Title:
Fast Computation of Accessibility Cones for Assisting Process Planning of 3+2 Axis Milling

Authors:
Masatomo Inui, masatomo.inui.az@vc.ibaraki.ac.jp, Ibaraki University
Shinji Nagano, 17nm934r@vc.ibaraki.ac.jp, Ibaraki University
Nobuyuki Umezu, nobuyuki.umezu.cs@vc.ibaraki.ac.jp, Ibaraki University

Keywords:
Offsetting, Depth Buffer, Hierarchical Bounding Boxes, GPU

DOI: 10.14733/cadconfP.2017.165-169

Introduction:
In the automobile industry, molds with very deep shape are often used for producing large plastic parts, such as instrument panels. In the usual 3-axis milling, cutters with long shank is necessary to avoid collisions between the holder and the mold in the semi-finishing. Since large deformation of the cutter is unavoidable with a long shank, it is difficult to realize the precise and stable machining. To solve this problem, many manufacturers in Japan use 3+2 axis milling for the semi-finishing. In this method, the machine executes a 3-axis milling program with a cutter locked in a tilted position using its 2 rotational axes. In the 3+2 axis milling, shank length can be reduced by properly selecting the cutter posture as shown in Fig. 1. Similar advantage can be expected by using a simultaneous 5-axis milling machine, however the cutter in a fixed posture is more rigid so much accurate machining results can be realized.

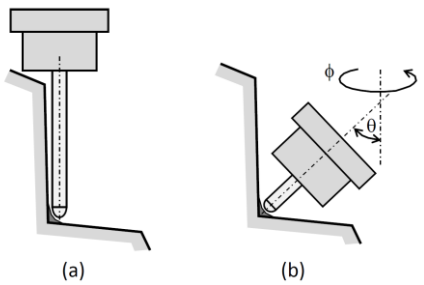

Fig. 1: 3-axis milling (a) and 3+2 axis milling (b) for removing material in a corner.

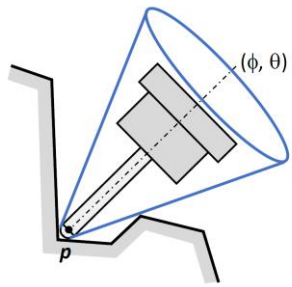

Fig. 2: An accessible cone for a cutter at a specific positon ***p*** and posture (ϕ, θ).

In the 3+2 axis milling, determination of the proper cutter posture is a critical task. In the following discussion, milling operation with a ball end cutter of radius r is assumed. Position ***p*** of the ball end cutter is represented by the center point of the spherical blade of the cutter. We use accessibility cone [2] as a measure for evaluating the appropriateness of the cutter posture at ***p***. For each cutter posture (ϕ, θ) specified by 2 rotational angles ϕ and θ around 2 mutually perpendicular axes of the milling machine, a cone whose axis is coaxial to the center axis of the cutter and smoothly contacting the spherical part of the cutter is considered. The peak angle of the cone is enlarged until the cone touches the mold part surface (see Fig. 2). Such cone with a contacting peak angle is called accessibility cone (AC) of the cutter at ***p*** in posture (ϕ, θ). AC represents the angular clearance between the mold and the cutter at a specific configuration (positon and posture).

Takeuchi et al. proposed a trial-and-error-based method for computing collision free cutter postures in the 5-axis milling [5]. Morishige et al. developed a method for determining collision free

cutting postures in the 5-axis milling using a C-space of the cutter posture [3]. Determination of the collision-free cutter posture is related to the accessibility problem of a point to a certain region on the offset surface of the workpiece. The "visibility cone" is defined as the feasibility range of the tool axis for a surface point. Tseng and Joshi [6] used the visibility cone to determine the accessibility of a cutter to a point. Spitz and Requicha developed a computation method of a visibility cone for the coordinate measurement machine using the perspective projection [4]. Morimoto and Inui extended their method for determining the cutter accessibility in the 3+2 axis milling [2].

Main Idea:

Input and output

In this paper, we propose a novel algorithm for automatically computing all appropriate cutter postures for the 3+2 axis milling. The input data consists of a polyhedral STL model that approximates the mold shape, radius r of a ball end cutter for milling, and a set of points representing the cutter positions in the milling operation. We assume that the cutter posture (ϕ, θ) can be fixed in every one degree in a range between 0 to 360 degree for ϕ and in a range between 0 to 90 degree for θ. The output of the algorithm is a set of postures appropriate for milling at all cutter positions given as input data, where "appropriate posture" means that the cutter in such posture can execute milling at all given points with a sufficiently large angular clearance.

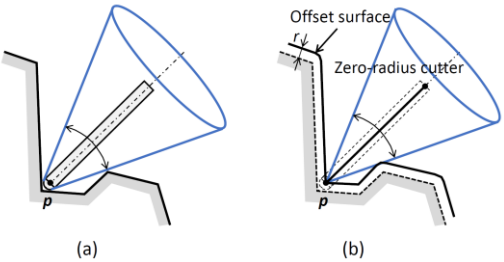

Fig. 3: AC for a cutter with respect to a mold surface (a) and AC for a zero-radius cutter with respect to an offset surface of the mold (b).

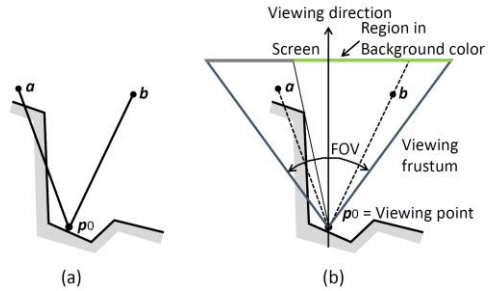

Fig. 4: Accessibility analysis of a zero-radius cutter $\boldsymbol{ap}_0$ and $\boldsymbol{bp}_0$ using the perspective projection.

Algorithm outline

In the computation of AC, we use a very thin cutter of zero-radius and an expanded mold shape obtained by offsetting the mold surface by the cutter radius r (see Fig. 3). An AC for a normal cutter with respect to the mold shape and another AC for the zero-radius cutter with respect to the offset shape have equal peak angle as shown in the figure.

Our main idea of the AC computation is to use the polygon rendering function of GPU (Graphics Processing Unit). We have proposed an AC computation algorithm using the rendering hardware in [2]. The algorithm proposed in this paper is an improvement of our prior one. Consider a problem to judge whether a single point $\boldsymbol{p}_0$ is machinable with a zero-radius cutter in a certain posture. Zero-radius cutter in two different postures is illustrated as segments $\boldsymbol{ap}_0$ and $\boldsymbol{bp}_0$ in Fig. 4(a). Cutter $\boldsymbol{bp}_0$ can machine the point $\boldsymbol{p}_0$. On the other hand, cutter in posture $\boldsymbol{ap}_0$ is not acceptable because the segment intersects the object. This cutter accessibility analysis can be achieved by using the perspective projection in the 3D computer graphics. A viewing point is placed on $\boldsymbol{p}_0$ and a displaying screen in a certain background color is prepared in a sufficient distance from $\boldsymbol{p}_0$. A viewing image of the object and points $\boldsymbol{a}$ and $\boldsymbol{b}$ from $\boldsymbol{p}_0$ are rendered in the screen using the perspective projection (see Fig. 4(b)). If the picture of the point appears in the background of the image (point $\boldsymbol{b}$ in the figure) then its corresponding cutter posture (segment $\boldsymbol{bp}_0$) is acceptable for machining the point, otherwise it is not.

By using a similar method, ACs of a zero-radius cutter for machining a point $\boldsymbol{p}_0$ can be computed [2]. A viewing point is placed on $\boldsymbol{p}_0$ and the offset shape of the mold part is rendered in the screen using the perspective projection. In the rendered image, regions in the background color represents a set of end points of segments corresponding to the zero-radius cutters machinable at $\boldsymbol{p}_0$. Consider a pyramid shape whose bottom face is the background color region in the screen and whose peak point corresponds to $\boldsymbol{p}_0$ (see Fig. 5). We call this shape "visibility pyramid". For each cutter posture (ϕ, θ)

allowed in the ranges for ϕ and θ, a straight line starting from $\boldsymbol{p}_0$ and being extended in the cutter axis direction is checked. If the line reaches the bottom face of the pyramid, then the cutter in this posture is judged accessible to $\boldsymbol{p}_0$. For each accessible line, a cone whose peak point is at $\boldsymbol{p}_0$ and coaxial to the line is considered. The peak angle of such cone is enlarged until the cone surface touches some side faces of the visibility pyramid as shown in the figure. Obtained cone with the maximum peak angle allowed in the visibility pyramid becomes the AC for the cutter posture.

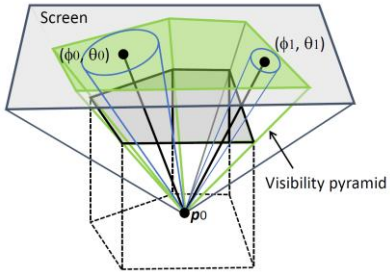

Fig. 5: Definition of a visibility pyramid and the determination of two ACs using the pyramid.

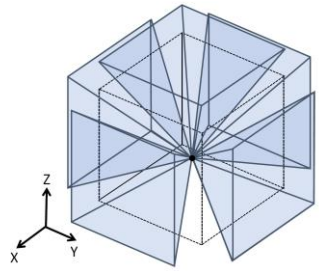

Fig. 6: Perspective projection for 5 different viewing directions.

The AC must be acceptable for multiple points given as the cutter positions. Such common AC can be computed by repeating the rendering operation of the offset shape for each point and overwriting the images to obtain a final image. The background-color-region in the final image is extracted and the visibility pyramid is computed. The AC obtained by using this pyramid corresponds to the common AC for all cutter positions. In the perspective projection, a viewing direction and a field of view (FOV) parameter must be specified. In our current implementation, perspective projections in 5 different viewing directions are used which are +X, -X, +Y, -Y and +Z directions. 90 degree is used as the Fovy value as illustrated in Fig. 6. After the rendering in the 5 directions, obtained images are stitched to realize a single image covering all cutter postures allowed in the ranges for ϕ and θ.

Visible surface offsetting

In the algorithm mentioned above, rendering of the offset shape is the most time-consuming task. Offset shape of a polyhedron by radius r corresponds to a Boolean union shape of spheres of radius r, cylinders of radius r and plate shapes of thickness $2r$ being placed on the vertices, edges, and faces of the polyhedron, respectively. In a simple method, the rendering of these component shapes is iterated for all cutter positions. The necessary cost for rendering an object is basically proportional to the number of polygons of the object. Since spheres and cylinders of the offset shape are finely tessellated before rendering, their rendering cost (especially rendering cost of tessellated spheres) dominates the total rendering cost of the offset shape.

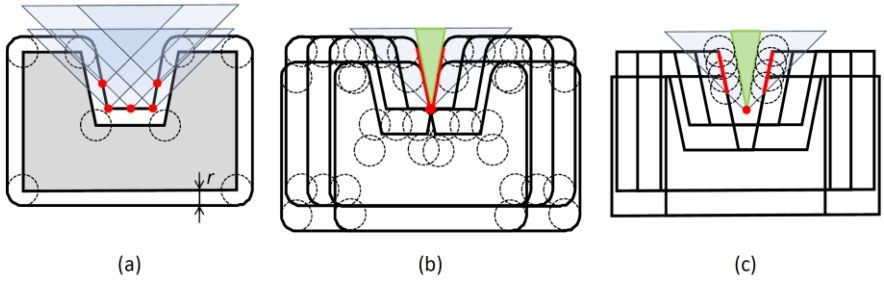

Fig. 7: A polyhedron of 8 vertices and 5 cutter positions (a). Overwriting result of 5 offset shapes of the polyhedron (b). Offsetting result of the visible surface obtained by overwriting 5 polyhedrons (c).

To reduce the rendering cost, new method named visible surface offsetting is developed. Fig. 7 illustrates the idea. In this figure, a rendering operation of an offset shape of a polyhedron with m vertices (m = 8 in the figure) is considered. This operation is iterated for n cutter positions. In Fig. 7(a), red points represent the cutter positions. For each cutter position, viewing frustum is illustrated as a

triangle. Since m is often more than 100,000 and n is more than 10,000, tremendous number of spheres must be rendered in the simple method. After rendering the offset shape n times, their overwritten image is remained in the screen. Visible surface in the screen is specified by red curves in Fig 7(b). Visibility pyramid (green shape in the figure) is constructed based on such visible surfaces. As shown in the figure, many spheres do not contribute the final image in the screen.

Our visible surface offsetting can eliminate the rendering of such non-contributing spheres. In this method, rendering operation of the mold shape (not offset shape) is iterated for each cutter position and their overwritten image is obtained as shown in Fig. 7(c). After the rendering, the coordinates of the points corresponding to pixels of the visible surface (red curves in Fig. 7(c)) are sampled. Such coordinates are easily computed using the pixel location in the frame buffer and its corresponding depth information. Spheres are then placed on the sampled points and their image is rendered again by using the perspective projection. Visibility pyramid is finally constructed based on the rendering result of the spheres. In this method, the number of spheres to render becomes at most the total number of pixels in the screen (1024 x 1024 in our implementation) which is usually far less than m x n.

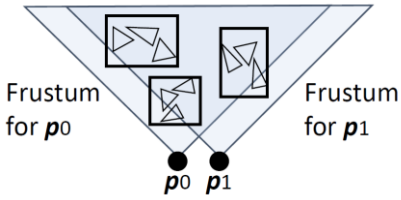

Fig. 8: Viewing frustum for 2 close viewing points.

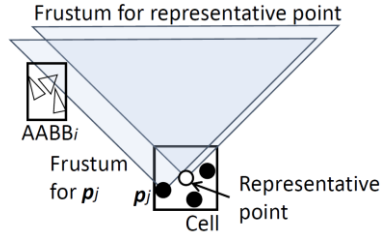

Fig. 9: Culling using the representative frustum.

Culling using hierarchical AABB and grid structure

To further reduce the number of polygons to render, culling operation is introduced. In the perspective projection, the polygons locating outside of the viewing frustum do not contribute the rendering result. These non-contributing-polygons can be efficiently detected and excluded from the rendering by using the hierarchical axis-aligned bounding boxes (AABB) [1]. The surface polygons of the input model are classified into small groups according to their proximity. For each group, an AABB that tightly contains the polygons within is defined. Obtained AABBs can be organized in a hierarchical binary tree structure. In the rendering operation with a viewing frustum of the perspective projection, hierarchical AABB tree is traversed from the root node in the depth first manner. At each node, positional relationship between the AABB corresponding to the node and the viewing frustum is checked. If the AABB locates completely outside of the frustum, then polygons in the AABB do not contribute the rendering result and the traversal of its child nodes can be canceled, otherwise the traversal continues. After the traversal, leaf AABBs holding the visible polygons are obtained.

Execution of the culling operation for each viewing point (= cutter position) cost much. This problem can be solved by using the spatial coherency in the perspective projection. In Fig. 8, $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are 2 close viewing points. In the perspective projection with the same viewing direction, a viewing frustum for $\boldsymbol{p}_0$ and another frustum for $\boldsymbol{p}_1$ concern similar set of AABBs as shown in the figure. To utilize this spatial coherency for reducing the number of culling operations, given cutter positions are classified to a spatial grid structure with small cubic cells. Culling operation with the hierarchical AABB is executed only once for each cell, and the culling result is shared by all point in the same cell. In this method, the center point of the cell is used as a representative viewing point for the cell and the viewing frustum for this point is used in the culling operation. Some AABBs visible from a point in the cell are not recognized as visible one because they locate outside of the representative frustum. For example, AABB$_i$ in Fig. 9 is visible from viewing point $\boldsymbol{p}_j$ in the cell, however it is not selected as a visible one because it locates outside of the representative frustum for the center point of the cell. To properly select such boxes, AABBs are expanded by the half size of the cell in the x, y, and z-directions before the culling operation.

Numerical experiment

A system for selecting appropriate cutter postures and their ACs common for all cutter positions was implemented using Visual C++, CUDA 7.5, and OpenGL. Series of computational experiments were

performed using a PC with Intel Core i7 Processor (2.6 GHz), 16 GB memory, and an nVIDIA GeForce GTX-960M GPU. We applied the system to five polyhedral models of molds and point set representing the cutter positions. For the purpose of maintaining confidentiality, computation result of a simple example model is only given. Fig. 10(a) illustrates a part with 45,174 polygons. Red points in the figure represents the cutter position data. 11,757 points are specified in this example. As shown in a close-up figure (b), points are classified to a spatial cell structure. In this example, a ball end cutter of radius 3.0mm is specified for semi-finishing. Fig. 10(c) shows the computation result. 3.1 seconds are necessary for this case. System using our improved algorithm is approximately 10 times faster than the system using the prior one. In the figure, colored surface is a Gauss map representing the appropriate cutter postures for 3+2 axis milling. Color on the surface corresponds to the peak angle of the AC for each cutter posture common for all cutter positions. Red corresponds to 1.0 degree and blue corresponds to the maximum peak angle of the ACs allowed by the sample case.

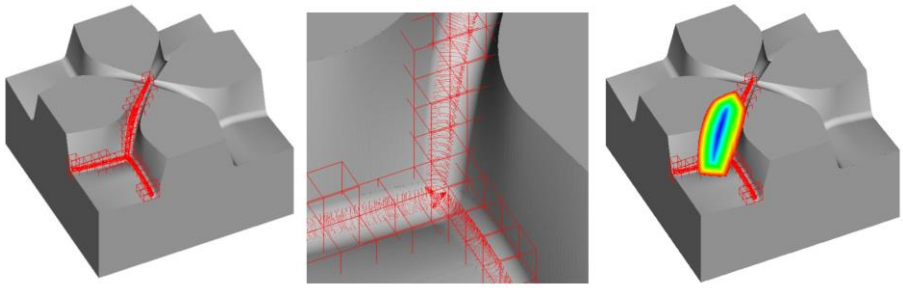

Fig. 10: Mold model and cutter positions (a), cell structure for recording the cutter positions (b), and appropriate cutter postures for executing 3+2 milling at all cutter positions (c).

Conclusions:

In this paper, a novel method is proposed for automatically computing all appropriate cutter postures for the 3+2 axis milling. Proposed algorithm realizes the computation using the polygon rendering hardware. For further accelerating the computation, the following two methods are introduced:

(1) Rendering cost of spheres in the offset shape dominates the total computation cost. To reduce the number of rendering spheres, a new method named visible surface offsetting is developed.

(2) Culling operation using the hierarchical AABB is introduced for further accelerating the computation. Spatial coherency in the perspective projection enables the reduction of the number of the culling operations.

References:

[1] Moller, T; Haines E.: Real-time rendering, A K Peters, 1999.

[2] Morimoto, K.; Inui, M.: A GPU based Algorithm for Determining the Optimal Cutting Direction in Deep Mold Machining, Proc. of IEEE International Symposium on Assembly and Manufacturing, 2007. DOI: 10.1109/ISAM.2007.4288473

[3] Morishige, K.; Takeuchi, Y.: 5-axis control rough cutting of an impeller with efficiency and accuracy, Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on, 1997, 1241-1247. 10.1109/ROBOT.1997.614307

[4] Spitz, S.N.; Spyridi, A.J.; Requicha, A.A.G.: Accessibility Analysis for Planning of Dimensional Inspection with Coordinate Measuring Machines, IEEE Trans. Robotics and Automation, 15 (4), 1999, 714-727. 10.1109/70.782025

[5] Takeuchi, Y.; Idemura, T.; Sata, T.: 5-axis Control Machining and Grinding Based on Solid Model, CIRP Annals - Manufacturing Technology, 40 (1), 1991, 455-458. https://doi.org/10.1016/S0007-8506(07)62028-9

[6] Tseng, Y.J.; Joshi, S.: Determining feasible tool-approach directions for machining Bézier curves and surfaces, Computer-Aided Design, 23 (5), 1991, 367-379. https://doi.org/10.1016/0010-4485(91)90030-Z