

# 5 Axis Milling Simulation using a Swept Volume via Gauss Map

Seok Won Lee\*, Andreas Nestler  
Institute of Manufacturing Technology  
Dresden University of Technology, D-01062, Dresden, Germany  
{swlee,nestler}@mciron.mw.tu-dresden.de

## ABSTRACT

In this paper, a novel approach to 5 axis NC milling simulation using GPU is presented. Modelling swept volumes (SV) of a milling tool that undergoes a translational and rotational motion is based on the idea behind Gauss maps. The main process is that it 1) transforms the tool shape and the moving direction of the tool from Euclidean space into an unit sphere by means of Gauss function and gets the spherical domains which are called Tool map (T-Map) and Contact map (C-Map) respectively 2) finds an intersection region of T-Map with C-Map and 3) transforms intersected region inversely into Euclidean space to get the envelope profile of the tool at prescribed instant time. As of now, Graphics Processing Unit (GPU) in graphics hardware is being developed more drastically as compared with Central Processing Unit (CPU). To achieve our purpose of updating the work piece GPU is exploited as follows: 1) SV is rendered using graphics hardware (Model building) 2) color and depth information is read back from the frame buffer in GPU to CPU (sampling) 3) at last, the updated work piece is built based on extracted data (surface reconstruction).

**Keywords :** 5 Axis Milling, Swept Volume, Gauss Map, Simulation, GPU Programming.

## 1. INTRODUCTION

Numerical control (NC) milling is one of the most prevailing processes among manufacturing processes. To ensure product quality and to avoid junks owing to undercut or collision between the tool and the material it is necessary to simulate the machining process before real cutting. Because the time-to-market gets shorter in modern production cycles, the verification of milling process plays a more crucial role in whole manufacturing phases [Jac91].

As the milling is the material removal process with a geometry-defined cutting tool, it is essential to subtract the tool swept volume (SV) continually from the raw stock in the simulation environment in order to verify tool paths.

There is a bunch of attempts to get SV and to verify the NC milling based on SV of a moving cutter. These can be roughly classified into three categories by mathematical description and practical implementations: 1) Line-based (1D), 2) Surface-based (2D) and 3) Volume-based (3D) methods.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
Copyright UNION Agency – Science Press, Plzen, Czech Republic.

**Volume-based method** is based on constructive solid geometry (CSG) [Kaw91]. It provides preciseness of the geometric expression and robustness for the Boolean operation that simulates the cutting processes. But computing performance using the CSG approach is  $O(n^4)$  where  $O$  is the upper bound function and  $n$  is the number of NC code blocks [Jer89]. The milling simulation with thousands of tool movements is comprised of sweep volume generation and successive Boolean subtraction from the stock, which is so time-exhaustive that it is usually not amenable in practical use.

Secondly, **surface-based method** is based on boundary representation (Brep) and focuses on finding the boundary description of SV. The core issues are how to find the envelope profiles that belong to the boundary surface covered by the superimposed outlines of a moving cutter. Then this problem can be reduced to find the outmost points (or critical points) contributing to the boundary surface at a given time. At last, swept boundary is obtained as a polyhedral approximation by means of interpolating a set of envelope profiles. In this category, several principles are introduced such as envelope theory [Wan86], Sweep Envelope Differential Equation (SEDE) [Bla97], Imprint method [Rot01].

To overcome the time complexity during Boolean operation, **line-based methods** are introduced by many researchers since earlier. Z-map [Chu98], Dixel [Hui94], Lawnmower method [Jer89] are classified

into this category. The main idea behind the line-based method is to find the intersections between rays and tool body. That is, it reduces the complexity of the Boolean operation from three to one dimensional calculation. So the time complexity is drastically reduced from  $O(n^4)$  to  $O(n^1)$  and the algorithms are robust to implement. Its major disadvantages, however, are the inevitable rough approximation when to mill such large models as molds and dies and the graphical ambiguity when to get intersection points between rays and tool boundaries that are almost parallel to rays. To overcome these disadvantages there are also some studies [Par05].

Graphics hardware is an indispensable component on PC for the visualization and the numerical calculation. Recent graphics hardware provides huge memory bandwidth and extremely fast computation due to highly parallelised chip architecture. There are countless applications using graphics hardware not only in traditional rendering but also in a variety of fields such as physical based simulation, signal and image processing and database etc [Owe05]. In this paper, however, graphics hardware is handled as a digitizer to sample an object and to read the sampled information back. The read back process at graphics hardware is actually the opposite process rather than what is normally used or writing. To the best of knowledge of the author, there are no attempts to use graphics hardware for the purpose of sampling geometric models. For detail see section 4.

This paper presents a novel method to perform Boolean operation on triangulated mesh, to keep track of the in-process geometry and to reconstruct the final geometry as polygon model by exploiting the recent development in graphics acceleration hardware. Main contributions in this paper are

- SV generation of tools undergoing five axis motion via Gauss map,
- the sampling technique using modern graphics hardware,
- Boolean operation of tessellated models,
- surface reconstruction in object space from the rendered image
- updating in-process work piece which can be saved in the memory storage at any step of simulation.

This paper is organized as follows: In Section 2 it provides some concepts and definitions that are useful through the paper. Section 3 explains the approach to get the three axis swept boundary using Gauss map and then extends the idea to the five axis swept volume of a generalized tool shape. The method of the material removal simulation is presented in section 4. In Section 5 intermediate results under working are illustrated with concrete examples. Section 6 concludes and summarizes the paperwork.

## 2. MATHEMATICAL BACKGROUND

To make this article self-contained, mathematical definitions and concepts are described.

*Definition 1. Gauss map and Gaussian sphere*

Each point on the  $C^1$ -surface has its normal vector. If the starting points of the unit surface normal vectors are translated to the origin, an image is built on a unit sphere. This mapping function is called Gauss map. The Gauss map of a orientable surface  $M$  is a function from  $M$  in Euclidean space to a unit sphere. It associates every point on the surface its oriented normal vector [DoC83].

*The Gauss map (GM) maps a surface in Euclidean space  $\mathbf{R}^3$  to the unit sphere  $\mathbf{S}^2$ . Namely, given a surface  $S$  lying in  $\mathbf{R}^3$ , the GM is a continuous map  $G:S \rightarrow \mathbf{S}^2$  such that  $G(p)$  is orthogonal to  $S$  at a point  $p$  on  $S$ .*

The range, which is mapped from a domain in Euclidean space by GM, is called **Gaussian sphere (GS)**.

*Definition 2. Convex hull*

The **convex hull**  $C$  or **convex envelope** of an object or a set of objects is the minimal convex set containing the given objects. In  $2D$  ( $3D$ ), the convex hull is found conceptually by stretching a rubber band (balloon) around the points (the object) so that all of the points (faces of the object) lie within the band (balloon). For  $N$  points  $\mathbf{p}_1, \dots, \mathbf{p}_N$  in space, convex hull  $C$  is given by the expression

$$C = \left\{ \sum_{i=1}^N u_i \mathbf{p}_i \mid u_i \geq 0, \sum_{i=1}^N u_i = 1, i = 1, 2, \dots, N \right\}. \quad (1)$$

several properties of GM can be derived from the definition.

*Property 1: Many-to-one mapping*

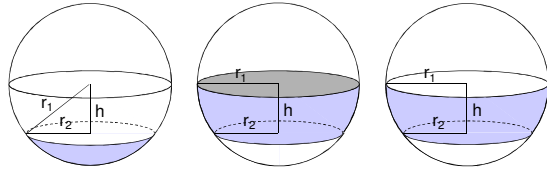
A point on a  $C^2$ -continuous convex hull is uniquely mapped to a point on the Gaussian sphere. But the piecewise  $C^1$ -continuous convex hull, e.g. points on a plane or on a line embedded in the plane, is repeatedly mapped to one point on Gaussian sphere that corresponds to the normal to the plane. Therefore, GM is a many-to-one relationship in general.

*Property 2: Existence of inverse function*

According to *property 1*, if a point on Gaussian sphere is chosen, its **inverse geometry** in the domain such as a plane, a line, a point on the convex hull is to be found using inverse GM. But the question which point on the geometry should be determined is yet open.

*Definition 3. A spherical cap, segment and zone*

A **spherical cap** is a portion of a thin sphere cut off by a plane. A **spherical segment** is the solid defined by cutting a sphere with a pair of parallel planes. It can be thought of as a spherical cap with the bottom truncated, and so it corresponds to a spherical frustum. The surface of the spherical segment (excluding the bases) is called a **zone** (see Fig.1) [Har98].



a) Spherical cap b) Spherical segment c) Zone  
Figure 1. Spherical geometry

### 3. SWEEP VOLUME GENERATION

[Lee06] presents a method for modeling SV of milling tools that undergoes a translational motion via GM. This section explains that basic idea and extends to the five axis motion following translational and rotational motion.

#### 3.1 The Proposed Approach

*Definition 4* : Contact map (C-Map)

Let  $\tau$  a arbitrary vector in the space and map  $\tau$  onto the unit sphere using Gauss function. Then a great circle is achieved by intersecting the unit sphere with a plane orthogonal to  $\tau$  passing the center of the sphere. This great circle is defined as Contact map (C-Map) of the **generating vector**  $\tau$  (Fig. 2). The C-Map  $CM$  of  $\tau$  is given as follows.

$$CM = \{n \mid n \cdot \tau = 0, |n| = 1, |\tau| \neq 0\} \quad (2)$$

*Lemma 1*. C-Map has such a characteristic that a position vector  $n$  on the sphere belongs to one of three different regions according to scalar vector product  $\tau \cdot n$ : 1) positive region ( $\tau \cdot n > 0$ ) 2) zero region ( $\tau \cdot n = 0$ ) 3) negative region ( $\tau \cdot n < 0$ ).

*Lemma 2*. Points on the hemisphere, that is the positive region in Gaussian sphere, are represented as a region of the tool participating in cutting the material in Euclidean space. Points on the hemisphere, that is the negative region, are represented as a region of the tool leaning against the material and points on the great circle that lies between the positive and negative region is represented as points on the blink of cutting and leaving the material in Euclidean space. Namely, GS is divided into 1) front-facing region ( $\tau \cdot n > 0$ ) 2) enveloping region ( $\tau \cdot n = 0$ ) 3) back-facing region ( $\tau \cdot n < 0$ ) (Fig. 2).

By *lemma 2* if a tool shape could be mapped on to Gaussian sphere, one can interpret the cutting behavior of the tool using C-Map. The cutting tool

can be geometrically represented as a Boolean sum and/or difference of primitives such as cylinder, sphere, cone and torus or quadratic solids like, if necessary, elliptic paraboloid and ellipsoid etc. But in this paper it is presumed that a cutting tool is a body of revolution which is convex hull.

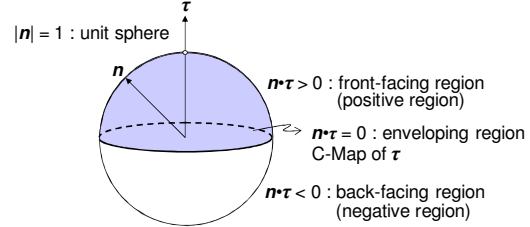


Figure 2. Contact Map (C-Map) of a vector  $\tau$

*Definition 5* : Tool map (T-Map)

By *Definition 1* the cutting tool can be mapped into points, circles, spherical cap or zone etc on Gaussian sphere because of the axisymmetric characteristic. That is, a cylinder is mapped into the great circle, cones into small circles, planes such as top and bottom of a flat-end mill into points, a part of torus into zone and sphere into spherical cap etc. A point P on the sharp edge can be mapped also onto  $n_p$  on spherical cap or zone by interpolating the extreme normal vectors  $n_p^1$  and  $n_p^2$  where  $n_p = \{n \mid n = n_p^1 \cdot (1-u) + n_p^2 \cdot u, |n| = 1, 0 < u < 1\}$  ( Fig. 3). The range resulting from the mapping is called Tool map (T-Map).

According to envelope theory the problem to find the envelope profile, which is involved in cutting at a subscribed instant time, can be reduced to find a set of points, where the scalar product of the surface normal vector  $n$  and velocity vector  $\tau$  on a surface point of a moving object should be zero ( $n \cdot \tau = 0$ ) [Wan86][Rad02].

Any shape of milling tools such as a ball-end mill, which has the convex hull property on the cutting part is applicable to this proposed method. For example, Automatically programmed Tool(APT)-type milling cutters have a convex hull property. Therefore, SV for tool shapes generated by any combination of shape parameters of APT-type tool can be obtained using GM.

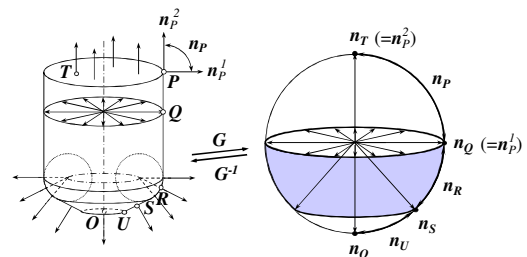


Figure 3. Tool Map (T-Map) : GM of an arbitrary tool shape

The idea to find envelope profile is, simple to speak, that it transforms the problems from Euclidean space to Gaussian sphere using GM, solve the problem in Gaussian sphere and get solutions in Euclidean space through inverse Gauss function. The detailed procedure is shown in the following:

1: At first, surface normal  $N(\eta, \xi)$  of a tool parameterized with  $\eta, \xi$  is calculated by Eq. (3) (Fig. 4(a)):

$$N(\eta, \xi) = \frac{\partial P(\eta, \xi)}{\partial \eta} \times \frac{\partial P(\eta, \xi)}{\partial \xi} \left/ \left| \frac{\partial P(\eta, \xi)}{\partial \eta} \times \frac{\partial P(\eta, \xi)}{\partial \xi} \right| \right. \quad (3)$$

where  $P(\eta, \xi)$  is a surface description of a tool shape.

2: A starting point of each normal vector is translated to the origin of the unit sphere so that the T-Map of the tool is obtained (Fig. 4(b)).

3: Given a moving direction vector  $\tau$  of the cutting tool the C-Map of  $\tau$  is obtained on an another unit sphere (Fig. 4(c)).

4: The both T-Map and C-Map are so overlapped that the origin of T-Map is coincident to the origin of C-Map. Then C-Map divides T-Map into three regions, namely, front- and back-facing region and envelope profile. See Lemma 2 (Fig. 4(d)).

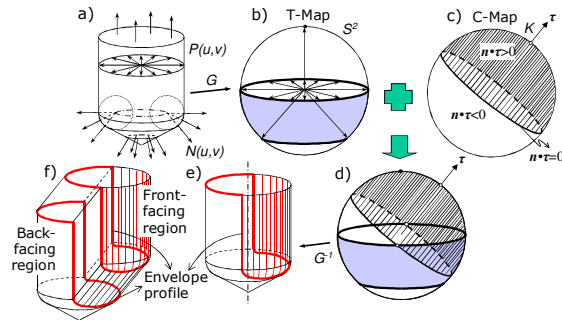


Figure 4. Principles to find SV using GM for translational motion

5: Using inverse GM the closed envelope profile is obtained in Euclidean space. The side of cylinder or cone is to get by *property 2*. (Fig. 4(e)).

6. Swept volume boundary is generated if the back-facing region of the tool at starting position, the front-facing region at ending position and the swept region, where the envelope profiles are linearly interpolated between two subsequent NC blocks, are stitched up (Fig. 4(f)).

### 3.2 Extension to 5 Axis Motion

In general the motion of a cutting tool can be described with both translation and rotation as follows(see Fig. 5 also).

$$V(t, u, v) = V_t(t) + \omega(t) \times \Phi(u, v, t) \quad (4)$$

where  $V_t$  and  $\omega$  is the translational and angular velocity and  $\Phi$  is the tool body parameterized by  $u$  and  $v$  at a prescribed time  $t$ . If there is no rotational motion or  $\omega(t)=0$  for all  $t$ , the translational velocity  $V_t(t)$  remains dominant. It means the resultant velocity  $V$  is constant on entire surface points. The C-Map of the generating vector  $V$  is also constant accordingly because the tool is solid. Otherwise, the tool undergoes five axis motion. Then the velocity  $V$  is hardly constant even at any point on tool surface. Hence, step (3) and step (5) have to be modified slightly in order to apply the same principle for five axis motion consistently. That is, the extended C-Map with respect to the velocity on the axial position along the tool axis is to be acquired and overlapped with the corresponding T-Map. The velocity vector  $\tau$  at a position  $x$  along the tool axis becomes the generating vector of the C-Map. The T-Map which corresponds to the tool component ruling the position  $x$ , must be overlapped with the C-Map.

Let focus on the conic tool component in Fig. 5 for instance. The cone rules the corresponding tool axis interval from  $c$  to  $e$ . Given a position i.e.  $d$  within the interval, The C-Map is generated from the velocity vector  $\tau_d$  at the position  $d$  (Step' (3)). The T-Map is generated from the cone but tagged with a surface position  $S_d$  induced from the axial position  $d$  because the surface position is necessary to determine the envelope profile by inverse Gauss function (Step' (5)).

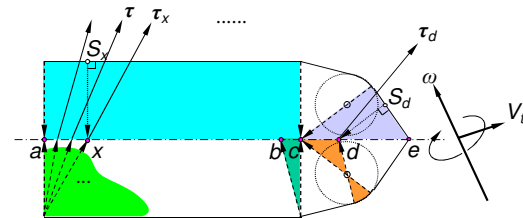


Figure 5. generating vector  $\tau$  on the axis of the tool undergoing 5 axis motion to find extended C-Map

## 4. MILLING PROCESS SIMULATION

### 4.1 Orthogonal Multi-Rays Model

To keep track of the up-to-date work piece SV is generated and subtracted from the current work piece at every G-code block. The Boolean subtraction can be realized with different methods. [Van86] named his work as a *dexel* which means depth segments. Dixel model has several advantages that it can be calculated very fast and accurately because of reduction of Boolean operation from 3 to 1 dimensional space but has also disadvantage that it is dependent on a viewport so that the work piece is not able to be rotated or zoomed up during simulation and is not a volumetric geometry but just an image which can be displayed on output devices. Therefore the

simulated geometry can't be saved as a geometric model for a subsequent usage. Moreover, it is difficult to get intersection with the surface that is almost parallel to the ray.

In this paper, however, orthogonal multi-rays (OMR) model, three 2D arrays of rays aligned with x, y, z axis respectively, is used in order to cope with such drawbacks. OMR is an extended version of single dixel model which is already mentioned in several literatures [Ben97][Mül03]. With OMR a surface can be sampled more exactly than with a single dixel model. See Fig. 6 to compare OMR with single dixel model. Fig. 6 denotes the maximal errors for both models reconstructing the surface from the original sphere with radius  $R=10\text{mm}$  with respect to the number of sampling in the given sampling interval  $2R$ . It implies that OMR requires much fewer rays in comparison with single dixel model to adhere strictly to the same geometric error. For example, OMR necessitates  $147(=3\times7\times7)$  rays while single dixel needs  $2916(=54\times54)$  rays, which are 20 times more than those of OMR, in order to guarantee maximal error 0.2 mm.

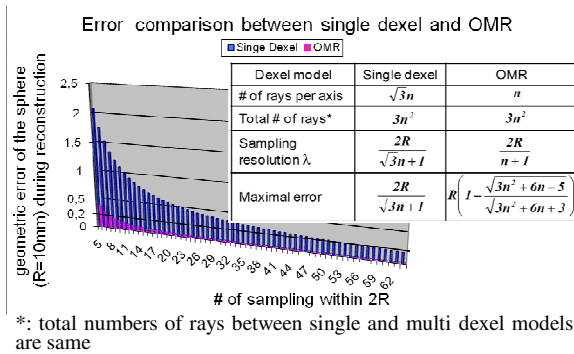


Figure 6. Error comparison between single and OMR model

## 4.2 Sampling the Swept Volume

Five axis SV of the tool is complex and time-varying. Moreover, computing intersections of the deformable object like SV with rays is expensive [Fol97]. That's why the most commercial NC milling verification softwares approximate SV with the sum of prismatic solids like cylinder, cube and sphere where the intersection test of ray and quadratic equations is performed fast [Chu98].

In order to relieve computational burdens and to calculate exact SV instead of the approximation we exploit the frame buffer on GPU by applying hybrid ray-triangle intersection (RTI) test to the triangulated mesh. At first, SV introduced in Section 3 is tessellated with triangles for the purpose of sampling. A triangle is composed of 3 vertices and a color. Its surface normal is calculated from 3 vertices. The color  $i$ , represented as a combination of red, green and blue components, is indexed as ID to identify the triangle  $\Delta_i$ , what is called coloring. Then the triangles

of SV are scan-converted with its own color. At last, the depth is read back for each pixel from the frame buffer on GPU for each side of the bounding box (BV) of SV and updated in OMR model (see Fig. 7, 8 and Alg. 2).

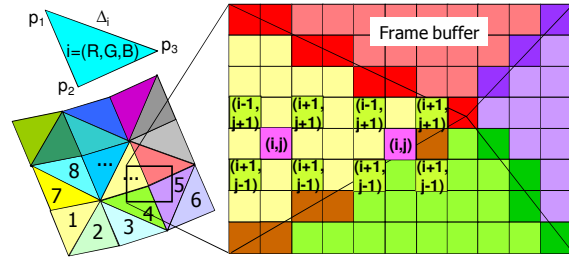


Figure 7. Depth sampling using hybrid RTI test by means of FD and scan-converted triangles in the frame buffer

24 bit depth buffer provides  $1/2^{24}$  depth resolution ( $=0.596\times 10^{-7}$ ) which is good enough for the sampled depth. The sampled position, however, can be misled due to the scan conversion. To eliminate the described errors an exact object-based intersection test is performed. The face distinguisher (FD) of a pixel  $p$  is the pixel congregation which constitutes the pixel  $p$  by itself (flower) and four neighbouring pixels (petals). Fig. 7 and Alg. 1 denote how the FD works. Firstly check if four petals have the same color (or triangle ID) as it traverses along the pixel array  $(i,j)$ . If the IDs of four petals are identical, the depth value is read directly from the depth buffer. If not, however, the triangles corresponding to different IDs within four petals are retrieved and the intersection point of the ray at pixel  $p$  with those triangles is calculated analytically.

### Algorithm 1 Algorithm to test hybrid RTI

HRTI(ScreenX I, ScreenY J, Triangle Delta, depth & Z)

```

for i in [1..I], j in [1..J]
  if ( ColorBuffer[i+j*I] is not white and
    colors of FD[i+j*I] are all same )
    Z[i+j*I] = DepthBuffer[i+j*I];
  else
    for each color k of FD[i+j*I] do
       $\Delta_k = \text{Delta}[k]$ ; //get triangle
       $R_i = \text{GetRay}(i,j)$ ; //get ray
      if ( RTI_test ( $R_i, \Delta_k, \&Z$ ) )
        Z[i+j*I] = z;
      else Z[i+j*I] = z;
    end for
  end
end

```

Briefly speaking we use the frame buffer on GPU to get the depth information in image space by means of reading back and calculate the exact ray-triangle intersection test using FD in object space on demand.

Through coloring and saving facet information several advantages can be obtained: at first, mesh connectivity is unnecessary because the color information from the frame buffer suffices for

identifying the facet. Secondly, the precise 24-bits (or higher) depth information can be read back from the depth buffer in graphics hardware with a proper scaling for free. The proposed algorithm to sample SV is as follows:

---

**Algorithm 2** Algorithm to sample the tessellated SV using hybrid RTI

---

```

SamplingSV(Triangle SV)
glShadeModel(GL_FLAT); //set OpenGL shading state
glDisable(GL_LIGHTING);
glEnable(GL_DEPTH_TEST); //set frame buffer state
for each axis i along x, y, z do
//set viewing transformations for BV of SV
glViewport(0,0,I,J) //define viewport
glOrtho(...) //set orthographic projection matrix
gluLookAt(...) //set a viewing matrix
glDepthFunc(GL_LESS); //set depth function
render_scene(SV); //draw SV
glReadPixels(...); //read back the color and depth buffer
HRTI(I, J, Delta,&Z[0][i]); //test hybrid RTI
SaveDepth(i,0,Z);
glDepthFunc(GL_EQUAL); //set depth function
render_scene(SV);
glReadPixels(...);
HRTI(I, J, Delta,&Z[1][i]); //test hybrid RTI
end for
end

```

---

Initially OpenGL primitives should be shaded flat because one computed color should be assigned to all the pixel fragments generated by rasterizing a single triangle and the lighting should be disabled in order that the computed color of a vertex is not influenced by the light so that it is the current color at the time the vertex was specified. After primary GL setting, SV is rendered for the sake of sampling. The viewport and projection transformation are computed based on the axis-aligned BV of SV. The frame buffer is read from near and far side of BV along x, y, z axis with the option GL\_LESS because the nearest fragments from the viewpoint are to be selected, which are the outmost points of SV at the same time. At last, the surface of SV is sampled through hybrid RTI test (see Alg. 1) for pixel position (i,j) by the use of the depth and color information from frame buffer.

### 4.3 Boolean Operation

Milling is the metal cutting process which removes the material cut by tool from the work piece continually. The formulation of such a process is well written mathematically as follows.

$$R = W - \left( \bigcup_{k=1}^{k=n} SV_k \right) = (((W - SV_1) - SV_2) - \dots) - SV_n \quad (5)$$

$$SV_i = \lim_{n \rightarrow \infty} \bigcup_{k=0}^{k=n} TV \left( i - 1 + \frac{k}{n} \right) \quad (6)$$

$$TV(t) = X(t) \cdot TOOL_{local} \quad (7)$$

$X$  is the transformation matrix and  $TOOL_{local}$  is the cutting tool located in the local coordinate system.

The swept volume  $SV_i$  of the tool at the  $i^{\text{th}}$  code block is the whole summation of the instant tool volumes  $TV$  within the normalized time interval  $[0,1]$  (Eq. (6)). Then the machined work piece  $R$  is to be calculated either by the subtraction of the raw stock  $W$  from the summation  $USV_k$  of swept volumes or by the continual subtraction of the actual work piece from the current swept volume  $SV_i$  (Eq. (5)). Our concern is to find the boundary surface  $\Phi R$  of the in-process work piece  $R$  and that means the boundary of each swept volume  $\Phi SV_i$  is to in order to update  $\Phi R$ .

$$\Phi SV_i = \Phi TV^-(i-1) + \lim_{n \rightarrow \infty} \bigcup_{k=0}^{k=n} \Lambda \left( TV \left( i - 1 + \frac{k}{n} \right) \right) + \Phi TV^+(i), i \geq 1 \quad (8)$$

The boundary surface  $\Phi SV_i$  is the sum of each contributing surfaces (Eq. (8)).  $\Phi TV^-$  and  $\Phi TV^+$  are the back-facing and front-facing surface of the tool respectively. Operator  $\Lambda$  is responsible for finding the envelope profile which is embedded on the surfaces of instant  $TV$  and  $SV$  at the same time (refer to section 3).

OMR is used as the model realizing Boolean operation. Using this model the Boolean subtraction can be simplified from the 3D geometrical CSG operation to 1D numerical subtraction for each axis so that the complex computation can be avoid. The SV is sampled and the work piece is constructed in the context of multi dixel model. Fig. 8 denotes the elementary operations of the dixel model which cuts SV from work piece along a ray at the different time.

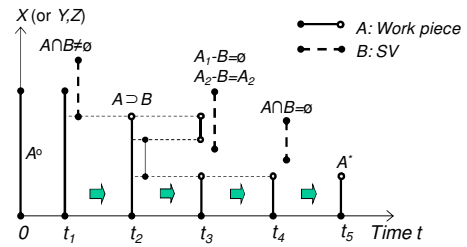


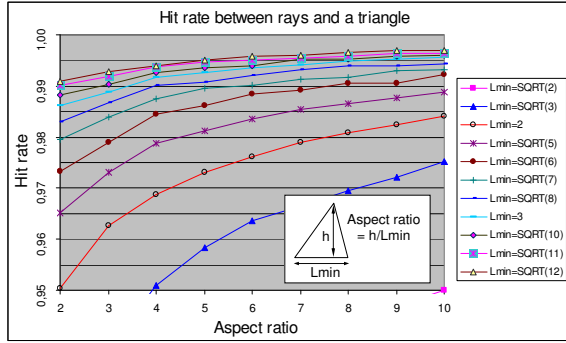
Figure 8. Elementary updating operations per pixel

### 4.4 Surface Reconstruction

Marching cube (MC) algorithm was originally used in the medicine to extract and visualize the iso-surface which has the same properties e.g. density from computer tomography [Lor87]. MC treats each cell independently to construct surface so that MC needs no connectivity information of mesh. MC has, however, following several limitations which result from the discretization.

- MC can't represent the feature with sharp edge or corner (sharp edge problem).
- The surface accuracy depends absolutely on the resolution of grids (sampling resolution).

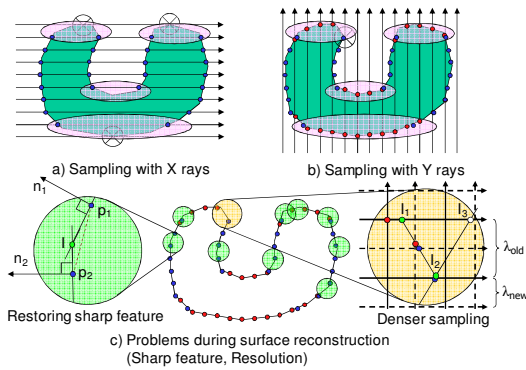
Theoretically sharp features are able to be recovered completely if the sampling resolution  $\lambda$



\*: grid size = 0.5 mm

**Figure 10. Hit rate of a triangle with orthogonal rays w.r.t. aspect ratio\***

converges to zero. In practice, however, the resolution



**Figure 9. Optimal sampling resolution and preserving the sharp feature**

$\lambda$  is determined as large as possible as long as the sampling deviation doesn't exceed the prescribed error.

To comprehend the problems it is explained with the polygon on the 2D plane. To solve the first problem the intersecting position as well as the outward normal vector to the line segment is saved when a ray meets a polygon (Fig. 9(a)-(b)). At first, the angle between two successive normals  $n_1$  and  $n_2$  is measured. If the angle is greater than a threshold angle  $\alpha$  then intersection point  $I$  is inserted between  $p_1$  and  $p_2$  and two supplementary line segments  $p_1I$  and  $p_2I$  are inserted in place of line segment  $p_1p_2$  so that the sharp feature is preserved (see Fig. 9(c) left) [Pur96][Kob01].

To the second problem, the sampling resolution  $\lambda$  should be so determined that the line segment of the polygon can be intercepted by at least one of rays on the plane. The Nyquist criterion in sampling theory says the sampling frequency  $f$  must be at least twice greater than the maximal frequency of a signal to recover the signal [Kuo02]. This criterion is adopted to our problem only with restricted meaning because it should be solved not in the frequency domain but in object domain. So the sampling resolution  $\lambda$  in the object domain, corresponding to the sampling time  $\Delta t$

in the time domain, is determined less than a half of the shortest one among projected lines of the line segments onto X and Y axis. By proposed methods the 2D polygon can be recovered completely from the original object.

Analogous to the polygon in the plane Fig. 10 shows the hit rate between a triangle in space and OMR with respect to the aspect ratio for the different minimal length of the triangle. The complement of hit rate is the probability to omit triangles during sampling and it causes the local approximation as a result. Fig. 9(c) right denotes that the sharp feature could be recovered by inserting two intersection points  $I_1$  and  $I_2$  with denser sampling resolution.

## 5. IMPLEMENTATION AND RESULTS

The algorithms described in this paper are implemented in C++, OpenGL version 2.0 [Shr05] and Standard template library (STL). The current version of the simulation program runs on AMD Athlon XP 2500+ 1.82 GHz and 1 GB RAM which is equipped with a GeForce 7600GS graphics accelerator. The ongoing implementation of SV algorithm for polyhedron and the milling simulation is tested. The open codes of [Möl97] for the ray-triangle intersecting test is used and the execution results are rendered as shown in Fig. 11(a)-(c) shows the SV of a moving tool: (a) and (b) are SV of 3 and 5 axis motion that was constructed by the proposed method via GM, and (c) is SV with GL\_FLAT option, which is prepared for sampling.

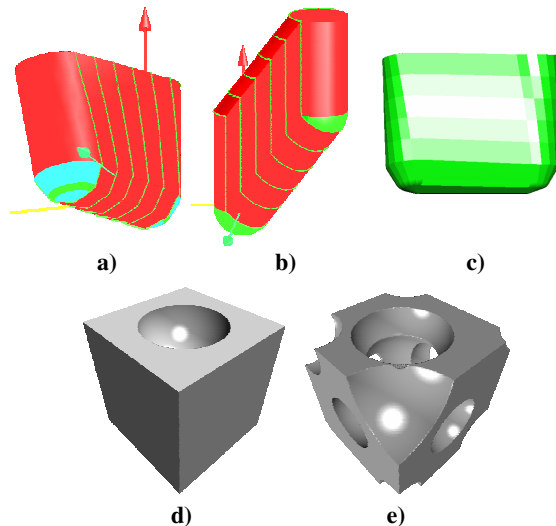
Fig. 11(d)-(e) denotes the interactive volumetric operation between spheres with variable radii and the cube of  $50^3 \text{ mm}^3$ . Grids are aligned with 128 for each axis so that the sampling resolution is 0.39 mm/grid. It shows the first and last Boolean subtraction of spheres from the cube.

## 6. CONCLUSION AND OUTLOOK

In this paper, a novel approach to finding SV of the tool undergoing five axis motion via GM and the algorithms and intermediate results for fast 5 axis milling simulation to keep track of the in-process work piece using OMR, object sampling/data extraction by graphics hardware and iso-surface reconstruction is presented. The method to find SV provides an intuitive and comprehensive understanding of cutting behavior for the generalized tool shape. The sampling technique and hybrid RTI test using commodity computer graphics hardware provides an accelerated tool to perform cutting simulation from the changeable polygonal mesh with the time. The ongoing results predict that the proposed method could be used for a practical application to simulate five axis milling process.

Following subjects are considered as near future research targets.

- Due to the use of a fixed grid, i.e.  $128^3$  grids, the dimension of a renderable object is quite restricted. Adaptive resolution may be helpful to make the machining simulation of the common parts in die and molding industries possible.
- Intelligent controller equipped with real time simulator capable of error analysis and tool path optimization is aimed as a next theme.



**Figure 11.** a), b) SV generation of 5 and 3 axis milling motion, b) flat rendering of SV for the sake of sampling and c) Boolean subtraction of spheres from a cube.

## 7. ACKNOWLEDGEMENTS

This work is a part of the research grant 15096BR in the IGF program fully funded by the German Federal Ministry of Economics and Technology (BMWi) via Arbeitsgemeinschaft industrieller Forschungsvereinigungen "Otto von Guericke" e.V. (AiF). Especially we thank our project partners, andron GmbH and GIB CAD&CAM mbH, for providing the machine tool controller and CAM-API.

## REFERENCES

- [Ben97] Benouamer, M.O., Michelucci, D.; Bridging the gap between CSG and Brep via a triple ray representation; ACM Symposium on Solid Modeling and Applications, Proceedings of the 4<sup>th</sup> ACM symposium on Solid modeling and applications, Atlanta, GA, pp.68-79, 1997
- [Bla97] Blackmore et al.; Sweep-envelope differential equation algorithm and its application to NC machining verification, CAD, Vol. 29, pp. 629-637, 1997
- [Chu98] Chung et al.; Modeling the surface swept by a generalized cutter for NC-verification, CAD, Vol. 30, pp. 587-593, 1998
- [DoC83] Do Carmo, M.; Differentialgeometrie von Kurven und Flächen, Vieweg Verlag, 1983
- [Fol97] Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.; Computer Graphics : Principles and practice, Addison Wesley, 1997

- [Har98] Harris et al.; Handbook of Mathematics and Computational Science, Springer-Verlag, 1998
- [Hui94] Hui, K.; Solid sweeping in image space – application to NC simulation, The Visual Computer, Vol.10, pp. 306-316, 1994
- [Jac91] Jacobs, B.; The expanding role of simulation, Tooling & Production, Vol. 71 no. 7, pp. 28-31, 2005
- [Jer89] Jerard et al.; Approximate methods for simulation and verification of numerically controlled machining programs, The Visual Computer, Vol.5, pp.329-348, 1989
- [Kaw91] Kawashima et al.; A flexible quantitative method for NC machining verification using a space-division based solid model, The Visual Computer, Vol. 7, pp.149-157, 1991
- [Kob01] Kobbelt, L.P., Botsch, M., Schwanecke, U., Seidel, H.; Feature Sensitive Surface Extraction from Volume Data, SIGGRAPH 2001 Conference Proceedings, pp.57-66
- [Kuo02] Kuo, B. C., Golnaragh, F.; Automatic Control Systems, 8<sup>th</sup> Edition, Wiley, ISBN: 978-0-471-13476-3, 2002
- [Lee06] Lee, S.-W., An Approach to Swept Volume Generation for NC machining using Gauss Map, In: Proceedings of 11th IEEE International Conference on Emerging Technology and Factory Automation (ETFA), Prague, pp.1137-40, ISBN: 1-4244-0681-1, 2006
- [Lor87] Lorensen, W.E., Cline, H.E.; Marching cubes: A high resolution 3D surface construction algorithm, Computer Graphics, 21(4):163-169, 1987
- [Möl97] Möller, T., Trumbore, B.; Fast, minimum storage ray-triangle intersection. Journal of graphics tools, 2(1):21-28, 1997
- [Mül03] Müller, H., Surmann, T., Stautner, M., Albersmann, F., Weinert, K.; Online Sculpting and Visualization of Multi-Dexel Volumes; 8<sup>th</sup> ACM Symposium on Solid Modeling and Applications, Seattle, WA, pp.258-261, 2003
- [Owe05] Owens, J.D. et al.; A Survey of General-Purpose Computation on Graphics Hardware, In: Eurographics 2005, State of the Art Reports, pp. 21-51, 2005
- [Par05] Park et al.; Hybrid cutting simulation via discrete vector model, CAD, Vol. 37, pp. 417-430, 2005
- [Pur96] Purgatofer, W., Tobler, R.F., Galla, T.M.; ACSGM-An adaptive CSG meshing algorithm, In Information Geometers Ltd, editor, Proceedings of CSG96, pages 17-31, Winchester, UK, April 1996
- [Rad02] Radzevich, S.P., "Conditions of proper sculptured surface machining", CAD, Vol. 34, pp. 727-740, 2002
- [Rot01] Roth et al.; Surface swept by a toroidal cutter during a 5-axis machining, CAD, Vol. 33, pp. 57-63, 2001
- [Shr05] Shreiner, D. et al.; OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2.0 (5th Edition), Addison-Wesley, ISBN 0321335732, 2005
- [Van86] van Hook, T.; Real time shaded NC milling Display; Dallas, August 18-22, SIGGRAPH '86, Volume 20, Number 4, 1986
- [Wan86] Wang et al.; Geometric modeling for swept volume of moving solids, IEEE Computer Graphics and Applications Vol. 6, no. 12, pp.8-17, 1986