Real Time Simulation and Visualization of NC Milling Processes for Inhomogeneous Materials on Low-End Graphics Hardware

Andreas Holger König and Eduard Gröller

{ koenig | groeller } @cg.tuwien.ac.at

Institute of Computer Graphics, Vienna University of Technology

Karlsplatz 13/186/2, A-1040 Wien, Austria,

http://www.cg.tuwien.ac.at/home/

Abstract

Simulation and visualization of NC milling processes has become an important step in computer aided manufacturing. The usage of stock materials with specific locally varying properties (like density, accuracy, color,...) becomes more and more important with new technologies emerging in the material industry. Our new approach, using volumetric representation, has been adapted to this needs and copes with inhomogeneous material properties. Taking color as one possible material property, our approach enables the visualization of milled wood or compound materials.

Furthermore, our approach has been developed with the usage of low-end graphics hardware in mind. The algorithms have been optimized to ensure interactive update rates even on standard personal computers without hardware graphics acceleration.

Keywords: *NC milling simulation, dexel approach, inhomogeneous material properties*

1. Introduction

NC milling simulation using computer graphics techniques was proposed some years ago to verify NC programs rapidly and precisely.

Simulation systems exist for the two common types of NC milling: 3-axis milling and 5-axis milling. Almost all industrial parts can be milled with three degrees of freedom x, y, z. The values of x and y describe the movement of the cutter in the plane defined by the top of the workbench, whereas z is the height of the cutter above the workbench. The axis of the cutter is fixed to be vertical. Some applications require the cutter axis to be tilted (5-axis milling). This introduces two additional degrees of freedom: α and β being the rotational tilt of the cutter axis. The generation of a tool path for 5-axis milling is quite sophisticated. Therefore, 5-axis milling is not as common as 3-axis milling.

During the simulation of NC milling usually graphical representations of the workpiece and the milling tool (the cutter) are displayed. The cutter can be observed moving along the toolpath, which is defined by the NC program. As the cutter is removing material, intersection calculations

are used to determine the removed parts of the workpiece. The workpiece is updated for every frame in the image sequence. The geometric representations of workpiece and cutter are displayed on the screen using more or less sophisticated computer graphics techniques.

The most important aspect in NC milling simulation is error assessment. The workpiece is compared to the desired design part. The local deviation can be used to map the error into color hue, which can easily be interpreted by the NC programmer. Special constraints to error simulation like cutter wear-out are usually not taken into account. An accurate simulation of this phenomenon would require detailed knowledge about material properties of the workpiece like density and heat coefficents.

Two major approaches to the simulation of NC milling processes have been developed: the exact, analytical approach and the approximate approach. The main problem with the accurate approach (mostly done in CSG) is its computational expense. The cost of simulation is reported to be $O(N^4)$ [9], where N is the number of tool movements. A complex NC program might consist of ten thousand movements, making the computation intractable. In order to increase efficiency, a number of approximate simulation methods have been devised. The computational cost of these methods simplifies to $O(N)$.

In the next section a detailed discussion of the previous approaches is given. Then the dexel approach, being the basis of our work is described. Our extensions and accelerations to this approach are discussed. Special emphasis will be given to inhomogeneous material properties, which are accounted for in our approach. Finally some results will be shown.

1.1. Accurate approaches to NC milling

Kawashima [13] used a special geometric modelling method called *Graftree* to speed up his solid modeling approach. Using CSG modelling, their approach allowed accurate and precise representation of the workpiece and the tool, while an octtree helped to decrease the number of ray-intersection calculations in rendering. NC milling was simulated by using a Boolean difference operator. Another approach to find the exact representation of the envelope of a swept volume was chosen by Sourin [1]. He described the cutting tool analytically with the use of procedurally implemented time-dependent defining functions. Boolean set operations are defined by Rvachevs *R-functions* [5]. The milling process is visualized by ray tracing the analytically defined scene, which is a very time-consuming process.

1.2. Approximate approaches to the simulation of NC milling

Van Hook [10] developed a real time shaded display of a solid model being milled by a cutting tool which follows an NC path. This approach utilized a *dexel* (depth element) representation of the workpiece and cutter geometries. The data structure is a run length encoded version of a volumetric data representation. An update rate of ten cutting operations per second was attained by using Boolean set operations on the one-dimensional dexels. The view point dependency of this approach was overcome by an extension of the method by Huang [12]. Huang also introduced the possibility of error assessment to Hooks method. Takafumi also used an extension of the z-buffer method (called *G-buffer*) to simulate NC milling [8]. A completly different approach was chosen by Jerard [2]. The design surface is approximated by a polygonal mesh, where the surface normals or other arbitrarily chosen vectors at the mesh points are intersected with a polygonal approximation of the

tool during simulation (therefore this approach is known as the *lawn mowing* approach). Yang [7] developed a method suitable for *wire-EDM* (wire cut electric discharge machining), where cutting is only done at the sides of the workpiece. Glaeser [4] used differential geometric techniques to efficiently generate the swept volume of a moving cutter. Intersection calculations are done in data structure that is an extension to the z-buffer data structure, called Γ-buffer.

For an overview of the techniques described see table 1.

2. The extended dexel approach to NC milling

The dexel approach described in [10] and [12] is a specialized version of run length encoding the stock material. A dexel (depth element) is a rectangular solid extending along one specified direction of the object to be represented. In [10] this direction is aligned with the viewing vector. The reolution of the dexel structure is fixed according to the screen reolution. Therefore color values stored with the dexels can be directly copied to the framebuffer for the visualization of the dexel structure. The approach needs enormous amounts of storage and interactive updates of the milling scene are only possible on special graphics hardware.

An independent coordinate system for the objects is used in [12]. Therefore the direction, in which the dexel extend, can be arbitrary. As this direction is no longer aligned with the viewing vector a special visualization technique has to be employed. The creation of a shaded contour display, consisting of lines, is discribed in this work. This visualization technique is also too expensive for low end graphics hardware.

For our purposes the dexel approach has been simplified and adapted to the needs of 3-axis milling. First of all, we are using a lower resolution than the original methods. This reduces the amount of data which has to be processed, therefore speeding up the method. A visualization technique had to be developed for this low resolution, which produces polygonial output. Rendering of these polygon meshes can utilize one of the standard 3D APIs like OpenGL.

Second, the dexel structure itself has been simplified. Our investigations in the NC milling industry have shown, that in 95 percent of all cases just simple forms of 3-axis milling are used. Most workpieces are flat plates, which are only machined from above. Our work is intended to give an inexpensive solution for these most common machine arrangements. No internal holes or slots at the side faces of the workpiece are produced by these simple kinds of NC machining. Therefore, no multiple layers of dexels have to be stored. As no linked list data structures have to be used like in [10] and [12], our dexel structure is simply a two-dimensional array. Detailed information on the dexel data sructure will be given in the following section.

2.1. The dexel set

Let a dexel data set D be defined as the quadruplet (R, X, L, C), with

- $R = (R_x, R_y) \in \mathbf{N}^2$ describes the *resolution* of the volume data set in x and y dimension.
- X is a set of $R_x * R_y$ *dexels*. The contents of a dexel will be described later on.
- $L \in \mathbf{R}^3$ is the *location* of the volume data set in 3D space. This point is usually the centre point of the grid plane containing the dexels.
- $C = (C_x, C_y, C_z)$, with $C_x, C_y, C_z \in \mathbf{R}^3$, is the local *coordinate system* describing the orientation of the dexel data set in 3D space.

A dexel is a rectangular solid extending in the z direction of the volume data set (refer to figure 1). Let a dexel $dex \in D.X$ be defined as the triplet $(top, bottom, \vec{n})$ where

Method	workpiece	cutter	ea	tool shape	vp	swept volume	axes
Graftree[13]	Octree CSG		yes	arbitrary	no	explicit as CSG	5
Rvachev functions method[1]	analytical		no	arbitrary	no	analytically	5
dexel approach[10]	dexel		no	arbitrary	yes	calculated in image space	3/5
view point independent dexels[12]	dexel		yes	arbitrary	no	calculated in image space	5
G-buffer[8]	ext. z-buffer		yes	arbitrary	yes	scanning of G-buffer	3
lawn mowing[2]	mesh with normals	polygons	yes	ball or cylinder	no	explicit as polygons	3/5
wire EDM[7]	R-buffer (cylindrical z-buffer)		yes	wire	no	explicit as polygons	4
Γ-buffer[4]	Γ-buffer	polygons	yes	arbitrary	no	scanning of Γ-buffer	3/5

Table 1. Comparison of previous techniques. Abbrevations used for the table columns: *ea*... error assessment, and *vp*... view point dependency.

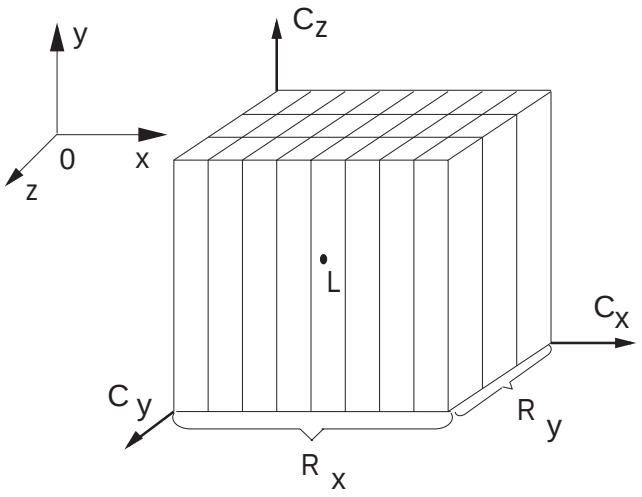

Figure 1. Dexel data set for a cube shaped object

- top is the top height-value of a dexel.
- $bottom$ is the bottom height-value. The number of bits used to describe these values determines the resolution which may be achieved in z-direction. If $top, bottom \in [0..2^m - 1]$ where m is the number of bits, then 2^m different height levels may be described by top and $bottom$.
- $\vec{n} \in \mathbf{R}^3$ is the *height gradient vector* for the dexel. It is precalculated by the central difference method using the top values of the dexel set.

$$dex_{i,j}.\vec{n} = \begin{pmatrix} dex_{i+1,j}.top - dex_{i-1,j}.top \\ dex_{i,j+1}.top - dex_{i,j-1}.top \\ -1 \end{pmatrix} \quad (1)$$

with $dex_{i,j} \in D.X$. The use of $\vec{n}$ singnificantly speeds up the simulation of milling free form surfaces, which is described in section 2.4. As 3-axis machining of free form surfaces is most common in industrial NC milling, this work will focus on improvements in this field.

2.2. Workpiece and cutter representations as dexel data sets

Two different objects modeled with the dexel representation are needed for our approach of simulating NC milling:

a workpiece DW and a cutting tool DC.

2.2.1 The workpiece

In 95% of cases, NC milling is dealing with 3-axis milling of flat plates. Therefore, we will describe the setup of the workpiece for the case of a cube shaped solid. The following algorithm fills the workpiece dexel structure $DW.X$:

```
for all dexels dex[i,j] in DW.X
   if (i=0 or j=0 or i=DW.Rx or j=DW.Ry)
      dex[i,j].top=0
   else
      dex[i,j].top=height
   dex[i,j].bottom=0
```

where $height$ is the defined height value of the desired block of material. Setting the height of border dexels to 0 assures, that the dexels on the edges of the dexel set produce a surface during the visualization step.

2.2.2 The cutter

Ball-end cutters are the most common tools in milling free form surfaces. Therefore the geometry of a ball-end cutter is described. The geometric definition of a sphere is used to describe the desired shape of the cutter DC. For every location (i, j) on the dexel grid of DC a ray is cast in direction of $DC.C_z$. This ray is analytically intersected with the defining sphere. The intersection points define the geometry of the dexel $dex_{i,j} \in DC.X$. Three cases may occur:

- Two intersection points ((i, j, z_b) and (i, j, z_t)) are found. $dex_{i,j}.bottom$ is set to z_b, the smaller one of the calculated z values. $dex_{i,j}.top$ is set to z_t.
- One intersection point is found: (i, j, z). Both $dex_{i,j}.bottom$ and $dex_{i,j}.top$ are set to z.
- No intersection point is found. Both $dex_{i,j}.bottom$ and $dex_{i,j}.top$ are set to 0.

Cylindrical cutters (as well as the cylindrical shaft of ball-end cutters) may be defined in a similar way.

2.3. Cutting the workpiece

An instances of motion approach (described in [12]) is used to move the cutting tool along the entire predefined tool path. At evenly spaced discrete locations, close enought that no dexels are skipped, the intersection operation is evaluated for cutter and workpiece.

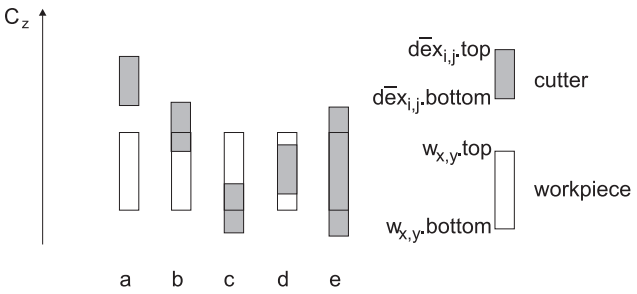

Figure 2. Possible intersections of cutter and workpiece dexels

The dexels of the cutting tool $DC.X$ are transformed into the coordinate system of the workpiece $(DW.C, DW.L)$. If a transformed dexel $\bar{dex}_{i,j}$ is within the boundary of the workpiece dexel structure, then the nearest neighbour $w_{x,y} \in DW.X$ is found. This discrete sampling approach could lead to aliasing artefacts. A better solution would be to take all dexels into account, which are covered by the projected area of the cutter dexel. Cutting should then be applied to all of these dexels, weighting the influence of the cutting operation by the fraction of the dexel area which is common to both the workpiece and cutter dexel. This approach is computationally much more expensive than chosing a single candidate dexel for cutting. As the visual differences in results are neglegible (present, but not visible), the inexpensive approach can be chosen.

The cutting operation is simplified to comparing the top

and bottom values of these two corresponding dexels. The following cases (depicted in figure 2) may occur:

- $w_{x,y}.top < \bar{dex}_{i,j}.bottom$: The cutter dexel is above the workpiece dexel. No intersection is found. (figure 2a)

- $w_{x,y}.bottom < \bar{dex}_{i,j}.bottom < w_{x,y}.top < \bar{dex}_{i,j}.top$: The cutter removes the top part of the workpiece dexel. $w_{x,y}.top$ is set to $\bar{dex}_{i,j}.bottom$. (figure 2b)

- $w_{x,y}.bottom < \bar{dex}_{i,j}.top < w_{x,y}.top$ and $\bar{dex}_{i,j}.bottom < w_{x,y}.bottom$: The cutter removes the bottom part of the workpiece dexel. $w_{x,y}.bottom$ is set to $\bar{dex}_{i,j}.top$. (figure 2c)

- $\bar{dex}_{i,j}.top < w_{x,y}.top$ and $w_{x,y}.bottom < \bar{dex}_{i,j}.bottom$: The cutter removes an inner part of the workpiece dexel (figure 2d). This would produce two dexels for $w_{x,y}$. We are assuming, that this case can not be produced in our 3-axis milling simulation. Changes to the workpiece dexel are ignored.

- $w_{x,y}.top < \bar{dex}_{i,j}.top$ and $\bar{dex}_{i,j}.bottom < w_{x,y}.bottom$: The cutter removes the workpiece dexel completely (figure 2e). $w_{x,y}.top$ and $w_{x,y}.bottom$ are both set to zero.

2.4. Visualization

Several methods for the visualization of dexel data sets do already exist. Van Hook [10] displays the data set by directly copying it to the video frame buffer, whereas Huang [12] uses a contour display method to render the dexel structure. As we are using a lower resolution than in the above approaches, we had to develop a new visualization technique which produces polgonial output.

Converting the 3-axis milled dexel structure into a surface representation is straightforward. The top locations of four neighbouring dexels $dex_{i,j}.top$, $dex_{i+1,j}.top$, $dex_{i+1,j+1}.top$, $dex_{i,j+1}.top$ determine the vertices of two triangles. Evaluting this for all dexels gives a representation of the top surface without holes (see figure 3).

Non-trivial cases occur, when top and bottom locations of a

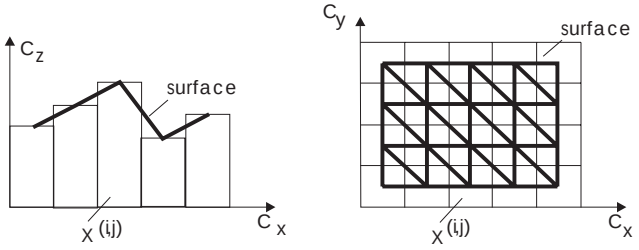

Figure 3. Connecting top locations for surface generation

dexel have the same value (see figure 4). It has to be defined, whether such dexels are still contributing to the border of the hole (like in figure 4a the leftmost and rightmost dexels) or if they are already part of a hole (the dexel produces no surface, like in figure 4b the leftmost and rightmost dexels). When only the top surface of the dexel structure has to be created, both alternatives depicted in figure 4a and 4b could be used. Dexels with both top and bottom value set to zero could be treated as lying on the border of the hole (still within material, figure 4a) or as lying in the hole (no material, figure 4b).

When the bottom surface of the dexels structure is also created, treating these dexels as border dexels (producing surface) leads to incorrect results. Refer to figure 4c and 4d for a comparison. The next section will focus on a special algorithm for the creation of visually correct surface representations for such non-trivial dexels.

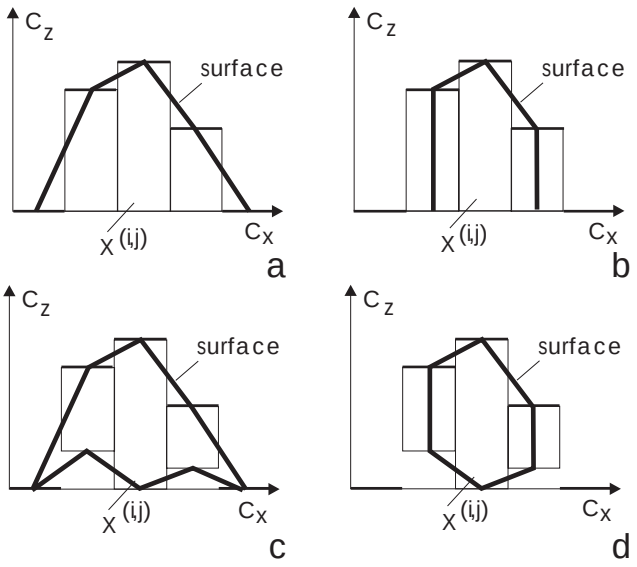

Figure 4. The non trivial case of connecting top and bottom surfaces

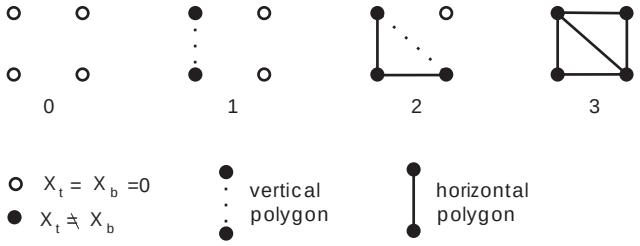

Figure 5. Possible cases of border/hole distribution among neighbouring dexels

2.4.1 Obtaining visually correct vertical edges

An adapted version of the well known marching cubes algorithm [11] is employed to treat all possible cases in a visually correct manner. Four neighbouring dexels are used to create a surface part. First they are classified, whether both top and bottom values are zero (”holes“) or not (”material“). This knowledge is used to create an index into a table of all possible cases. As with the original marching cubes algorithm, all possible cases can be derived from a basic set by rotation and inversion. See figure 5 for the setup of basic possibilities. Some cases (like two holes on a diagonal) can not be produced by rotation or inversion from the basic set. These cases dissolve automatically into basic cases, when successive rows of dexels are processed. Four different cases may occur (refer to figure 5):

- All four dexels are classified as ”hole“ (figure 5(0)). No polygons are created at all.
- Two dexels sharing an edge in the square of interest are classified as ”material“ (figure 5(1)). A vertical polygon is created between them to close the side face of the dexel structure.
- Three dexels are classified as ”material“ (figure 5(2)). One horizontal polygon is created to close the top surface, one vertical polygon along the diagonal is created to close the side face.
- All four dexels are classified as ”material“ (figure 5(3)). Two horizontal polygons are created to close the top surface.

The use of the vertical polygons along the diagonal reduces the influence of object-space aliasing. Instead of having a set of vertical polygons where two adjacent faces are either coplanar or meet perpendicularly, now also a angle of 45 degrees is possible. Nevertheless, aliasing is still visible. The next section deals with a method to overcome this problem.

2.4.2 Antialiasing

Aliasing occurs at the side faces of the dexel structure. The top surface polygons are shaded with the precalculated height gradient normal vectors, therefore aliasing artifacts are rare. As the side faces are always vertical, their gradient vectors lose one degree of freedom and aliasing is introduced. To overcome this problem, a standard antialiasing technique, *blurring*, was adapted to be used for the normal

vectors. For 2D raster images blurring provides antialiasing by averaging the weighted color values of adjacent pixels. This convolution operation averages the color values and high frequency changes are eliminated from the image. We are using averaging on 3D vectors: the normal vectors of neighbouring dexels and the normal vector of the dexel are weighted and summed up to get the blurred normal vector for the dexel. See figure 6 for a graphical representation of the process. When these antialiased normal vectors are

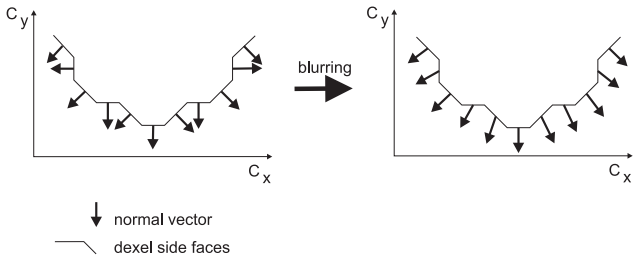

Figure 6. Antialiasing by blurring the normal vectors

used for shading, the lighting calculation for the polygons is modified. The change of color between adjacent polygons is less dramatic than when using the exact normals of the side face polygons. The geometric representation is of course unaltered and still precise (with respect to the underlying resolution). See figure 7 for a comparison.

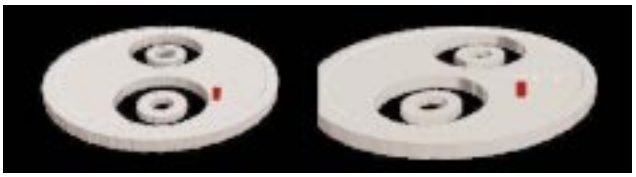

Figure 7. shading without antialiasing (left) and with antialiasing (right)

2.5. Error assessment

Error assessment can easily be achieved by comparing the dexels of the workpiece $DW.X$ with the dexels of the design part $DD.X$. Local error assessment compares single dexels whereas global assessment compares all the dexels. Overcutting and undercutting may occur (see figure 8). Overcutting occurs at regions, where already too much material has been removed. Undercutting means, that there is still material in excess at some location.

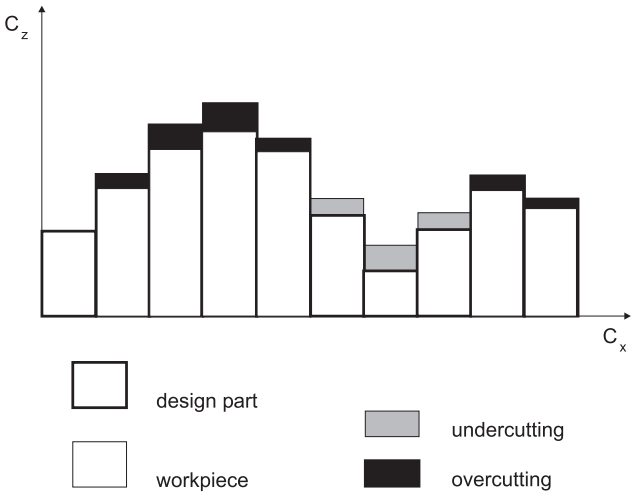

Figure 8. Comparison of workpiece and design part dexels gives information on over- and undercutting

2.5.1 Local error assessment

Two strategies have been investigated to visualize the local error.

- A rough overview about regions, where the tool path has to be reprogrammed, is gained, when design part and workpiece are rendered simultaneously. If the workpiece is drawn transparently, even more information about the undercutting error is visible. For an ex-

ample see figure 9. If overcutting has to be visualized, the design part has to be drawn transparently.

Figure 9. Error assessment by drawing the workpiece transparently over design part

- A more sophisticated stategy is to encode the local error in the surface representation of the workpiece by the use of color. The difference between workpiece dexel and design part dexel ($dex_{i,j}.top - dp_{x,y}.top, \quad dex_{i,j} \in DW.X, dp_{x,y} \in DD.X$) is mapped via a transfer function into a color value. This value is used for the vertices of the top surface part created by this dexel. One possible transfer function could map overcutting to red, whereas undercutting is mapped to green. If no error occurs, the vertices will be assigned the color white. For an example of this visualization method see figure 10.

2.5.2 Global error assessment

Global error assessment is done by summing up the local errors.

$$E = \sum_{i,j} \left(dex_{i,j}.top - w_{i,j}.top\right)^2 \tag{2}$$

where $dex_{i,j} \in DW.X, w_{i,j} \in DD.X$ and E is the global error. Information about the global error can be given as an absolute value (E) or as a relative value (E_{rel}):

$$E_{\text{rel}} = \frac{E}{\sum_{i,j} w_{i,j}.top} \tag{3}$$

2.6. Material properties

Nowadays often special metal alloyes, compound and other syntethic materials are used for NC milling. Therefore homogenity throughout the whole block of stock material can no longer be assumed. In our new approach to NC milling simulation, this fact is taken into account. We are investigating the possibilities when dealing with material properties in the next sections. Assuming that material properties may vary in different regions of the workpiece, various fields of application are possible. Some examples will be given: If cutter speed limits can be defined for different regions of the workpiece, the automated tool path generation can be further optimized in means of time. Parts of the workpiece with higher density have to be milled slower in order to not wear out the cutter too early. Different materials could also be milled with different rotation speeds of the cutting tool. Some other machine parameters could also be controlled with varying material properties. For instance during milling of a compound material consisting of metal and wood the cooling liquid of the NC machine has to be stopped during wood milling, in order to prevent the wood from soaking up the polluted liquid. Another application could be the definition of varying error tolerances for different parts of the workpiece. Automated tool path generation could generate high accuracy movements for, e.g., bearings and surfaces of contact, whereas not so important parts could be treated with higher error tolerance limits.

Color as a material property must be taken into account during the simulation of NC milling. Milling a piece of wood or a block of compound material will not give the same visual impression as milling a block of grey aluminium. Furthermore, color is a simple and intuitive example of a varying property, which may change with high frequency. The next sections will focus on the usage of color for the simulation of NC milling processes.

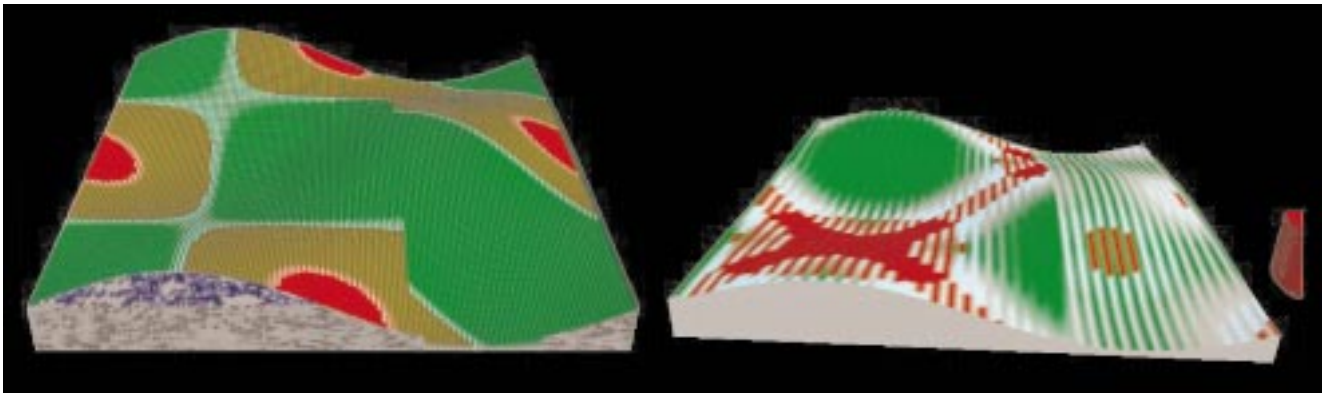

Figure 10. Error assessment by color coding

2.7. Color

Different methods are possible to simulate the use of color. Color can be predefined by the user or it can be associated with special local properties and created on the fly during isosurfacing. Predefining colors can also be done by simply painting them as 2D images or specifying them with the use of analytical functions.

2.7.1 Procedural defined 3D textures

Procedural defined textures can be evaluated for points in 3D space. The procedural texture function is evaluated for every point used to build the surface representation. We are not doing real texture mapping for all points in the interior of a triangle. By only assigning color values to the vertices and using smooth shading, large polygons become clearly visible for the lack of color change in their interior. Therefore, large polygons (and especially the extremely thin and long (*bad behaved*) polygons of the side surfaces) have to be split into a set of smaller (*well behaved*) polygons. Refer to figure 11 for a graphical representation of bad and well behaved polygons. One simple example for a procedural texture could describe the block of stock material as consisting of two layers of differently colored plastic: Let D_T

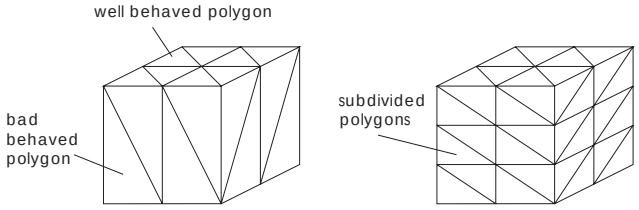

Figure 11. Bad behaved polygons have to be converted into well behave ones

be the texture function for dexel set D with

$$T(x,y,z) = \begin{cases} black & z > \text{half the block height} \\ yellow & \text{otherwise} \end{cases} \quad (4)$$

The color value returned for point (x, y, z) (in dexel coordinates) only depends on the z value. If it is within the top part of the workpiece, the texture function will evaluate to $black$, whereas it will be $yellow$ if the desired point is below half the height of the workpiece dexel set. An image of a workpiece milled with this function is shown in figure 12. More sophisticated procedural texturing functions include the stochastic textures described in [3]. For images rendered with these textures see figure 14 to 16.

2.7.2 Physically based Textures

Another possibility to define a texture for a dexel set is to interpret local geometric properties of the dexels. For

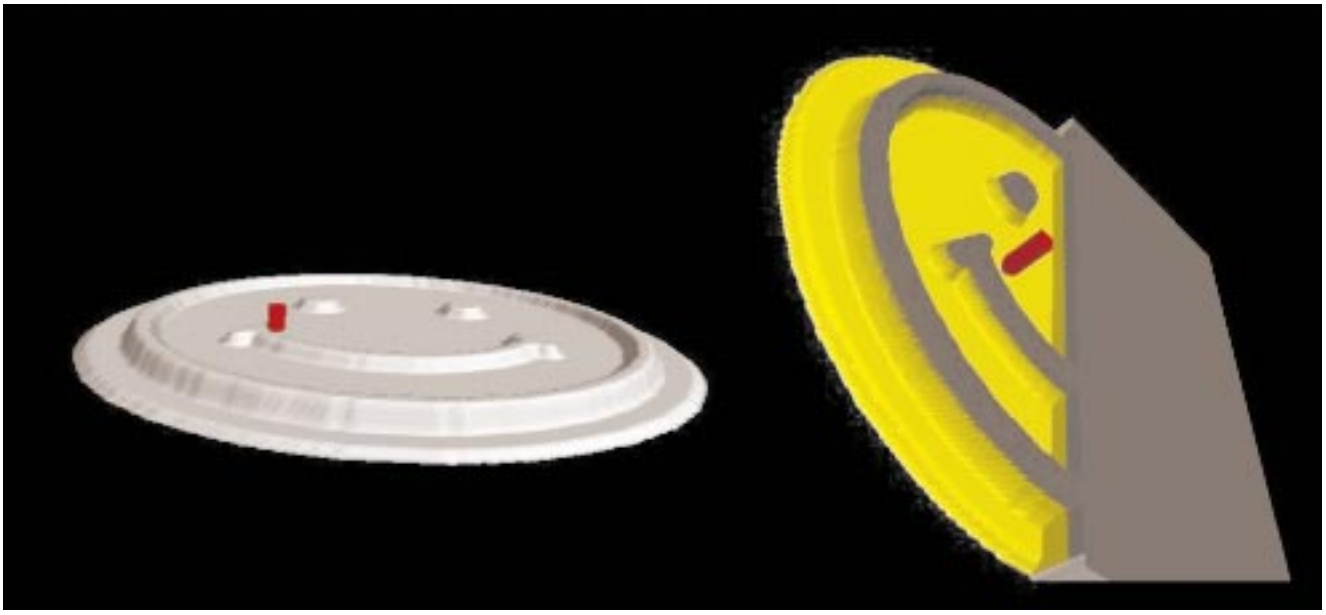

Figure 12. Design part (left) milled from two layered compound material (right)

instance, the local derivative of the dexel height could be mapped into color values. High frequency height changes could be emphasized, therefore accentuating edges. Other possibilities include the highlighting of regions where the dexel set is very thin, e.g., the difference between dexel height and bottom values is small. This knowledge is very important for the NC programmer in order to treat fade-out regions carefully, because they tend to produce cracks.

3. Results

In order to prove the usability of the concepts described in the previous sections, a software prototype was implemented. As the method should be suitable for low-end graphics hardware, an IBM compatible 200 MHz Pentium Pro computer was chosen as the target platform. No 3D graphics accelerator boards were used. The prototype was implemented using Microsoft Visual C++ (ver. 4.2) under Microsoft Windows NT 4.0. The rendering of the polygonial representations of the dexel structures was done with OpenGL. All images presented in the following were created with this prototype implementation.

For high quality color versions of the images presented here, please see our web page at

`http://www.cg.tuwien.ac.at`
`/research/vis/VolVis/NC/nc.html`

The workpiece shown in figure 13 has been modeled with the aid of a scanned photograph of the instrument panel of a Honda Fireblade motorbike.

An analytically defined workpiece (with milling still in progress) is shown in figure 14. It has been milled from a block of stock material consisting of white and blue marble.

Figure 15 shows another geometry milled from two layers of marble.

Figures 16 shows a replication of a tsuba, which is a part of an ancient japanese sword. The workpiece has been milled from bronze.

	instrument panel (figure 13)	smiley (figure 12)	free form surface (figure 14)
resolution	80×40	128×128	256×256
cutter size	9×9	12×12	6×6
cutting operations per second	30	28	32
complete surface extraction	20 sec	1 min 20 sec	2 min 30 sec
local update of surface	100 msec	120 msec	80 msec
instances of motion	6400	16512	65792
simulation time	10 min 40 sec	33 min 4 sec	1 h 27 min 7 sec
cutting only	3 min 30 sec	9 min 42 sec	34 min 16 sec

Table 2. Comparision of processing times for three different workpieces

3.1. Statistics

Table 2 gives a comparison of the processing times for three different workpieces. This test runs were investigated with our prototype implementation on a 200MHz Pentium Pro computer without any hardware graphics acceleration.

The first row of table 2 gives the resolution of the dexel data sets for the workpiece in x and y direction.

The second row describes the resolution of the cutting tool. All workpieces presented here were created by ball shaped cutters.

The third row gives the number of cutting operations, which are possible per second.

The fourth row of table 2 gives the time neccessary for the visualization of the complete dexel data structure of the workpiece. This has usually only to be done once at the beginning of the simulation process. During the simulation, when the cutter moves over the workpiece, only the local part of the scene has to be recreated which is affected by the cutter motion. The time for this local update operation is given in the fifth row. As it can be seen by the small amounts of update time, a continuous real time display of about 10 frames per second is possible. A smooth animation of the moving cutter can be observed.

The sixth row gives the number of instances of motion for the different workpieces. The tool path has been generated automatically. The geometry of the design part was scanned for height in x-axis parallel stripes. When the scanning process reached the edge of the design part, the instruction to lift the cutter was added to the NC programm. The cutter was instructed to return to the other edge without milling, where scanning of the design part was continued. This kind of machining is called *zig-milling* [6]. As in our implementation the instances of motion calculation could not be turned off for the moving cutter, intersection calculations were also made on the parts of the tool path where the cutter just returned to the other edge of the design part without milling. Therefore the number of instances of motion (given in the sixth row of table 2) is in general two times the minimal number of instances neccessary to produce the workpiece.

The last two rows of table 2 show the simulation time for milling the three different workpieces. The row before the last gives the overall simulation time for the different workpieces. For the measurements given in the last row (”cutting only“), the animated visualization of the milling process has been turned off. In this case just the cutting calculations were made without the continuous update of the graphical representation of the milling scenario.

4. Conclusions

A new approach to the simulation of NC milling processes was created. The well known dexel approach uses a high resolution data structure, which requires enormous amounts of processing power. The dexel approach has been adapted to the limited performance capabilities of low-end graphics hardware by using a lower resolution of the data structure. Therefore a special visualization technique had to be developed, which produces renderings of low resolution dexel structures. Even on low end IBM compatible platforms, update rates are interactive.

Another aim of this work was the integration of inhomogeneous material properties. As the dexel approach is a volumetric representation of data, various different data values can be stored and processed along with the geometric information of the workpiece. Color was chosen as an intuitive example of a locally varying property. Using procedurally defined 3D textures, visually realistic results were obtained.

References

[1] Sourin I. A. and Pasko A. A. Function representation for sweeping by a moving solid. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):11–18, 1996.

[2] Jerard R. B., Hussaini S. Z., Drysdale R. L., and Schaudt B. Approximate methods for simulation and verification of numerically controlled machining programs. *The Visual Computer*, 5:329–348, 1989.

[3] Ebert D. *Texturing and Modeling: A Procedural Approach*. AP professional, Cambridge, 1996.

[4] Glaeser G. and Gröller E. Efficient volume generation during the simulation of NC-milling. TR at the Institute of Computer Graphics, University of Technology, Vienna., 1997.

[5] Rvachev V. L. *Methods of Logic Algebra in Mathematical Physics*. Naukova Dumka Publishers, Kiev, 1974.

[6] Held M. A geometry-based investigation of the tool path generation for zigzag pocket machining. *The Visual Computer*, 7:296–308, 1991.

[7] Yang M. and Lee E. NC verfication for wire-EDM using an r-map. *Computer Aided Design*, 28(9):733–740, 1996.

[8] Takafumi S. and Takahashi T. NC machining with the G-buffer method. *Computer Graphics*, 25(4):207–216, 1991.

[9] Chappel I. T. The use of vectors to simulate material removed by numerically controlled milling. *Computer Aided Design*, 15(3):156–158, 1983.

[10] Hook T. van. Real time shaded NC milling display. *Computer Graphics*, 20(4):15–20, 1986.

[11] Lorensen WE. and Cline HE. Marching cubes: a high resolution 3D surface reconstruction algorithm. *Computer Graphics*, 21:163–169, 1987.

[12] Huang Y. and Oliver J. H. NC milling error assessment and tool path correction. *Computer Graphics Proceedings, Conference Proceedings July 24-19, 1994, (Proc. SIGGRAPH '94)*, pages 287–294, 1994.

[13] Kawashima Y., Itoh K., Ishida T., Nonaka S., and Ejiri K. A flexible quantitative method for NC machining verification using a space-division based solid model. *Visual Computer*, 7:149–157, 1991.

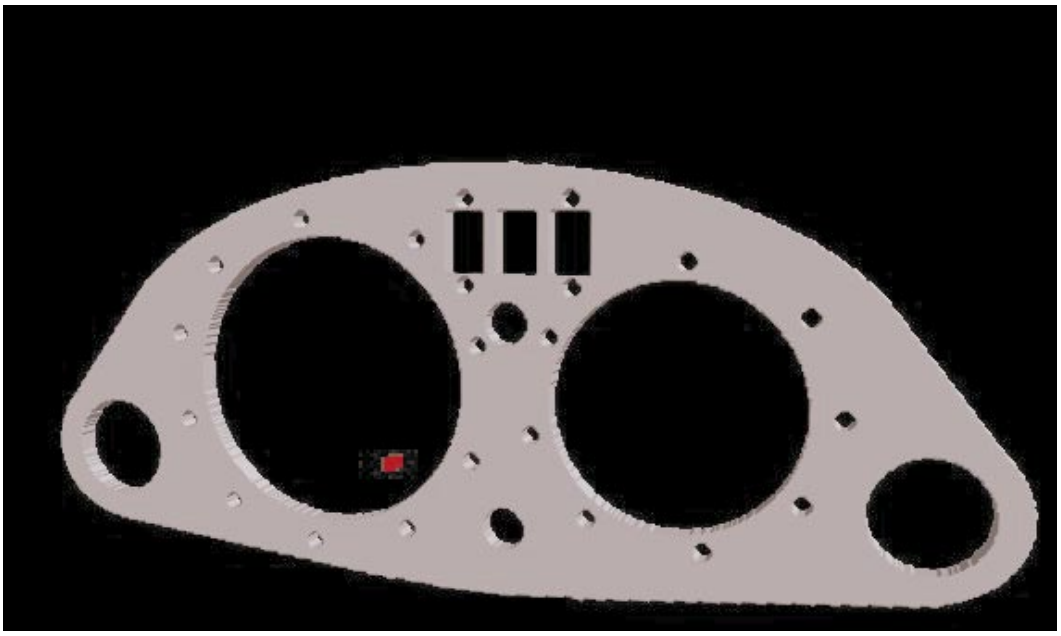

Figure 13. Instrument panel, part of a motorbike

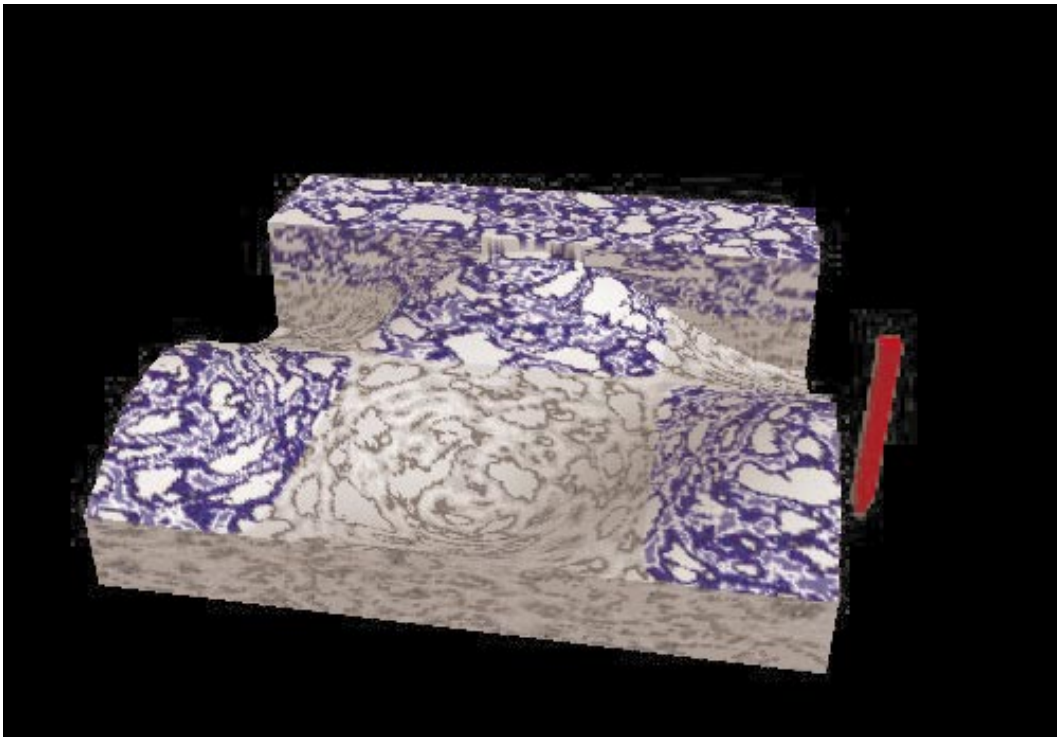

Figure 14. Free form surface milled from marble, cutter in red

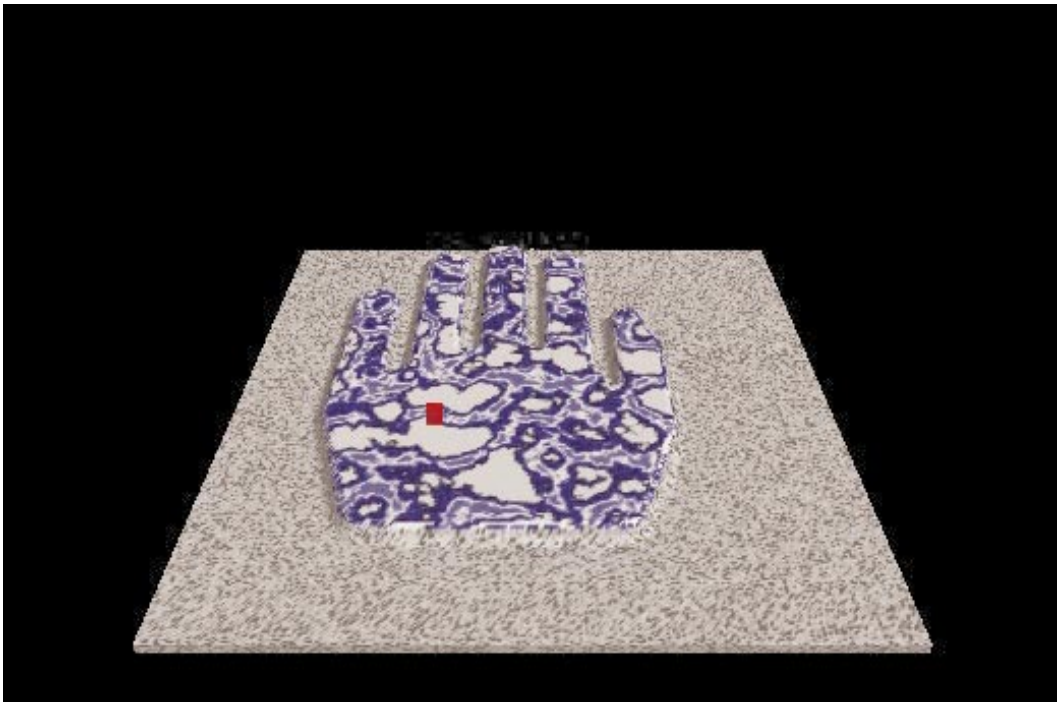

Figure 15. Another example for a workpiece milled from two layers of marble

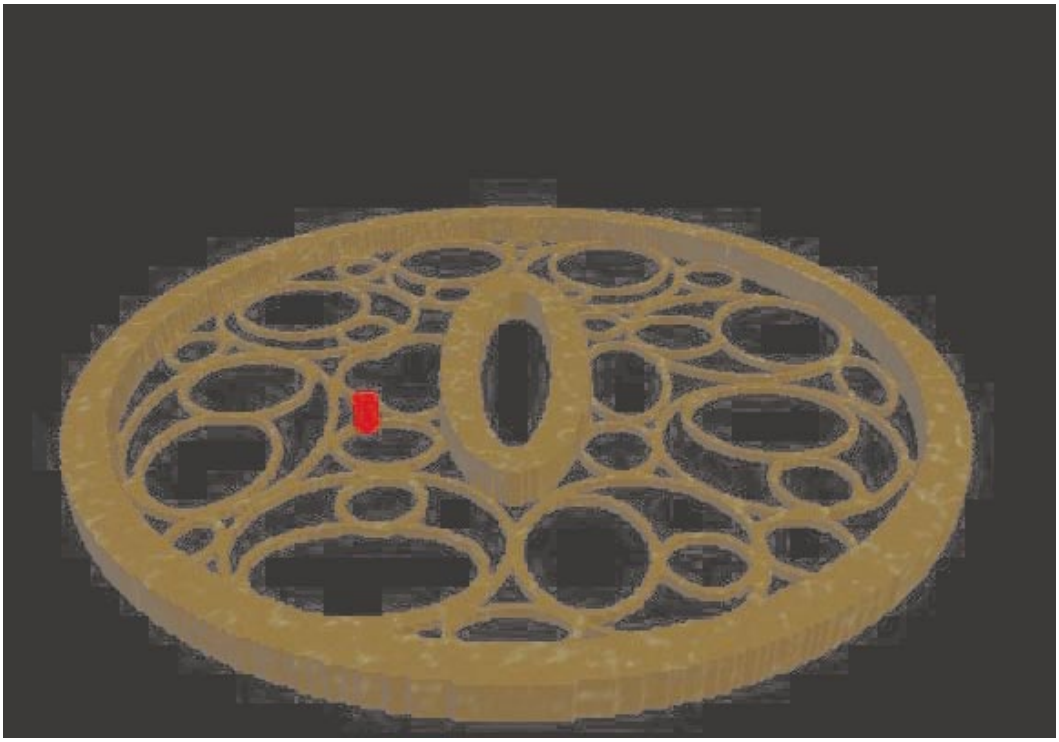

Figure 16. Tsuba milled from bronze