibility map will be defined for a single surface point $\mathbf{p}$ on a surface S, for the set of all surface points on a surface S and for all surface points contained in a voxel v. A voxel v that contains surface points is called a boundary voxel. But a boundary voxel may contain no points from ∂M, because parts of a surface may be no part of ∂M.

For a surface point $\mathbf{p}$, let the global visibility map $V_g(\mathbf{p})$ represent the directions of rays that start at $\mathbf{p}$ and do not intersect M.

For a surface S, let the global visibility map $V_g(S)$ represent the directions for which every ray starting at a surface point on S does not intersect M, i.e.,

$$V_g(S) = \bigcap_{\mathbf{p} \in S} V_g(\mathbf{p}).$$

For a boundary voxel v, $V_g(v)$ represents the directions for which every ray starting at a surface point contained in v does not intersect M, i.e.,

$$V_g(v) = \bigcap_{S \in \mathcal{M}, \mathbf{p} \in v \cap S} V_g(\mathbf{p}).$$

For the definitions of local visibility maps we consider only the local geometry of a surface rather than the whole model M. The normal $\mathbf{n_p}$ at a point $\mathbf{p}$ on a surface gives one direction in which the point is visible from infinity if the ray has no intersection. However, the same point may be visible from many other directions, up to a hemisphere $H(\mathbf{n_p})$ of directions bounded by the tangent plane at $\mathbf{p}$. Therefore we define the local visibility map $V_l(\mathbf{p}, S)$ of a point $\mathbf{p}$ on a surface S with normal $\mathbf{n_p}$ by $V_l(\mathbf{p}, S) = H(\mathbf{n_p})$. Note that for an intersection point of multiple surfaces different local visibility maps may exist.

Analogous to the global visibility maps we define the local visibility maps of surfaces and boundary voxels as

$$V_l(S) = \bigcap_{\mathbf{p} \in S} V_l(\mathbf{p}, S)$$

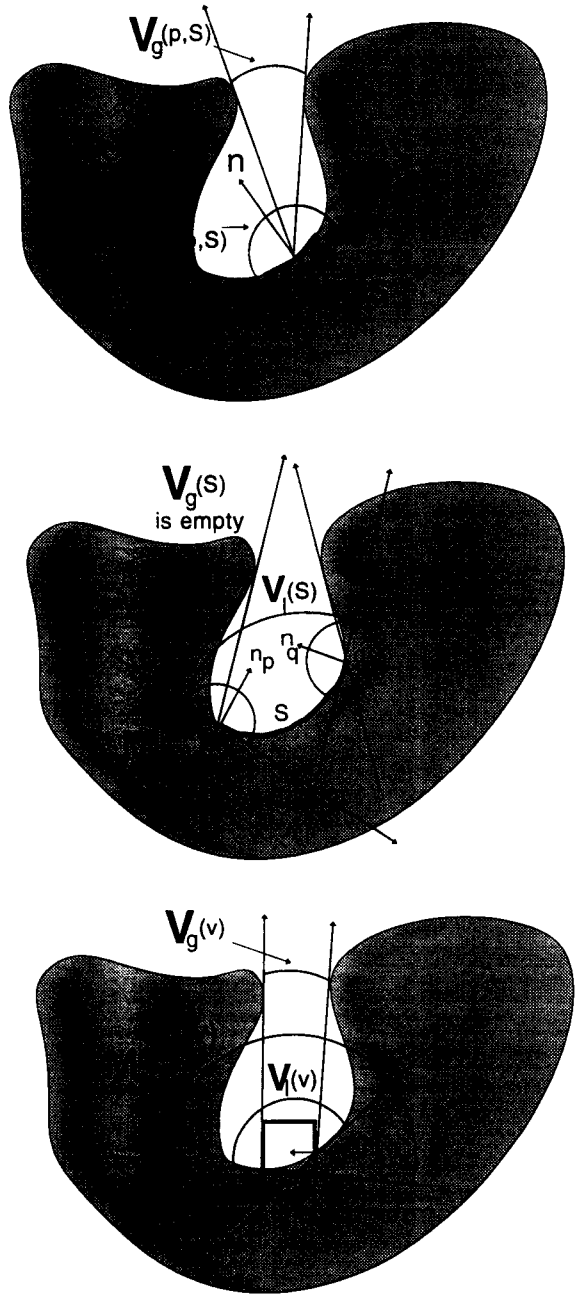

Figure 5. 2D examples of global and local V-maps for a point p (top), a surface S (middle) and a voxel v (bottom). $V_l(S)$ and $V_l(v)$ consist of the intersection of all hemispheres $H(n_\mathbf{p})$, for which p denotes a point on a surface S or a surface point in a voxel v, respectively. The bottom example shows also a light map $L(v)$ of the voxel v. $L(v)$ consist of the union of all hemispheres $H(n_\mathbf{p})$ for which p denotes a surface point in a voxel v.

and

$$V_l(v) = \bigcap_{S \in \mathcal{M}, \mathbf{p} \in v \cap S} V_l(\mathbf{p}, S).$$

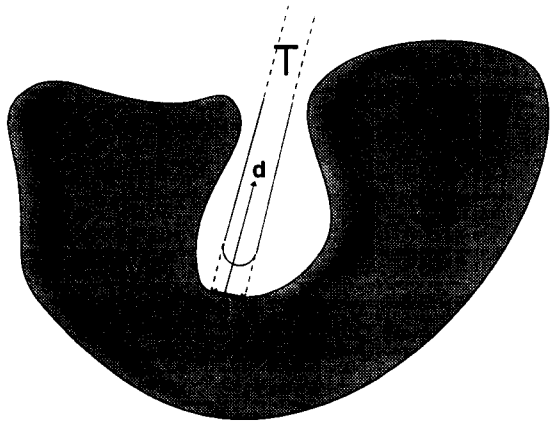

Figure 6. M is accessible at q by T with direction d (2D example).

For a boundary voxel v let the light map $L(v)$ represent all directions of rays starting at any surface point contained in v, i.e.

$$L(v) = \bigcup_{S \in \mathcal{M}, \mathbf{p} \in v \cap S} V_l(\mathbf{p}, S).$$

We define accessibility of a point on ∂M as follows:

Definition. *A model M is accessible at a point $\mathbf{q}$ on ∂M by a ball-end milling tool T with direction $\mathbf{o}$ if and only if the spherical eraser can touch $\mathbf{q}$, i.e., $\mathbf{q} \in \partial \mathcal{H}_\alpha(M)$, with direction $\mathbf{o}$, i.e., an environment of $\mathbf{q}$ is visible by a cylinder of rays with radius α.*

Figure 6 illustrates this definition. It can be generalized to voxels as follows:

Definition. *A voxel v of model M is accessible by a ball-end milling tool T with direction $\mathbf{d}$ if $\partial M \cap v \neq \emptyset$ and M is accessible at every point on $\partial M \cap v$ by T with direction $\mathbf{d}$.*

4.2 Method to determine accessibility

In this section we present a method to build a global voxel accessibility map. For a detailed description of this method we refer to [Tangelder, 1996].

In [Gan, 1992], for a surface S, $V_g(S)$ is obtained from $V_l(S)$ by deleting interfering directions as follows: All surfaces of M are sampled and for each pair of sample points its interfering direction is computed. All interfering directions are generated by taking the direction vectors between each pair of offsets $\mathbf{q} + \alpha \mathbf{n_q}$ and $\mathbf{p} + \alpha \mathbf{n_p}$ of sample points $\mathbf{q}$ and $\mathbf{p}$. $\mathbf{q}$ and $\mathbf{p}$ may be sample points on the same surface (see figure 7, top) as well sample points on different surfaces (see figure 7, bottom).

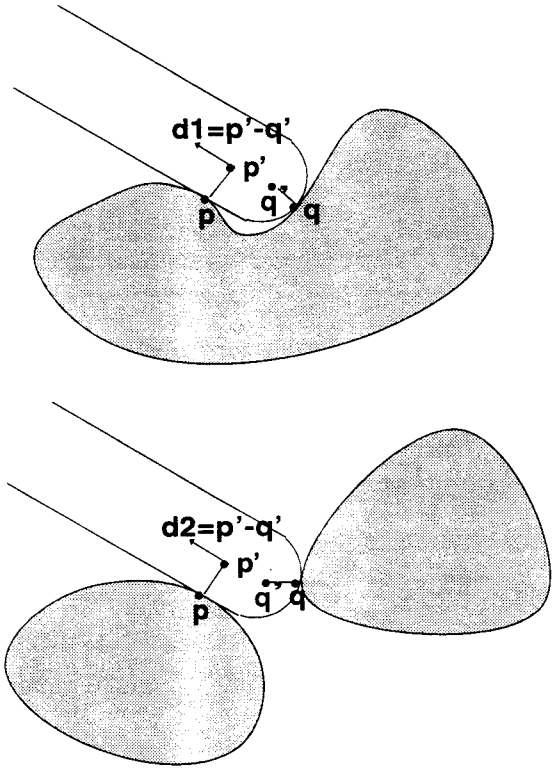

Figure 7. Intrasurface interfering direction d1 (top) and intersurface interfering direction d2 (bottom) (2D example).

This method can be optimized if the spatial coherence of M is represented in an explicit way with a spatial data structure, e.g., with a voxel map. Also face octrees [Brunet et al., 1993] can represent coherence of B-spline surfaces in an explicit way. Therefore we propose to represent M by a voxel map and to approximate the visibility of each voxel, instead of each surface, with a V-map. In order to save running time and storage, we consider only a finite number of directions instead of the infinite number of all possible directions. Since accessibility depends also on the curvature of the surfaces and the number of surfaces contained in a voxel, we include also the maximal curvature and the number of surfaces per voxel in our visibility voxel map.

We implement this as follows:

1. Apply surface sampling with a sample distance d' to build a visibility voxel map M' with a resolution $d \gg dd$, and approximate at each voxel v, the local V-map $V_l(v)$ by $V_l'(v)$ and the light map $L(v)$ by $L'(v)$ (as described below). Also record at each voxel v the maximal curvature and maintain an index to the surface for which v contains sample points. Set this index to null if v contains sample points from more than one surface.
2. Determine the interior, boundary and exterior voxels.
3. Compute at each voxel v, the approximated global visibility map $V_g'(v)$ from $V_l'(v)$ and from the $L'(w)$ maps of the boundary voxels $w \neq v$.
4. Derive from these global visibility maps a voxel map containing for each voxel global tool access directions.

4.3 Apply surface sampling to obtain the local V-maps

In [Gan, 1992] it is proposed to derive the local V-map from the Gaussian map (G-map), which is a map of normals on the unit sphere U. The local V-map can be constructed by the intersection of hemispheres, each having as its pole a point on the G-map.

This V-map can be computed as a subset of U in time $O(n \log n)$ where n is the number of normals on the spherical convex hull of the G-map [Gan, 1992, Woo, 1994]. First a surface is sampled and its G-map is built, next its V-map is built from the G-map, finally its G-map can be deleted.

Because we build a V-map per voxel, also storing a G-map per voxel would be necessary. To avoid storing the G-maps we integrate surface sampling with building the V-maps. This is possible if we approximate each V-map by a subset of a set N containing a finite number of preselected normals on U with the scheme presented below. Let (V, T) denote a tessellation of U, where V is a set of vertices and T is a set of tessel. Each tessel is a triangle $t = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ with $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in U$. A tessel $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ corresponds with a triangular surface patch on U between the vertices $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. Let $N = \{\mathbf{n} \in U \mid \exists t \in T : \mathbf{n}$ is contained in t's patch and $\mathbf{n} \perp t\}$. Hence, N is a finite set of normal vectors on U, that contains per tessel $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ exactly one vector. This normal vector points outwards the unit sphere and is perpendicular to the triangle $\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$.

When a sphere is tessellated, ideally we would like the tessels to be symmetrical, be identical in shape, and possess equal areas. Such a tessellation can only be obtained using one of the five Platonic polyhedra [Pugh, 1976]. The icosahedron (see figure 8, left) is the Platonic polyhedron with the maximal possible number of tessels. It contains 20 triangular tessels. In order to obtain smaller cell resolutions with triangular tessels that are approximately equal in area and shape, q-frequency polyhedra can be generated using the "alternate method" (see e.g. [Pugh, 1976]). The "alternate method" divides each triangular tessel of the icosahedron in q^2 triangles by lines running parallel to the original edges of the triangles. Hence, a set N of normal vectors is obtained, that contains $20q^2$ normals. This method has been applied by [Chen and Kak, 1989]. Figure 8 shows a one-, two- and four-frequency icosahedron tessellation of the unit sphere.

Given a normal vector $\mathbf{n} \in U$, the tessel whose patch contains $\mathbf{n}$ can be determined in $O(1)$ running time, as described in [Tangelder, 1996].

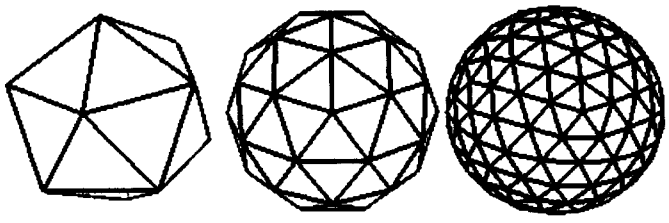

Figure 8. Three tessellations of the unit sphere obtained with an icosahedron (left), a two-frequency icosahedron (middle) and a four-frequency icosahedron (right).

We approximate local V-maps and light maps of voxels v and hemispheres of tessels t by subsets $V_l'(v)$, $L'(v)$ and $H(t)$ of N. For each vertex $\mathbf{v}$ we compute the hemisphere with pole $\mathbf{v}$ as a subset of N by $H(\mathbf{v}) = \{\mathbf{n} \in N \mid \mathbf{n} \bullet \mathbf{v} \geq 0\}$, where "$\bullet$" denotes the dot product.

For each tessel $t = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ we approximate its visibility hemisphere by $H_V(t)$ defined as

$$H_V(t) = H(\mathbf{v}_1) \cap H(\mathbf{v}_2) \cap H(\mathbf{v}_3)$$

and its light hemisphere by $H_L(t)$ defined as

$$H_L(t) = H(\mathbf{v}_1) \cup H(\mathbf{v}_2) \cup H(\mathbf{v}_3).$$

Note that the sets $H_V(t)$ and $H_L(t)$ can be precomputed and with this scheme the V_l' and L' maps can be computed as follows:

- Initialize for each voxel v, $V_l'(v)$ with U and $L'(v)$ with $\emptyset$.
- Perform surface sampling with sample distance $d' \ll d$. For each sample point determine the voxel v containing it. Determine the normal $\mathbf{n}$ at the sample point as described at the end of this subsection and the tessel t whose patch contains $\mathbf{n}$. Intersect $V_l'(v)$ with $H_V(t)$ and unify $L'(v)$ with $H_L(t)$. If v contains sample points from only one surface then record an identifier of this surface. If v contains sample points from multiple surfaces then set the identifier to null. Also record at each voxel v the maximal curvature, which is computed as described below.

Note that with this scheme subsets of N like visibility maps, light maps and hemispheres can be implemented by variables that require k bits storage, where $k = |N|$. For $N = \{\mathbf{n}_1, \mathbf{n}_2, \ldots \mathbf{n}_k\}$, the i^{th} bit of such a variable indicates whether the subset contains $\mathbf{n}_i$. Hence, intersection and unification can be implemented with the bitwise AND operation and the bitwise OR operation, respectively. Hence, this step has $O(kn)$ running time, where n is the number of sampled points. Note that the light maps represent also the boundary voxels implicitly, since a voxel v contains a sample point if and only if $L'(v) \neq \emptyset$.

Computation of normal vectors and curvature The surfaces are sampled per patch and the sample points are stored in a 2D array $\mathbf{p}(i,j)$. From these sample points we derive the partial u-, v-, uu-, uv- and vv-derivative at $\mathbf{p}(i,j)$. The normal vector is computed as the cross-product of the u- and v-derivative value at $\mathbf{p}(i,j)$. The partial u-, v-, uu-, uv- and vv-derivative values at $\mathbf{p}(i,j)$ are used to compute the maximal surface curvature at $\mathbf{p}(i,j)$ with the method described by [Boehm, 1993]. Since for some parameterizations partial derivatives computed analytically are vanishing, we prefer this geometrical approach above an analytical approach. We refer to [Tangelder, 1996] for a detailed description of these computations.

4.4 Determine the interior, boundary and exterior voxels

With the surface sampling algorithm the boundary voxels have been identified. We apply Dijkstra's paint spreading algorithm [Dijkstra, 1959] to distinguish the interior and exterior voxels as described in [Tangelder, 1995]. The algorithm has $O(m_0)$ computing time, where m_0 denotes the number of boundary voxels.

4.5 Obtain the global V-maps

The algorithm to obtain the V_g maps from the V_l maps is based upon the following observation.

Observation. *Let* $\mathbf{n} = (\mathbf{q}-\mathbf{p})/|\mathbf{q}-\mathbf{p}|$ *be a ray that leaves* M' *at a point* $\mathbf{p} \in v$ *and intersects* M' *again at a point* $\mathbf{q} \in w$. *Then* $\mathbf{n} \in V_l(v)$ *and* $-\mathbf{n} \in L(w)$.

Given k directions and a voxel resolution d the following straightforward algorithm approximates the global V-map of each voxel v by $V_g'(v)$. The running time is $O(km_1m_2)$, where m_1 is the number of boundary voxels of M' and m_2 is the mean number of number of voxels intersected by a cylinder of rays.

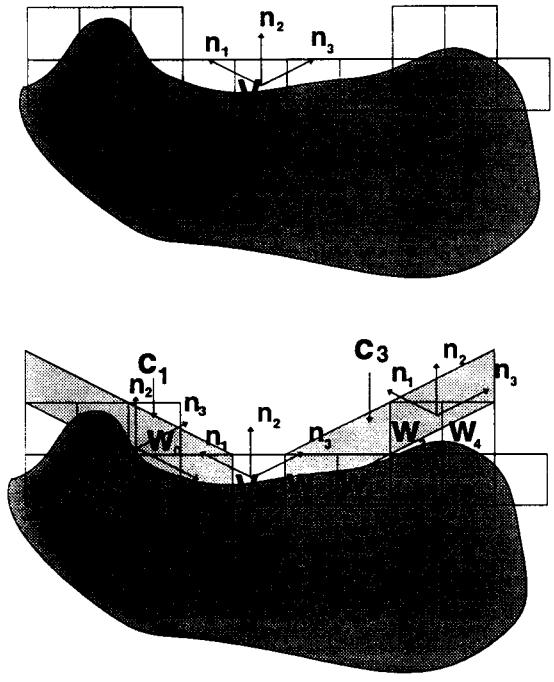

Figure 9. The local visibility map of v contains normal vectors n_1, n_2 and n_3 (top). The light map of w_0 contains $-n_1$. The cylinder c_1 with direction n_1 intersects w_0. Hence, n_1 is deleted from the global visibility map of v. The light maps of w_1, w_2, w_3 and w_4 do not contain n_3. Although the cylinder c_3 with direction n_3 intersects these voxels, n_3 is not deleted from the global visibility map of v. (bottom).

```
FOR EACH boundary voxel v
    V'_g(v):=V'_l(v);
    FOR EACH direction n ∈ V'_g(v)
        compute the cylinder c of rays starting at
        v with direction n;
        FOR EACH voxel w intersected by c
            IF w is an interior voxel OR
            w is a boundary voxel with −n ∈ L'(w)
            THEN
              V'_g(v) := V'_g(v)/{n};
            END IF;
        END FOR;
    END FOR;
END FOR;
```

Figure 9 illustrates this algorithm. For an efficient implementation of this algorithm, a list of voxels that are intersected by a cylinder of rays should be generated efficiently (see, e.g., [Tangelder, 1996]).

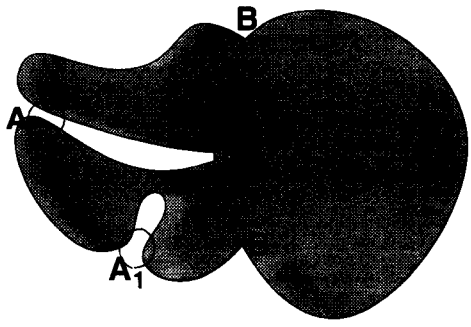

Figure 10. If accessibility condition 1. (location A1 and A2) or accessibility condition 2. (location B) of lemma 1 is violated, it is assumed that M is not accessible at these locations.

4.6 Obtain a voxel map containing tool access directions

Let the milling tool T be positioned with access direction $\mathbf{n}$. Let $\partial T_{\mathbf{n}}$ denote the boundary of the spherical eraser outside the tool cylinder. Hence, $\partial T_{\mathbf{n}}$ is a hemisphere with pole $-\mathbf{n}$. If $\partial T_{\mathbf{n}}$ intersects a voxel v, it may intersect also other voxels. Let $E_{\mathbf{n}}(v)$ be the set consisting of the voxel v and the other voxels that $\partial T_{\mathbf{n}}$ may intersect. If $d \geq 2\alpha$, where d denotes the voxel resolution and α the tool radius, $E_{\mathbf{n}}(v)$ consists of the voxels that share at least a vertex with v.

The following algorithm computes for each boundary voxel v an accessibility map $A'(v)$ containing suitable tool access directions.

Initialize for each boundary voxel v of M', $A'(v)$ with $\emptyset$. For each boundary voxel v of M' and for each direction $\mathbf{n} \in V'_g(v)$ check if v can be machined with direction $\mathbf{n}$, i.e., check the following three conditions:

1.
$$\mathbf{n} \in \bigcap_{w \in E_{\mathbf{n}}(v)} V'_g(w).$$

2. The maximal surface curvature does not exceed $1/\alpha$ in $E_{\mathbf{n}}(v)$.
3. $E_{\mathbf{n}}(v)$ contains only one surface.

Add $\mathbf{n}$ to $A'(v)$, if these conditions are met.

This algorithm has $O(km_1)$ running time, where m_1 is the number of boundary voxels. Figure 10 illustrates cases for which the algorithm decides that M is locally inaccessible, i.e., at least one boundary voxel with an empty accessibility map will be computed. Since the algorithm does not distinguish convex from concave surface intersections, it is always assumed that a voxel v is not accessible with direction $\mathbf{n}$, if $E_{\mathbf{n}}(v)$ contains more than one surface. Figure

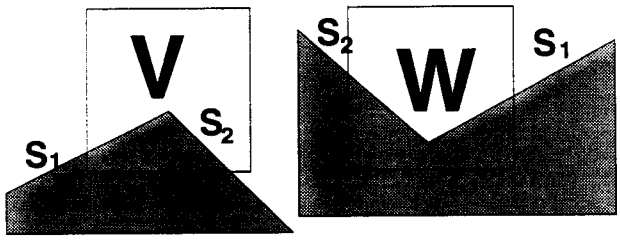

Figure 11. The voxel visibility map cannot distinguish convex from concave surface intersections.

11 illustrates a violation (concave surface intersection at a voxel v) of accessibility condition 2. and a case (convex surface intersection at a voxel v) that does not violate this condition, but for which the algorithm decides that M is locally inaccessible. The algorithm cannot determine the accessibility at v, because our 3D visibility data structure cannot distinguish concave from convex surface intersections.

5 CONCLUSIONS AND FURTHER RESEARCH

We have presented an algorithm that, given a complex three-dimensional model to be machined, builds a voxel map that partitions the boundary ∂M of M. Each voxel v contains a number of global tool access directions for v. With these directions, tool paths can be generated in a number of stages with a fixed tool direction per stage using a depth buffer milling strategy [Saito and Takahashi, 1991, Tangelder and Vergeest, 1995]. The algorithm also identifies voxels that contain parts of ∂M, that cannot be machined with any tool direction. The algorithm has $O(MAX(m_0, k\text{MAX}(m_1 m_2, n)))$ running time and requires $O(MAX(m_0, km_1)$ storage, where m_0 is the number of voxels, m_1 is the number of boundary voxels of M' and m_2 is the mean number of number of voxels intersected by a cylinder of rays, n the number of sample points and k the number of preselected normal vectors on the unit sphere. Currently, this algorithm is implemented. The algorithm will be included in a new milling process planner, that generates a number of milling stages with arbitrary tool access directions. The voxel map will be used to select regions that can be machined with the same tool access direction.

The face octrees described in [Brunet et al., 1993] may provide a good alternative to represent the spatial coherence of B-spline surfaces in an explicit way, because the size of the face nodes is based on the local surface curvature. With this data structure also concave and convex surface intersections can be distinguished.

Problems with the surface geometry may be avoided by converting the B-spline surface geometry to a voxel map with a relatively small resolution. For each voxel v from this voxel map a normal vector can be estimated from the geometry of v and the voxels in its neighbourhood (see [Yagel et al., 1992]). Because the number of voxels is much larger than with our scheme, computing a global V-map from the local V-map would require much more computing time.

ACKNOWLEDGMENT

The authors thank the Sculpturing Robot staff for their support.

REFERENCES

[Boehm, 1993] Boehm, W. (1993). Differential geometry II. In Farin, G., editor, *Curves and Surfaces for Computer Aided Geometric Design: a Practical Guide, 3rd. ed.*, chapter 22, pages 389–405. Academic Press, San Diego.

[Brunet et al., 1993] Brunet, P., Navazo, I., and Vinacua, A. (1993). A modelling scheme for the approximative representation of closed surfaces. *Computing Suppl.*, 8:75–90.

[Chen and Kak, 1989] Chen, C. and Kak, A. (1989). A robot vision system for recognizing 3-D objects in low-order polynomial time. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(1):1535–1563.

[Chen et al., 1993] Chen, L., S., C., and Woo, T. (1993). Separating and intersecting spherical polygons: computing machinability on three-, four-, and five-axis numerically controlled machines. *ACM Transactions on Graphics*, 12(4):305–326.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numer. Mathem.*, 1:269–271.

[Edelsbrunner and Mücke, 1994] Edelsbrunner, H. and Mücke, E. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72.

[Elber, 1994] Elber, G. (1994). Accessibility in 5-axis milling environment. *Computer Aided Design*, 26(11):796–802.

[Elber and Cohen, 1995] Elber, G. and Cohen, E. (1995). Arbitrarily precise computation of Gauss maps and visibility sets for freeform surfaces. In *Proc. Solid Modeling '95*, pages 271–279, Salt Lake City, Utah, USA.

[Gan, 1992] Gan, J. (1992). Set-up orientations of workpieces for machining by three-axis numerical control machines. *Journal of Design and Manufacturing*, 2:59–69.

[Gupta et al., 1995] Gupta, P., Janardan, R., Majhi, J., and Woo, T. (1995). Efficient geometric algorithms for workpiece orientation in 4- and 5-axis NC-machining. In *Proc. Fourth Workshop on Algorithms and Data Structures (WADS '95)*, pages 171–182, Kingston, Ontaria, Canada.

[Pugh, 1976] Pugh, A. (1976). *Polyhedra: A Visual Approach.* University of California Press, Berkeley, California.
[Saito and Takahashi. 1991] Saito, T. and Takahashi, T. (1991). NC machining with G-buffer method. *Computer Graphics*, 25(4):207–216. (Proc. SIGGRAPH '91).
[Spyridi and Requicha. 1990] Spyridi, A. and Requicha, A. (1990). Accessibility analysis for the automatic inspection of parts. In Volz, R., editor, *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1284–1289, Cincinnatti, Ohio.
[Tang et al., 1992] Tang, K., Woo, T., and Gan, J. (1992). Maximum intersection of spherical polygons and workpiece orientation for 4- and 5-axis machining. *Journal of Mechanical Design*, 114(3):477–485.
[Tangelder, 1995] Tangelder, J. (1995). Fully automatic milling of free-form CAD-defined objects using an industrial robot. Technical report, Delft University of Technology.
[Tangelder, 1996] Tangelder, J. (1996). Implementation notes for the SRPLAN2 direction selector module. Technical report, Delft University of Technology.
[Tangelder and Vergeest, 1994] Tangelder, J. and Vergeest, J. (1994). Robust NC path generation for rapid shape prototyping. *Journal of Design and Manufacturing*, 4:281–292.
[Tangelder and Vergeest, 1995] Tangelder, J. and Vergeest, J. (1995). Z-buffer machining with interference avoidance for tool and toolholder using Minkowski operations. In *Fourth SIAM Conference on Geometric Design.*
[Woo, 1994] Woo, T. (1994). Visibility maps and spherical algorithms. *Computer Aided Design*, 26(1):6–16.
[Yagel et al., 1992] Yagel, R., Cohen, D., and Kaufman, A. (1992). Normal estimation in 3D discrete space. *The Visual Computer*, 8:278–291.