4

Modeling and Visualization of the Surface Resulting from the Milling Process

Tobias Surmann
Institute of Machining Technology, Technische Universität Dortmund
Germany

1. Introduction

In milling of arbitrary free formed surfaces many errors can occur which may lead to inacceptable surface quality because of dynamic effects. One of these dynamic effects are tool vibrations which consist of a static part i. e. a static tool deflection and of a dynamic part. A special case are self excited vibrations called chatter vibrations. These chatter vibrations can grow up to a very high amplitude and thus may lead to high tool wear or even tool breakage. In any case chatter leads to a high surface roughness.

Besides the prediction of whether a milling process is free of chatter or not, the resulting surface quality is of high importance. Though chatter free milling produces a smooth surface structure, the absolute contour error may be high because of high amplitudes of the tool vibration since chatter-free does not necessarily mean vibration-free (Surmann & Biermann, 2008). Every change of feed rate or a change of the feed direction may also lead to surface artifacts. Chatter produces a high roughness and so called chatter marks (Fig. 1) with a negative surface location error in the majority of cases (i. e. too much material is removed).

There has been done a lot of research in the field of chatter theory and milling simulation (Altintas, 2000; Schmitz & Smith, 2008) for non transient milling processes i. e. the tool engagement conditions do not change over time. In general milling processes for free formed surfaces, the engagement conditions change often and rapidly. For that reason, it is hard to control the process parameters in order to optimize the process and the surface quality over an entire NC-program.

A simulation of the dynamic milling process, which calculates the tool vibration trajectory along arbitrary NC-programs, can help to predict the surface quality by converting the trajectory into a geometric model of the workpiece. This model has to provide the possibility of rendering and of measuring the surface error. In this work, the surface model is represented by a discrete set of surface points which are created during a simulation run. In this way, the simulation is able to start at any position within an arbitrary NC-program and does not have to simulate all the way up to the position of interest.

For the prediction of process forces, which are the source of tool vibrations in the milling of free formed surfaces, a simulation technique is required, which provides the modeling of continuous chip shapes from arbitrary NC-paths with the option to feed back tool deflections into the geometric model. This is necessary for the modeling of self excited milling tool vibrations.

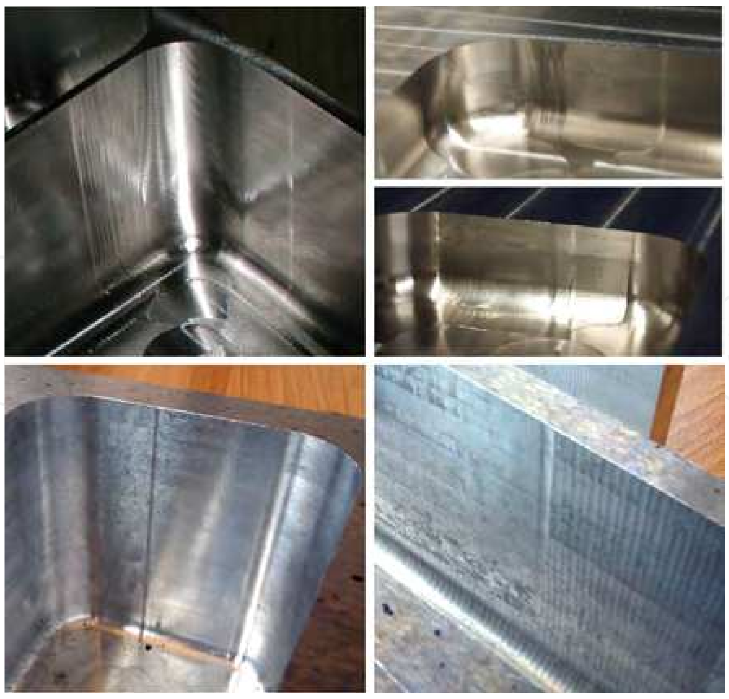

Fig. 1. Typical surface artifacts on milled surfaces. Top left: Marks resulting from changes of feed velocity. Top right: Chatter marks before and in corners. Botton left: Artifact resulting from change of feed direction. Bottom Right: High roughness because of chatter marks.

This article presents a simulation, which is capable of expressing the chip forms as continuous volume models with the ability of reproducing varying chip thicknesses resulting from tool vibrations. The cutting force calculation generates surface points, which are used to generate the surface model. By knowledge of the surface points and the tool vibration trajectory, a realistic surface model can be used to predict surface errors and chatter marks.

2. Modeling of the chip form

As chip form we define the form of the part of the material that is removed while the tool moves along one tooth feed. From this chip form the chip thickness distribution along the cutting edges can be calculated in order to compute the actual cutting force.

2.1 Set theoretical approach

Taking into account that the tool moves along an NC-program in steps of the tool feed f_z, the chip form C_n at the n-th tooth feed can be expressed as a set formula

$$C_n = W_0 \backslash \left(\bigcup_{i=1}^{n-2} T_i \right) \backslash T_{n-1} \cap T_n, \tag{1}$$

where W_0 is the stock material and T_i is the envelope of the rotating tool at the corresponding position of the i-th tooth feed. Hereby, the boolean operators $\cup, \cap$ and $\backslash$ denote the union, the intersection and the difference respectively of the point sets represented by the workpiece and the tool models. Since the cutting velocity is assumed to be much greater than the feed velocity, the equation delivers a perfect description of the removed material at the n-th cut.

The equation can be directly converted into a CSG (Constructive Solid Geometry) tree. At the same time the sets W_0 and T_i are represented by CSG subtrees themselves.

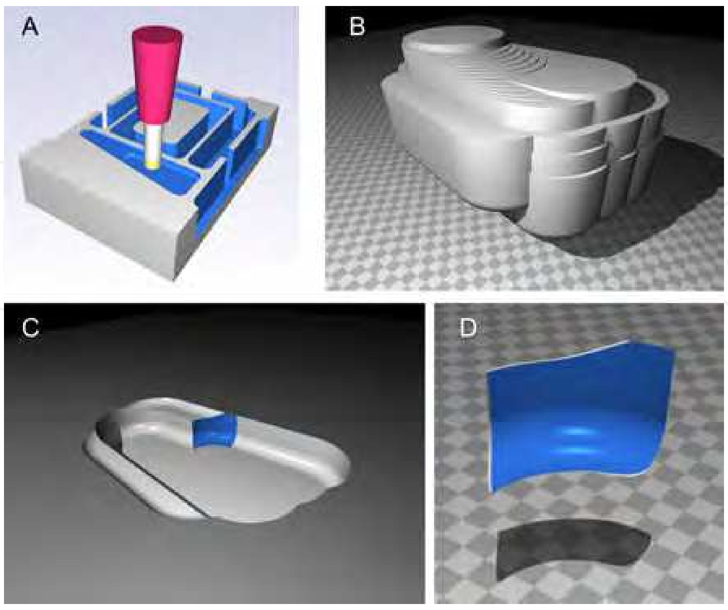

Fig. 2. A local workpiece model (C) at a tool position (A) can be modelled by subtracting the sweep volume (B) of the tool from a stock workpiece. The intersection between this local workpiece and the tool envelope results in the chip shape (D)

2.2 Minimizing the CSG tree

The reason why milling simulations, which are based on CSG, are said to be inefficient is that the size of the set union in the set equation may become very large and the further processing of the chip shape becomes very slow. However, instead of modeling the whole workpiece like Kawashima et al. (1991) did, it is sufficient for the construction of the chip form to model only the local region close to the milling tool at the position of the current cut n (Fig. 2). This is done by taking only those T_i into account, which intersect with the tool model entity T_n. In order to find these necessary T_i efficiently, an octree (Foley et al., 1994) space division data structure is applied to store the NC-program (i. e. the line segments of the given tool path from which the tool positions are taken). So it is possible to exclude most of the unneeded tool path segments with only a few simple intersection tests between box shaped subspaces of the octree and the bounding box of the milling tool model T_n.

3. Force computation

In order to compute the forces acting on the milling tool while removing the chip shape we divide each cutting edge into small linear segments of length b. We assume that each of these segments represents a small cutting wedge for which the mechanics of the oblique cut (Fig. 3) are supposed to be valid and the Kienzle-Equation

$$F_i = b \cdot k_i \cdot h^{1-m_i} \qquad \text{with} \quad i \in \{c, n, t\} \tag{2}$$

applies, where the k_i and m_i are the specific cutting force parameters which can be looked up in tables or have to be determined experimentally.

The force acting on a specific wedge results from its related uncut chip thickness h and the

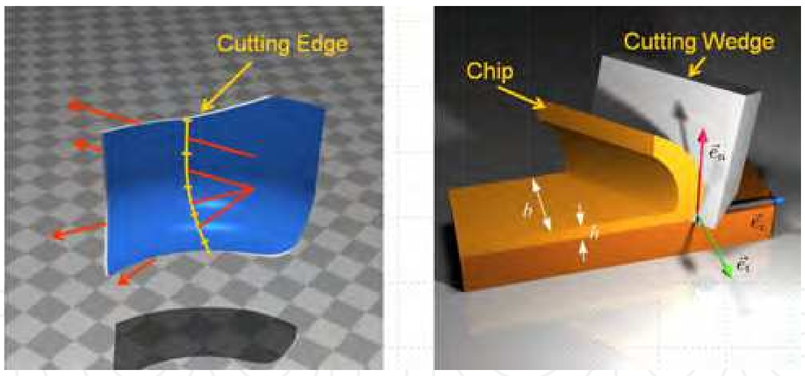

Fig. 3. By casting rays through the chip shape the chip thickness h can be evaluated (left). By the model of the oblique cut (right) the cutting forces can be calculated.

local coordinate system determining the cutting direction $\vec{e}_c$, the normal direction of the cut $\vec{e}_n$ and the tangential direction $\vec{e}_t$.

These vectors are dependent on the position $l, 0 \leq l \leq 1$ on the cutting edge and on the current angular position ϕ of the cutting edge on the tool envelope. Thus, the chip thickness h is a function of l and ϕ as well. The over all force $\vec{F}(l,\phi)$, acting on a cutting edge segment at position l,ϕ can be expressed as

$$\vec{F}(l,b,\phi) = F_c(b,h(l,\phi)) \cdot \vec{e}_c(l,\phi) + F_n(b,h(l,\phi)) \cdot \vec{e}_n(l,\phi) + F_t(b,h(l,\phi)) \cdot \vec{e}_t(l,\phi). \tag{3}$$

The chip thickness $h(l,\phi)$ can now be computed by intersecting a ray

$$\vec{x} = \vec{p}(l,\phi) + t \cdot \vec{e}_n(l,\phi) \tag{4}$$

with the CSG-model of the chip form. Hereby, $p(l,\phi)$ is a point on the cutting edge at the revolution angle ϕ and the position l on the paramtric curve representing the cutting edge. The distance between entry and exit point is the chip thickness h.

Let φ be the current angle of revolution of the tool, n the number of cutting edge segments and $\Delta\phi$ the pitch angle between the cutting edges. The Force acting on then j-th cutting edge is then

$$\vec{F}(j,\varphi) = \sum_{i=0}^{n-1} \vec{F}\left(\frac{i}{n}, b, j \cdot \Delta\phi + \varphi\right). \tag{5}$$

Now we have to sum up the forces over all z cutting edges in order to get the force acting on the entire tool at rotation angle φ

$$\vec{F}(\varphi) = \sum_{i=0}^{z-1} \vec{F}(i,\varphi). \tag{6}$$

In order to model the cutting forces along a tool path we model the chip forms in steps of the tooth feed f_z. Furthermore, each partial revolution along a tooth feed is devided into j angle steps $\Delta\varphi$ with

$$\Delta\varphi = \frac{2\pi}{zj}. \tag{7}$$

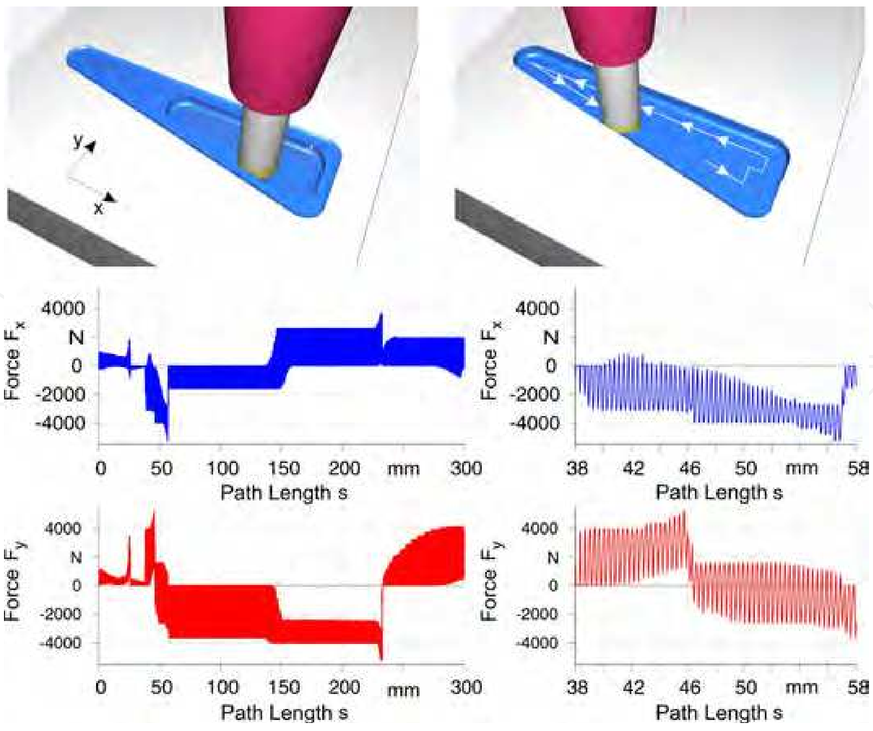

Fig. 4. Milling forces along a section of an NC-program.

From the spindle speed n we get a time step Δt of

$$\Delta t = \frac{j}{zn}. \tag{8}$$

At the beginning of a simulation run the current rotation angle φ and the time t is set to 0 and the first chip form is constructed. For this chip form the milling force $\vec{F}(\varphi)$ is computed and saved as $\vec{F}(t)$. As a next step as well φ as t are increased by $\Delta\varphi$ and Δt respectively and the milling force is computed again. After the force computation at the angle

$$\varphi = \frac{2\pi}{z} - \Delta\varphi \tag{9}$$

φ is reset to 0 again and the next chip form is modelled and processed in the same way. Fig. 4 shows the force progression along a section of a tool path producing a pocket.

4. Machine model

Since real machines cannot be assumed to be ideal stiff, they will respond to an acting force not only with a static deflection but also with a vibration. In order to model this response we use a set of harmonic oscillators for each coordinate direction. Each oscillator consists of three modal parameters: natural angular velocity ω_0, modal mass m and damping constant γ. With these values we can compute the response on applied impulses on the tool tip.

4.1 Response of a harmonic oscillator to an impulse

At first we deal with only one oscillator. We asume that the oscillator has at time t_0 the velocity v_0 and the displacement x_0. Further we assume that a force F_0 acts for a period of Δt. The new values x_1 and v_1 after Δt can be computed by

$$x_1 = A \cdot e^{(iw-\gamma)\Delta t} + B \cdot e^{(-iw-\gamma)\Delta t} \tag{10}$$

$$v_1 = A \cdot (-\gamma + i\omega) \cdot e^{(iw-\gamma)\Delta t} + B \cdot (-\gamma - i\omega) \cdot e^{(-iw-\gamma)\Delta t}. \tag{11}$$

Here, A and B are defined as

$$A = \frac{1}{2i\omega} \cdot \left(v_0 + \frac{\Delta t}{m} \cdot F_0 + (\gamma + i\omega) \cdot x_0 \right) \tag{12}$$

and

$$B = \frac{1}{2i\omega} \cdot \left(-v_0 - \frac{\Delta t}{m} \cdot F_0 + (-\gamma + i\omega) \cdot x_0 \right) \tag{13}$$

respectively. Generally, starting at a displacement x_j with the velocity v_j we can now compute the related values x_{j+1}, v_{j+1} of the next time step after application of a force F_j by a linear combination of the current values and the actual force

$$x_{j+1} = a \cdot x_j + b \cdot v_j + c \cdot F_j \tag{14}$$

$$v_{1+1} = d \cdot x_j + e \cdot v_j + f \cdot F_j. \tag{15}$$

The constants $a \dots f$ can be precalculated from the above equations and are valid as long as the time step Δt does not change.

4.2 MDOF oscillator model

In order to model the response of a milling machine with respect to the tool tip we use a set of uncoupled oscillators for each coordinate direction. A force acting on the tool tip is then applied to each oscillator and the responses are superposed. Let m be the number of oscillators used for one direction and let $x_{k,j}$ and $v_{k,j}$ be the displacement and the velocity respectively of the k-th oscillator at time step j, then the overall response $\hat{x}_{j+1}$ of the machine tool at time step $j+1$ reads as

$$\hat{x}_{j+1} = \sum_{k=0}^{m-1} x_{k,j+1}. \tag{16}$$

The modal parameters modal mass, natural frequency and damping constant of the single oscillators have to be adjusted to the measured frequency responce functions (FRF) of the

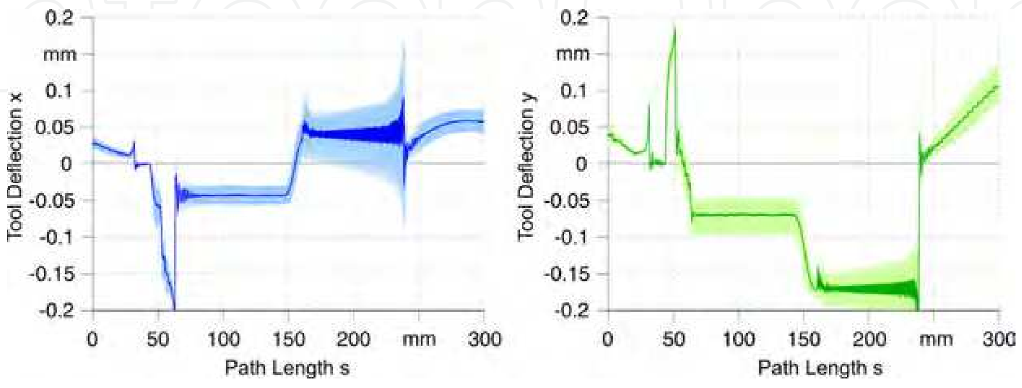

Fig. 5. Milling tool vibrations along a section of an NC-program (dark colors denote the displacement at the point in time of surface generation).

actual machine tool in x- and y-direction in advance. For this, an evolutionary algorithm is applied which optimizes the modal parameters in a way that the FRF of the superposed oscillators fit the measured FRF.

5. Modeling the regenerative effect

As seen in Fig. 4, the cutting forces are not constant and thus they excite vibrations of the milling tool. Vibrating tools produce a wavy surface which in return produces a modulation of the milling force. This feed back is called regenerative or self-excited vibrations. The geometric model of the chip form can be easily used for simulating this regenerative effect.

The equations 4 to 6 describe the force computation for one tooth feed in discrete rotation angle steps $\Delta\varphi$, where each angle φ is assigned a specific point in time t. Thus, instead of $\vec{F}(\varphi)$ we can write $\vec{F}(t)$. The two tool models T_{n-1} and T_n of Eq. 1 generate the surfaces of the last cut and the current surface which are responsible for the regenerative effect.

Let $\vec{x}(t)$ and $\vec{x}(t-T)$ be the tool displacements at time t and time $t-T$ respectively. We now move the tool models according to those two displacements just before the evaluation of the chip thickness along the cutting edges. After computation of the Force $F(t)$ and the computation of the resulting displacement $\vec{x}(t+\Delta t)$, the displacements are reset, the tool is rotated by $\Delta\varphi$ and the procedure is repeated with the current rotation angle and the displacements $\vec{x}(t+\Delta t)$ and $\vec{x}(t+\Delta t-T)$.

Fig. 5 shows the simulated tool vibration along the same tool path as Fig. 4.

6. Point-based workpiece model

In contrast to the work of Gao et al. (2006); Li & Shin (2006); Omar et al. (2007), where the surfaces are modelled as a high resolved continous triangulation, in this work a simpler way is presented which generates a point-based surface model which is capable of rendering the surface of complex parts generated by NC-milling processes.

In NC-milling of geometrically complex parts problems or errors resulting in artifacts on the surface cannot be recognized until the process is finished. Especially for the industrial application of a simulation system it is important that the simulation provides a surface model which is capable of displaying these surface errors. Thus, this section will deal with the geometric modeling and visualization of the resulting surface structure as a point-based surface model.

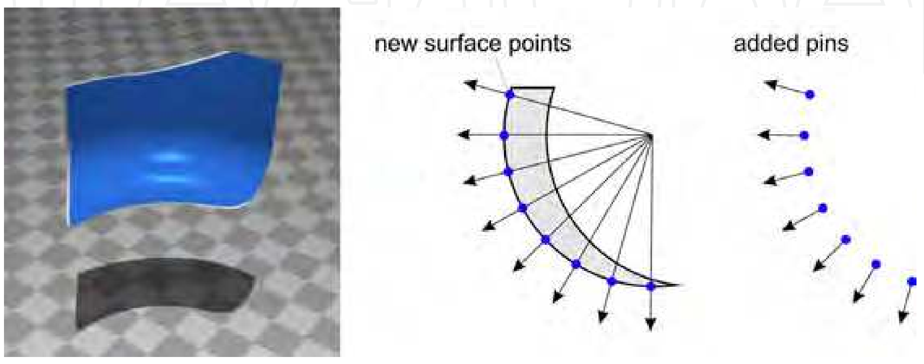

Fig. 6. The exit points of the rays used to compute the chip thicknesses are applied to form a surface model consisting of pins.

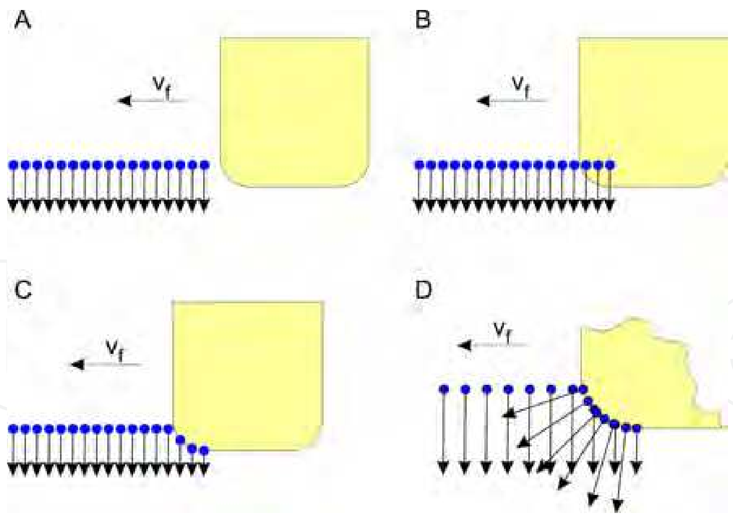

Fig. 7. The material removal from the pin model (A) is conducted by cutting the pins which intersect with the tool envelope (B) while the tool moves through the workpiece. The points are moved to the intersection points (C) and finally the new pins are added (D).

6.1 Adding points

Since the presented milling simulation system does not make use of a global workpiece model but of a locally modelled chip form it is capable of starting at any point within an NC-program. Thus, for visualization of the surface we need a model that produces the newly generated surface of a section of the tool path. Therefore, we make use of the fact that the backside of a chip form (Fig. 6) for a short time represents a small part of the surface, i. e. the newly generated surface of the current cut. When computing the chip thickness by casting rays through the chip form the exit points of the rays represent surface points. At each cut these points together with a normal vector of length d are added to a set of points representing the workpiece surface. The couple of point and normal vector is called *pin* because of its appearance. Each tooth feed produces a set of new pins (Fig. 6).

6.2 Removing material

We suppose there is already a set of pins (Fig. 7 A), through which the tool model, i. e. the envelope of the rotating tool, is moving. Points, which come to lie inside the tool (Fig. 7 B) are moved along their normal vector upon the tool envelope. The tie points stay the same, i. e. the pin becomes shortened (Fig. 7 C). Pins which entirely lie inside the tool envelope are deleted.

By successive cutting, removing and adding pins a model of the newly machined surface grows. Fig. 8 displays the result by a 3D-Example from the milling simulation system.

6.3 Estimation of the tool deflection before cutting

In the previous section we assumed the tool model to move exactly along the given tool path. This enables a visualization of the ideal suface but does not produce surface errors as the actual process does. For a material removal featuring surface errors we determine the tool displacement at each cut at the point in time where the remaining surface is produced. This displacement is applied to the tool additionally before conducting the cut of the pin model.

Again, we use the set of newly generated pins of a tooth feed which is displayed in Fig. 9 by means of the chip form of Fig. 2. Since most of the pins are surface points only for a short

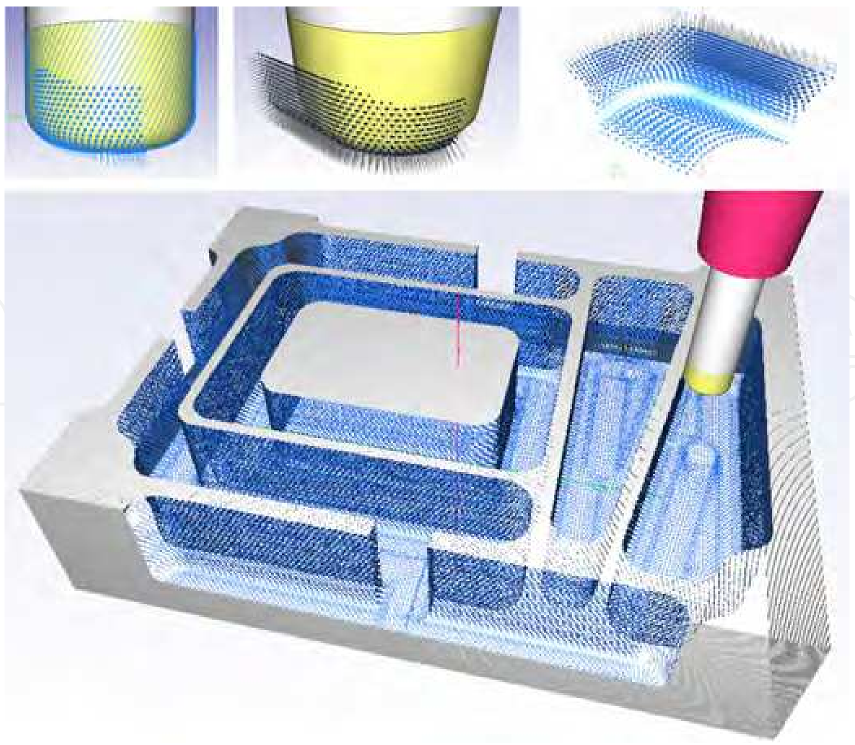
Fig. 8. Example of a pin model generated by an NC-program. Top row: points of the chip form (left) and generated surface part (middle and right)

period of time and will be removed again by the following cut, we have to find out those points which will contribute to the final surface. For these points P_i applies that their normal vectors $\vec{n}_i$ are perpendicular to the feed direction $\vec{v}_f$:

$$\vec{n}_i \cdot \vec{v}_f = 0 \Rightarrow P_i \text{ is finally surface point.} \tag{17}$$

Unfortunately, the tool rotaion is divided into discrete angular steps $\Delta\varphi$, so that the criterion has to be understated like

$$\frac{\vec{n}_i \cdot \vec{v}_f}{|\vec{n}_i| \cdot |\vec{v}_f|} \leq \cos(\Delta\varphi) \Rightarrow P_i \text{ is nearly surface point.} \tag{18}$$

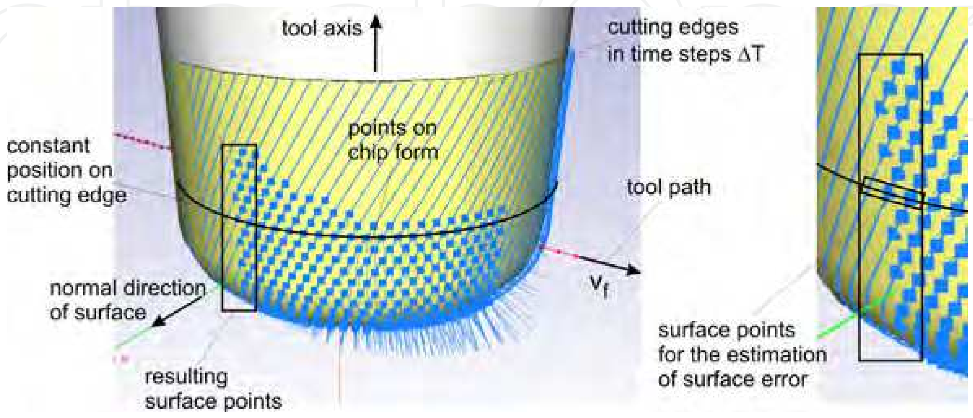

Fig. 9. The points on the chip form whose normal is nearly perpendicular to the feed direction v_f are used to determine the tool deflection at the moment of surface generation.

We can test each point for this criterion directly on its creation. In the positive case the point is added to a set S carrying all the points that form the remainig part of the surface (marked in Fig. 9). Additionally, each point $P_i \in S$ is assigned with the displacement $\vec{d_i}$ the tool had when the specific point was created.

In order to estimate the tool displacement for cutting the pins we use the average tool displacement $\vec{d}_{avg}$ of the points from a subset of S lying on a specific height. This height can be the middle of the total axial immersion (Fig. 9, right). Before cutting the pins the tool model is translated by $\vec{d}_{avg}$ from its ideal position. The result of many succeeding cuts is a point-based surface model including the surface errors resulting from dynamic effects.

7. Point based rendering

Processed surfaces like generated by digitizing often are available as point sets without any relationship between the points. A triangulation often is too time consuming or error-prone if the shape of the object is geometrically complex. Especially when the surface is

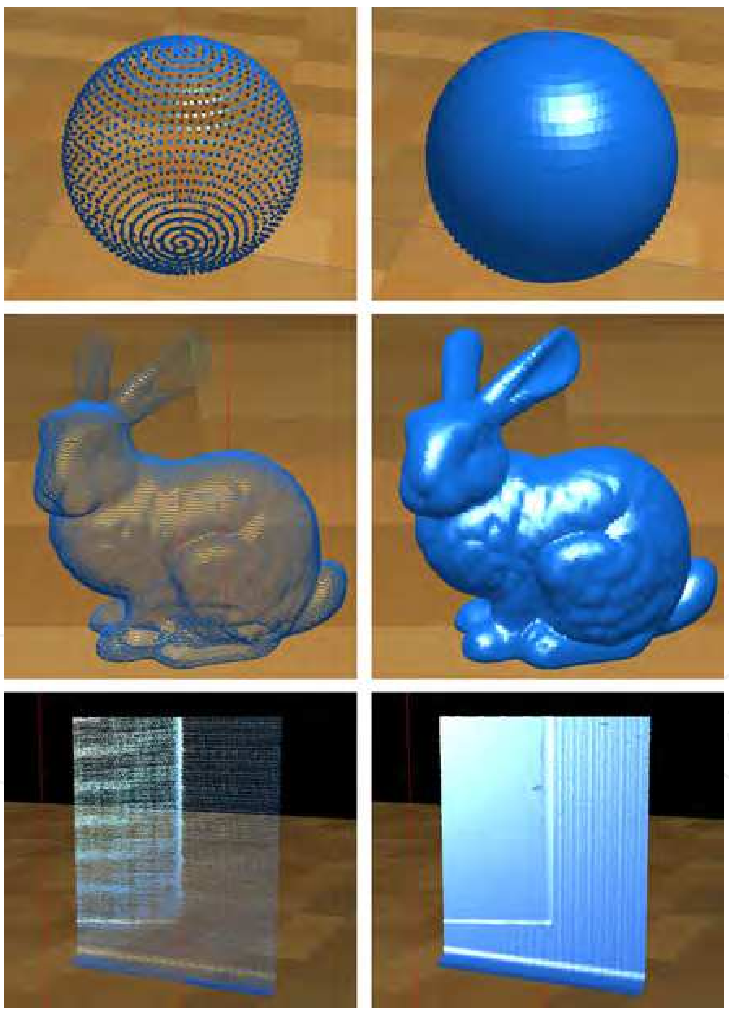

Fig. 10. Splat-rendered objects (Sphere, Stanford Bunny and milled surface containing chatter marks) with low splat size (left) and large splats to make the impression of a closed surface.

continually modified like in the milling simulation a very fast and efficient rendering method is demanded.

Taking a look at Fig. 8 reveals, that a pure rendering of the points together with a shading by usage of normal vectors already provides a good visulaization of the surface. The next sections deal with two direct point-based rendering techniques.

7.1 Splat rendering

In case the points carry normal vectors, like they do in the pin model, it is possible to make use of the graphics hardware and just render the points (Foley et al., 1994; Wright et al., 2010) on the screen. Hereby, the points are shaded by using their normal vectors and are rendered as big splats (Akenine-Möller & Haines, 2002). From a specific splat size the surface seems to be a closed surface (Fig. 10). Using this technique the points become two-dimensional not before putting them on the screen whereby the simplest method is to draw squares with the edge length of a. The splatsize $a(d)$ can additionally adapted to the distance d to the viewer with a certain constant k:

$$a(d) = \frac{k}{d}. \tag{19}$$

The constant k determines the splat size and has to be adjusted dependently on the density of the points, that the surface seems to be closed at all distances (Fig. 10).

This method is useful when there is no demand for very high visualization quality but for visualization eficiancy and speed. However, surface errors can be recognized directly while the simulation is still in progress.

7.2 Point based raytracing

For a photorealistic rendering of simulated surface models the ray tracing technique (Akenine-Möller & Haines, 2002; Foley et al., 1994; Shirley, 2005) is much more suitable. Of high importance for a ray tracing system is that the objects provide a possibility to compute intersections with a ray. Therefore, the next section is about an algorithm for the direct computation of intersections between a ray and a point-set based on the method presented by Linsen et al. (2007) and Goradia et al. (2010).

The points within a point-set surface can be considered as representatives for a closed surface. Together with a normal vector and other properties like color and material these

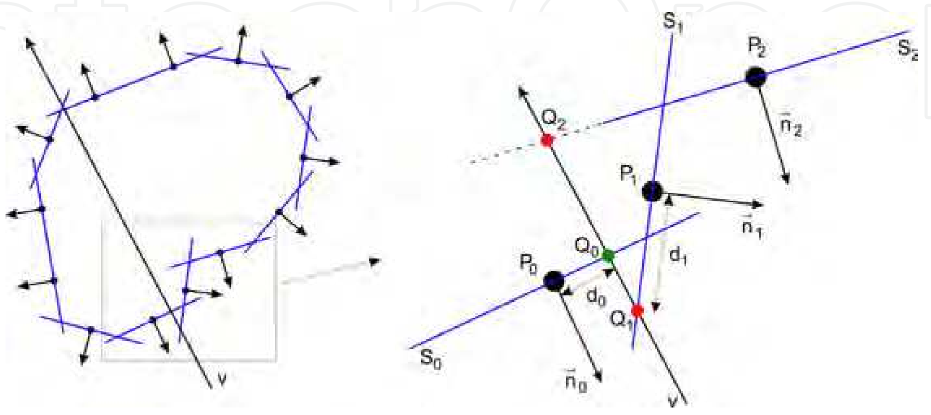

Fig. 11. Principle of ray intersection with surfels. The Point Q_0 has the minimal distance to its related surface point P_0 and thus is selected as the intersection result.

representatives are called *surfel* (surface element). These surfels are a circular disc with radius r and with the related point in the center. Additionally, it is alligned perpendicular to the related normal vector.

7.2.1 Intersection of ray and surfel

A surfel S describes a plane f by its point P and its normal vector $\vec{n}$ by

$$f: \qquad (\vec{x} - \vec{P}) \cdot \vec{n} = 0. \tag{20}$$

The intersection point Q with the view ray v starting at the camera position B with the viewing direction $\vec{v}$

$$v: \qquad \vec{x} = \vec{B} + t \cdot \vec{v} \tag{21}$$

results from inserting Eq. 21 into Eq. 20, and by computing the factor t

$$t = \frac{\vec{P} \cdot \vec{n} - \vec{B} \cdot \vec{n}}{\vec{v} \cdot \vec{n}}. \tag{22}$$

Using the factor t with Eq. 21 we get a point

$$\vec{Q} = \vec{B} + t \cdot \vec{v}. \tag{23}$$

This point is intersection point with the surfel if

$$d = |\vec{Q} - \vec{P}| \leq r \tag{24}$$

applies.

7.2.2 Multiple Intersections

A set of surfels representing a closed surface generally leads to intersections of a ray v with more than one surfel (Fig. 11). At first we ignore the surfels whose normals point into the same direction as the view ray v and thus

$$\vec{n}_i \cdot \vec{v} \geq 0 \tag{25}$$

applies. Thus, in Fig. 11 only Q_0 and Q_1 remain as possible intersections. A simple and effective method to decide which intersection to use is to take that point Q_i whose distance d_i from the related surfel center is minimal. In Fig. 11 this is point Q_0.

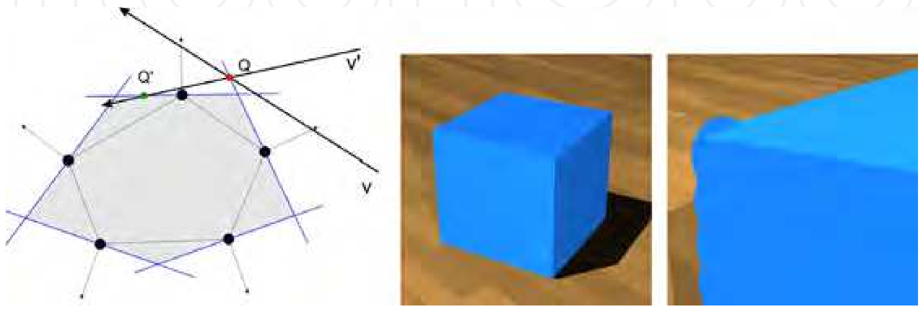

Fig. 12. Artifacts occur, if the object has sharp edges.

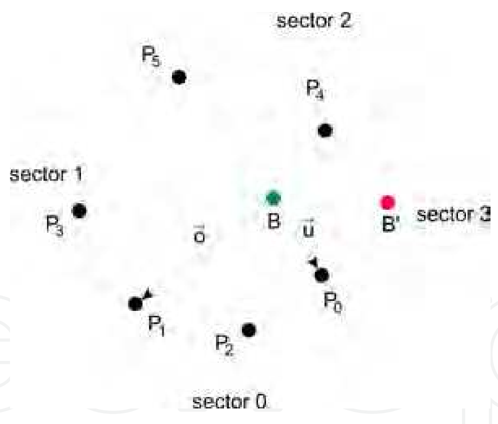

Fig. 13. First step in testing whether a point B is inside the convex cover of a 2D point set.

7.2.3 Point in convex cover

Fig. 12 shows on the left hand side a surface represented by a triangulation of the points. The presented intersection algorithm with the shown ray v would result in the point Q which definitly lies far outside the triangulation. The ray v' which also passes through Q would intersect the surfels in point Q'. This algorthm is obviously view dependent.

Therefore, the silouette of a point set surface with an inhomogenious density would have a frayed silhouette or undesired shadow castings especially when using large surfel radii (Fig. 12). To overcome this issue we improve the intersection algorithm in a way that only those rays can lead to an intersection, which intersect the convex cover of the point set.

Therefore, at first, we only take the projection of the points $P_0 \cdots P_n$ of the point set surface as well as the camera position B onto the view plane into account. At the beginning of the algorithm we compute the vectors $\vec{u} = \vec{P}_0 - \vec{B}$ and $\vec{o} = \vec{P}_1 - \vec{B}$.

The vectors $\vec{u}$ and $\vec{o}$ span together with the point B four sectors (Fig. 13). Now we test each

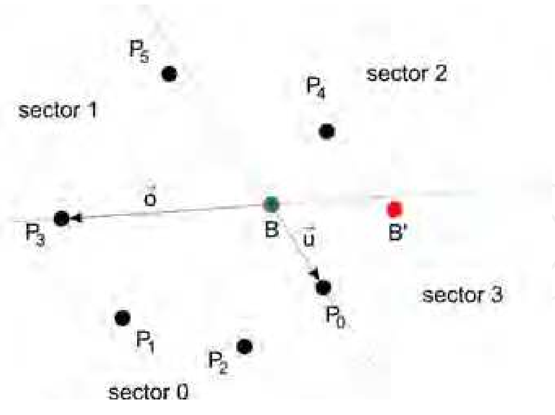

Fig. 14. Second step in testing whether a point B is inside the convex cover of a 2D point set.

point P_i, $i > 1$ in which sector it lies:

Sector 0 Nothing happens and we continue with tne next point.

Sector 1 Vector $\vec{o}$ is replaced by $\vec{P}_i - \vec{B}$ and the next point is tested.

Sector 3 Vector $\vec{u}$ is replaced by $\vec{P}_i - \vec{B}$ and the next point is tested.

Sector 2 The point B lies inside the convex cover and the algorithm terminates.

If all the points P_i have been tested and no point fell into sector 2, B does not lie inside the convex cover and the ray does not hit the point-set surface. In the example of Fig. 13 with point P_2 nothing would happen. After testing P_3 we have the situation shown in Fig. 14. P_4 lies in sector 2 when tested and terminates the algorithm. Thus, point B lies inside the convex cover.

Using B' for computing the vectors $\vec{u}$ and $\vec{o}$ no points P_i come to lie inside sector 2 and thus B' would be recognized as not lying in the convex cover.

By performing this test before computing the intersection with the surfels, the silhouette of point-set surfaces is rendered very sharp (Fig. 15).

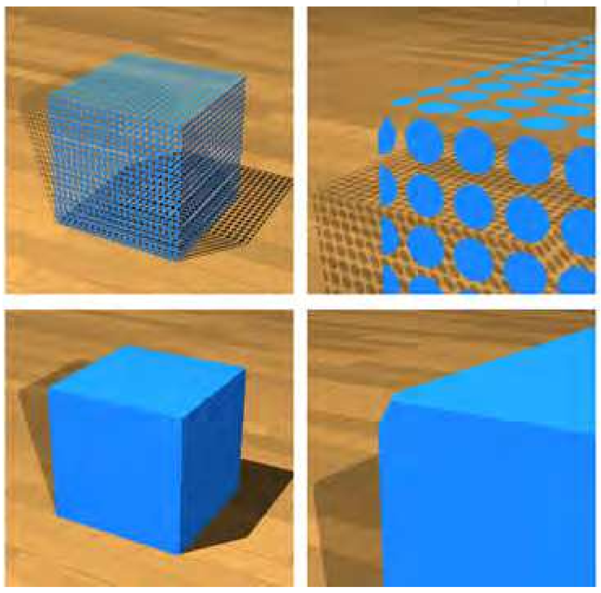

Fig. 15. Using the convex cover criterion, the edges appear sharp and without artifacts. However, corners may be cut because the silhouette of objects now appears as if it was triangulated.

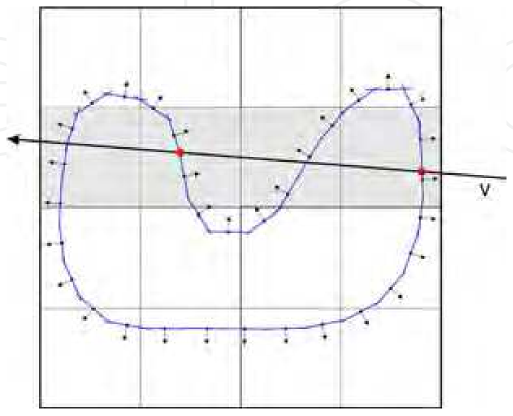

Fig. 16. By partitioning of the points into patches even intersections between a ray and concave objects can be computed.

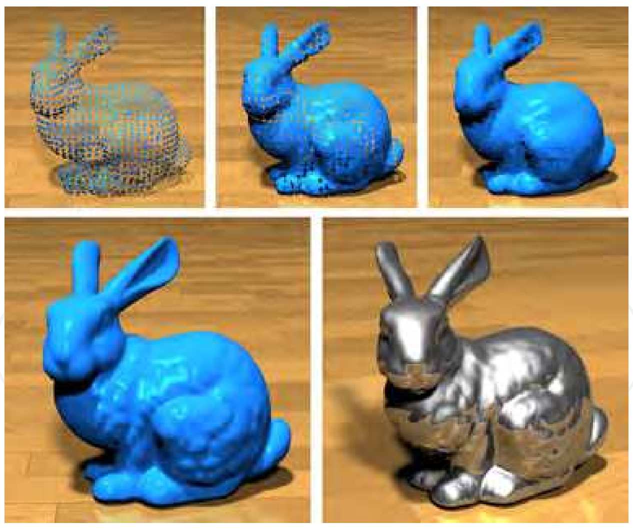

Fig. 17. Stanford Bunnies with a low resolution point-set and different surfel radii (top). High resolution Bunny with different materials (bottom) (source: Stanford University Computer Graphics Laboratory).

7.2.4 Partitioning into patches

In order to render concave objects the point-set surface has to be partitioned into small patches with a size similar to the smallest features of the object. (Fig. 16). Hereby it is possible to relate multiple intersections to different parts of the surface. For higher efficiency of the intersection algorithm the patches should be organized in a space partitioning datastructure like an octree.

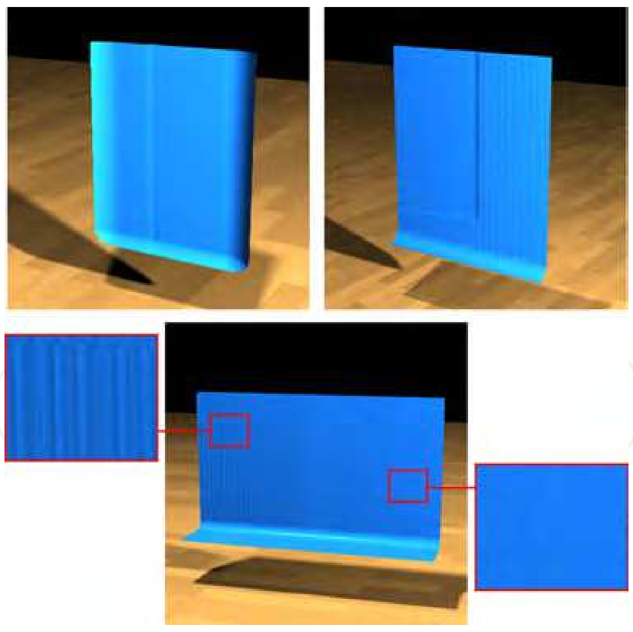

Fig. 18. Even milled surfaces can be rendered photorealistically. The surfaces on the top are related to the two bottom surfaces in Fig. 1. The bottom surface shows the result of a transient process with stable milling (right) and chatter (left).

Fig. 17 shows results for the Standford Bunny and Fig. 18 for some surfaces generated by the milling simulation, where surface structures are clearly visible (cf. Fig. 1).

8. Conclusion

By usage of concepts from the field of computer graphics and geometric modeling it is possible to simulate geometrically complex processes such as the milling process. We have shown that a CSG approach is applicable for usage in milling simulations. Together with an physics based oscillator model even regenerative chatter vibrations can be simulated.

The direct surfel-based raytracing has been improved in order to render objects with sharp edges. This enables the application of the direct surfel-based raytracing for technical surfaces such as milled workpieses.

9. Acknowledgement

This article features developments of the collaborative research center SFB/TR10 which is kindly supported by the German Research Foundation (DFG).

10. References

Akenine-Möller, T. & Haines, E. (2002). *real-Time Rendering*, AK Peters, Natick, Massachusetts.

Altintas, Y. (2000). *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design*, Cambridge University Press.

Foley, J. D., v. Dam, A., Feiner, S. K. & Hughes, J. F. (1994). *Grundlagen der Computergraphik*, Vol. 1 of *1*, 1 edn, Addison-Wesley Publishing Company, Inc.

Gao, T., Zhang, W., Qiu, K. & Wan, M. (2006). Numerical simulation of machined surface topography and roughness in milling process, *Journal of Manufacturing Science and Engineering* 128(1): 96–103.

Goradia, R., Kashyap, S., Chaudhuri, P. & Chandran, S. (2010). Gpu-based ray tracing of splats, *18th Pacific Conference on Computer Graphics and Applications (PG)*, pp. 101–108.

Kawashima, Y., Itoh, K., Ishida, T., Nonaka, S. & Ejiri, K. (1991). A flexible quantitative method for nc machining verification using a space-division based solid model, *The Visual Computer* pp. 149–157.

Li, H. & Shin, Y. C. (2006). A comprehensive dynamic end milling simulation model, *Journal of Manufacturing Science and Engineering* 128(1): 86–95.

Linsen, L., Müller, K. & Rosenthal, P. (2007). Splat-based ray tracing of point clouds, *Journal of WSCG* 15: 51–58. rosenthallinsenvcgl.

Omar, O., El-Wardany, T., Ng, E. & Elbestawi, M. (2007). An improved cutting force and surface topography prediction model in end milling, *International Journal of Machine Tools and Manufacture* 47(7-8): 1263 – 1275.

Schmitz, T. L. & Smith, K. S. (2008). *Machining Dynamics*, Springer, New York. ISBN 978-0-387-09645-2.

Shirley, P. (2005). *Fundamentals of Computer Graphics*, 2 edn, Peters, Wellesley. ISBN-13: 978-1568812694.

Surmann, T. & Biermann, D. (2008). The effect of tool vibrations on the flank surface created by peripheral milling, *CIRP Annals - Manufacturing Technology* 57(1): 375 – 378.

Wright, R. S., Haemel, N., Sellers, G. & Lipchak, B. (2010). *OpenGL SuperBible: Comprehensive Tutorial and Reference*, 5 edn, Addison-Wesley Longman, Amsterdam. ISBN-13: 978-0321712615.