COMPUTER-AIDED DESIGN & APPLICATIONS, 2018
VOL. 15, NO. 5, 667–676
https://doi.org/10.1080/16864360.2018.1441234

Fast computation of accessibility cones for assisting 3 + 2 axis milling

Masatomo Inui, Shinji Nagano and Nobuyuki Umezu

Ibaraki University, Japan

ABSTRACT
In this paper, we propose an algorithm for computing all appropriate cutter postures in the 3 + 2 axis milling of the mold part. As a measure of the appropriateness of the cutter posture, the peak angle of the accessibility cone (AC) is used. We use the polygon rendering function of the graphics processing unit for computing the AC. In this method, the necessary cost in the computation is basically proportional to the number of polygons to render in generating an image of the offset shape. A novel technology named "visible surface offsetting" is developed for reducing the rendering cost of the offset shape. An experimental system is implemented and some computation results are demonstrated. Our system can determine the cutter postures appropriate for 3 + 2 axis milling of the mold part in a few minutes.

KEYWORDS
Cutter posture determination; accessibility analysis; offset computation; parallel processing; GPU

1. Introduction

In the mold machining, rough milling with a large cutter is applied first to efficiently remove excess material, then semi-finishing with a smaller cutter removes the material remained in corner shapes of the mold. In the automobile industry, molds with very deep shape are used for producing large plastic parts, such as instrument panels and bumpers. 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 process (see Fig. 1(a)). Since large deformation of the cutter is unavoidable with a long shank, it is difficult to realize stable machining in the 3-axis milling.

To solve this problem, many manufacturers in Japan use 3 + 2 axis milling for the semi-finishing in the mold machining. 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(b). 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.

In the 3 + 2 axis milling, determination of a proper cutter posture is a critical task. In the following discussion, milling operation with a ball-end cutter of radius r is assumed. Position $\boldsymbol{p}$ of the ball end cutter is represented by the center point of the spherical blade of the cutter. We use accessibility cone [7] as a measure for evaluating the appropriateness of the cutter posture at $\boldsymbol{p}$. For each cutter posture (ϕ, θ) specified by 2 rotational angles ϕ and θ around 2 mutually perpendicular axes of the milling machine (see Fig. 1(b)), a cone whose axis is coaxial to the spindle 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 surface as shown in Fig. 2. Such cone with the maximum peak angle is called accessibility cone (AC) of the cutter at $\boldsymbol{p}$ in posture (ϕ, θ). AC represents the angular clearance between the mold and the cutter at a specific configuration (position and posture).

In this paper, we discuss an algorithm for computing all appropriate cutter postures for the 3 + 2 axis milling, where "appropriate posture" means that the cutter in such posture can execute machining at all given points with a sufficiently large angular clearance (= with an AC of sufficiently large peak angle). We use the polygon rendering function of the graphics processing unit (GPU) for computing the AC [7]. In this method, the necessary cost for the computation is basically proportional to the number of polygons to render in generating an image of the offset shape. In this paper, a novel technology named "visible surface offsetting" is proposed for reducing the rendering cost of the offset shape. In the next section, some related studies are briefly reviewed. In Section 3, the outline of our AC computation algorithm is illustrated.

CONTACT Masatomo Inui masatomo.inui.az@vc.ibaraki.ac.jp; Shinji Nagano 17nm934r@vc.ibaraki.ac.jp; Nobuyuki Umezu nobuyuki.umezu.cs@vc.ibaraki.ac.jp

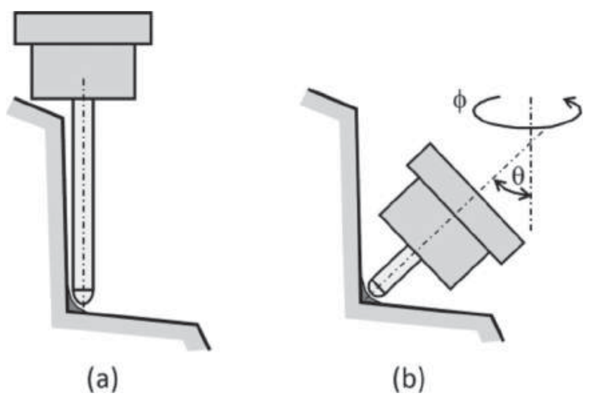

Figure 1. 3-axis milling (a) and 3 + 2 axis milling (b) for removing excess material in a corner.

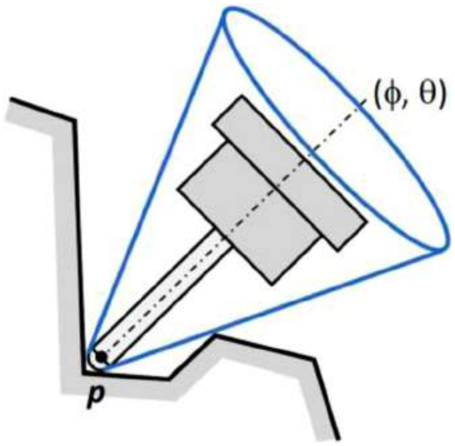

Figure 2. An accessible cone for a cutter at a specific positon ***p*** and posture (ϕ, θ).

Details of the visible surface offsetting, the novel contribution of this paper are explained in Section 4. Experimental computation results are given in Section 5, and we summarize our conclusions in Section 6.

2. Related studies

5-axis milling is becoming popular due to its ability to handle a workpiece with complex shape. Many research results are known for automatically determining the optimal posture in the 5-axis milling [5]. They can be classified to two groups, which are studies on the positioning of the cutter with respect to the surface to machine it without having gouges, and studies on the determination of the cutter posture without collisions between the shank/holder and the workpiece. Since the ball-end cutter is basically capable of machining the surface without regarding its orientation, research works in the latter category concern our study. Takeuchi et al. proposed a trial-and-error-based method for computing collision free cutter postures in the 5-axis milling [13, 14]. 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 [8, 9]. Kaneko et al. introduced the GPU technology for accelerating Morishige's algorithm [3], but their method is different from ours.

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 cutter posture (ϕ and θ) for milling a surface point. Tseng and Joshi [15] and Kang and Suh [4] used the visibility cone to determine the cutter accessibility in the 5-axis milling. Convex-hull properties of the free-form surface are used for bounding the feasible axis direction. Spitz and Requicha developed a computation method of a visibility cone for the coordinate measurement machine using the perspective projection [12]. Morimoto and Inui extended Spitz and Requicha's method for determining the cutter accessibility in the 3 + 2 axis milling [7].

In the computation of the accessibility cone, the offset surface of the object is necessary. Conventional techniques for offsetting 3D objects [1, 10, 11] are often computationally expensive, and its model reconstruction process can be unstable. Since a picture of the offset shape is only required in the perspective-projection-based computation of the AC [7], much simple and robust method is applicable. The picture of the offset shape of a polyhedral object can be obtained by rendering spheres, cylinders and thick plates placed on the object surface as follows [2];

- Spheres of radius r are placed on all vertices of the surface where r means the offset radius.
- On each edge of the surface, a cylindrical pin shape of radius r is placed so that its center axis and the edge become coincident.
- On each polygonal face, a plate shape of the same area and thickness of $2r$ is placed so that the center plane of the plate and the face become coincident.

The rendering operation of the offset shape is the most time-consuming task in the AC computation using the perspective projection. Visible surface offsetting proposed in this paper can reduce the rendering cost of the offset shape without deteriorating the picture quality. This method is especially effective in a case of milling a complex shape with many cutting positions.

3. AC computation using perspective projection

3.1. Input and output

The input data of our algorithm consists of a polyhedral 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. Most commercial CAD systems provide a function to output the model data as a group of triangular polygons, such as in the STL format. In our current implementation, STL models are prepared so that the shape difference between the original model with the curved surface and the mesh model obtained by the tessellation becomes less than 0.01 mm. Cutter position data are obtained by using a conventional CAM system for 3-axis milling. 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 cutter postures with their corresponding ACs. The cutter in such posture can execute $3 + 2$ axis milling at all given points with a certain angular clearance given as the peak angle of its corresponding AC.

3.2. Algorithm outline

In the computation of AC, we consider a thin cutter of zero-radius and an expanded mold shape obtained by offsetting the mold surface by the cutter radius r (see Fig. 3). Points representing the cutter positions in the machining locate on the offset surface. An AC for a normal cutter with respect to the mold shape and another AC for the zero-radius cutter with respect to the expanded mold shape have equal peak angle as shown in the figure. In the following discussion, we explain the AC computation based on the zero-radius cutter and the offset shape of the mold part using Morimoto and Inui's method [7]

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

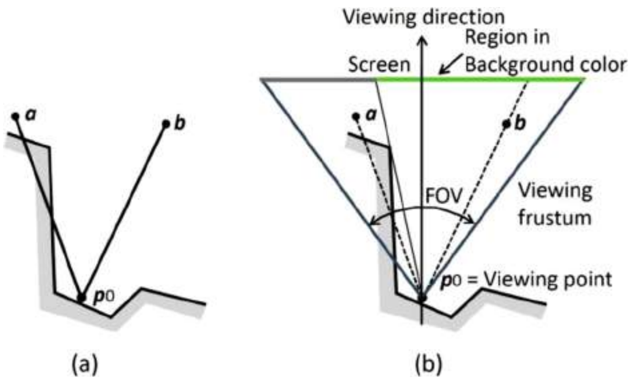

Figure 4. Accessibility analysis of a zero-radius cutter ***ap***0 and ***bp***0 using the perspective projection.

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 display 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. A viewing point is placed on $\boldsymbol{p}0$ and the offset shape of the mold part is rendered 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 display 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

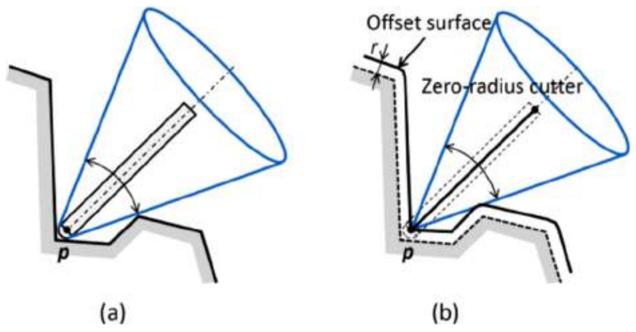

Figure 3. AC for a cutter with respect to a mold surface (a) and AC for a zero-radius cutter with respect to the offset surface of the mold part (b).

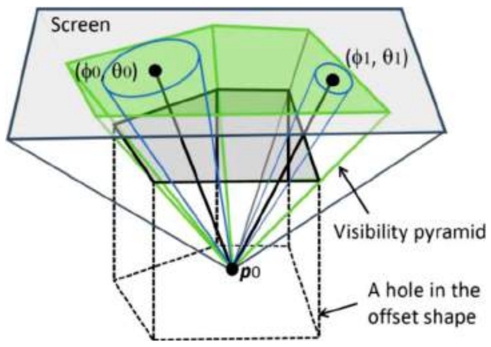

Figure 5. Definition of a visibility pyramid and the determination of two ACs using the pyramid.

is 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.

Consider an AC 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 applicable to all cutter positions. Computation of the common AC can be realized in a different manner. Select a representative point from the point set. The viewing point for the perspective projection is fixed at this representative point. For each point in the cutting positions, a vector from the point to the representative point is computed and stored. In the rendering process, the offset shape of the mold is translated by each stored vector and rendered. This translation and rendering operation is iterated for all the stored vectors. After rendering all translated offset shapes, the background color region in the frame buffer is extracted and the visibility pyramid is constructed. The latter method is employed in our AC computation software.

In the perspective projection, a viewing direction and a field of view (FOV) parameter are necessary. Since FOV must be less than 180 degree, the peak angle of the visibility pyramid becomes less than 180 degree also. Therefore, a single projection cannot support the AC computation for all cutter postures (ϕ, θ) allowed in the range (0 to 360 degree for ϕ and 0 to 90 degree for θ). A picture projected to the screen has large distortion if too large FOV value (near 180 degree) is used, and it causes less accuracy in the AC computation result. To solve this problem, we compute multiple pictures with different viewing directions and a smaller FOV value, and combine the pictures to a single one for computing the AC for all possible cutter postures. 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 FOV value as illustrated in Fig. 6. After rendering in the 5 directions, obtained images are stitched to a single image covering all cutter postures allowed in the ranges for ϕ and θ.

4. Visible surface offsetting

In the algorithm mentioned above, the offset shape of a mold part in different positions must be rendered many

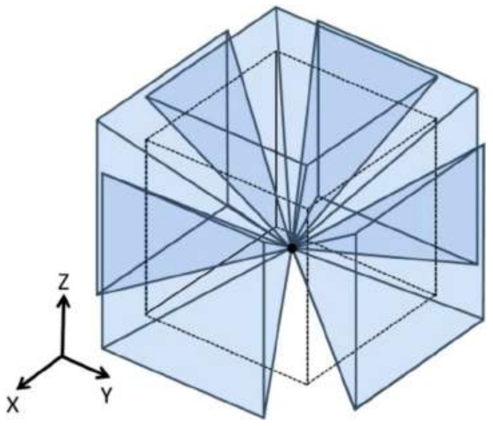

Figure 6. Perspective projection for 5 different viewing directions.

times. Offset shape of a polyhedral object corresponds to a combined shape of spheres, cylinders, and plate shapes being placed on the vertices, edges, and faces of the object, 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 the rendering, their rendering cost (especially rendering cost of tessellated spheres) dominates the total rendering cost of the offset shape.

4.1. Reduction of rendering spheres

To reduce the rendering cost, new method named visible surface offsetting is developed. Fig. 7 illustrates the basic idea. In this figure, a rendering operation of the 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), 5 red points $\boldsymbol{p}0$, $\boldsymbol{p}1$, $\boldsymbol{p}2$, $\boldsymbol{p}3$, $\boldsymbol{p}4$ represent the cutter positions. Since m is usually more than 100,000 and n is more than 10,000, huge number of spheres must be rendered in our original method. After rendering the offset shape n times, their overwritten image is obtained in the frame buffer. In Fig. 7(b), such rendering result is illustrated for a case that $\boldsymbol{p}2$ is selected as a representative viewing point. A viewing frustum is given in light blue color in the same figure. Visible portion of the offset surface is specified by red curves. Visibility pyramid (green shape in the figure) is constructed based on such visible surfaces. As shown in the figure, most spheres of the offset shape do not contribute the final image.

Our visible surface offsetting can eliminate the rendering of such non-contributing spheres. In this method, rendering operation of the part shape (not its offset shape) is iterated for each cutter position and their

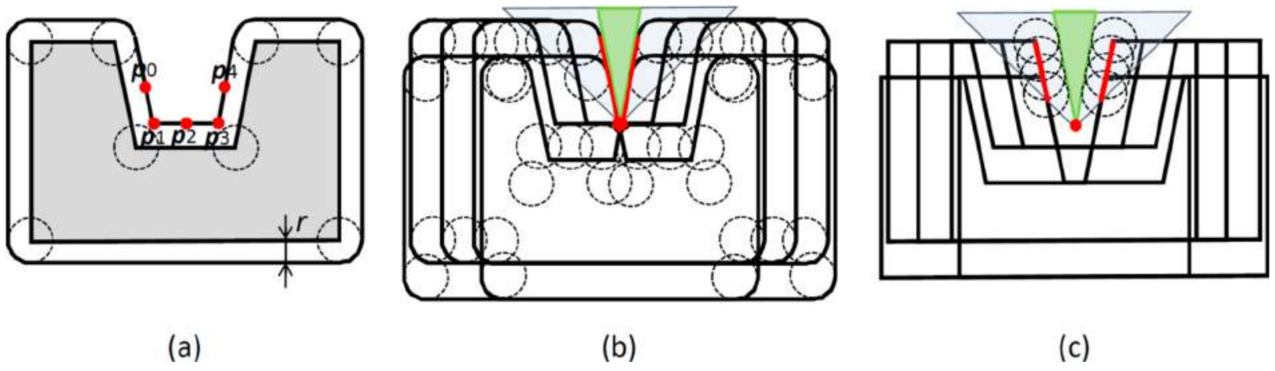

Figure 7. A polyhedron with 8 vertices, its offset shape, and 5 cutter positions (a). Overwriting result of 5 offset shapes of the polyhedron (b). Offsetting result of the visible surface obtained by overwriting 5 polyhedrons (c).

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 computed using the pixel location in the frame buffer and its corresponding depth information. In Fig. 8, a determination process of the coordinates of a visible point ***p*** at pixel (*i*, *j*) with depth value *d* in the frame buffer is illustrated. In the hidden surface removal using the depth buffer, two clipping planes named *near* and *far* perpendicular to the viewing direction are defined [6]. These two planes limit the visible range in the viewing direction. Depth value 0.0 and 1.0 are assigned to *near* and *far* planes, respectively. Consider a ray from the viewing point going through the pixel at (*i*, *j*). Coordinates of a visible point ***p*** on the ray can be determined by using the depth information *d* representing the relative position of ***p*** with respect to *near* and *far* planes. Spheres are placed on such visible points in the display and their image is rendered again by using the perspective projection. Visibility pyramid is finally constructed based on the rendering result of the spheres as shown in Fig. 7(c). In this method, the number of spheres to render is limited by the total number of pixels in the display (1024×1024 in our implementation) which is usually far less than *m* x *n*.

Rendering result using the visible surface offsetting (Fig. 7(c)) becomes identical to the rendering result obtained by our original method (Fig. 7(b)). Image difference between the two rendering results can occur when the offset result of certain points hidden by other shapes appear in the display, but such result is never obtained in the perspective projection. Point ***q*** in Fig. 9 is such an invisible point hidden by a wedge shape. Consider a line connecting ***q*** and the viewing point. A visible point ***p*** must exist on the line between ***q*** and the viewing point as shown in the figure. Since ***p*** is visible, a sphere ***S*** of an offset radius *r* is placed as its center point is at ***p*** in our visible surface offsetting. Consider a sphere ***T*** of the same radius is placed at ***q*** as an offset result of the hidden point ***q***. Since ***T*** is more distant than ***S*** from the viewing point,

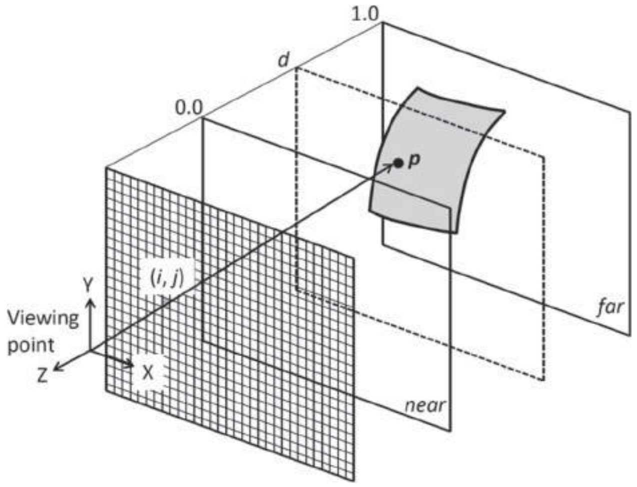

Figure 8. Determination of the coordinates of a point ***p*** visible at pixel (*i*, *j*) with the depth value *d*.

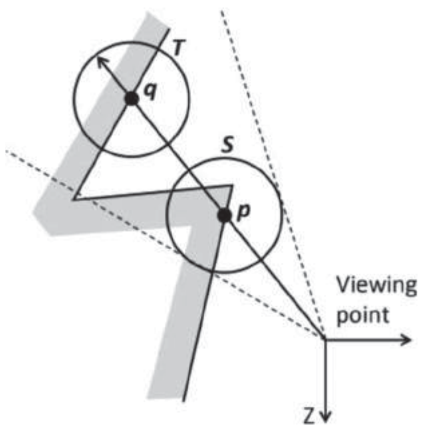

Figure 9. Offset shape ***T*** of an invisible point ***q*** never appears in the display because offset shape ***S*** of a visible point ***p*** hides ***T***.

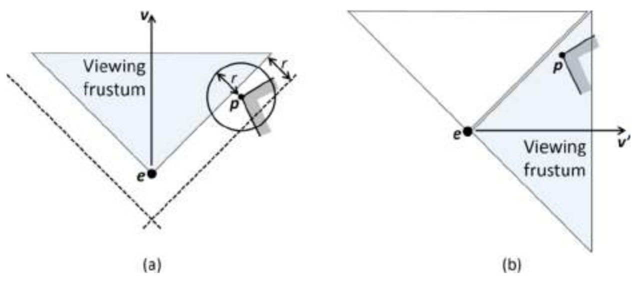

Figure 10. Offset shape of a point ***p*** existing in the outside of the viewing frustum can affect the rendering result of the offset shape.

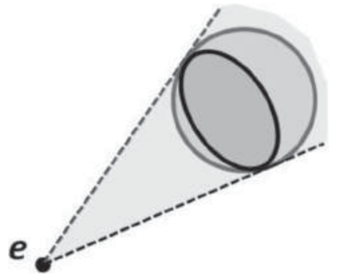

Figure 11. Silhouette of a sphere viewing from point ***e***.

S completely hides ***T*** and the image of ***T*** never appears in the display.

In our method, some surface points locating outside of the viewing frustum must be considered in the rendering operation of the spheres. Fig. 10(a) illustrates a case. Consider a surface point ***p*** locating in the external part of the viewing frustum. This point is not visible when observing the objects from ***e*** in the viewing direction ***v***. A part of the sphere of radius *r* whose center point is at ***p***, however, can enter the viewing frustum and it affects the rendering result of the offset shape. To evaluate the effect of such points locating outside of the viewing frustum, we implement the visible surface offsetting algorithm in the following 2 step manner. Since such point ***p*** is visible when observing in a different viewing direction (viewing direction ***v'*** in Fig. 10(b)), our algorithm executes the rendering operation of the part shape for 5 different viewing directions (+X, -X, +Y, -Y, and + Z) first and collects all visible surface points. In the second step of the algorithm, rendering operation of the spheres is executed for each viewing direction to obtain the picture of the offset shape. In this process, spheres are rendered not only for the surface points in the viewing frustum but also for some points locating outside of the viewing frustum if their distances from the viewing frustum are less than or equal to *r*.

In the computation of the AC, a background color region left in the frame buffer after the rendering is extracted and used as the bottom face of the visibility pyramid. A silhouette picture of the offset shape is only necessary in this purpose. As shown in Fig. 11, the silhouette of a sphere viewing from a point ***e*** becomes a circular disk corresponding to the tangent curve between the sphere and a cone whose peak vertex is at ***e***. We improved the AC computation software so that it does not render the tessellated spheres, but it renders only their silhouettes. The number of polygons required for approximating the circular disk is much smaller than the number of polygons covering the tessellated spheres, therefore the silhouette picture of the offset shape can be obtained in a shorter time period

4.2. Culling using hierarchical AABB and grid structure

To further reduce the computation time, culling operation is introduced in the rendering of the part shape. 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) [6].

The surface polygons of the input model are classified into small groups according to their proximity. Consider *n* polygons forming the model surface. An AABB that tightly confines the polygons is defined by measuring the coordinate ranges of the polygons in the x-, y-, and z-directions. One root AABB is defined that holds all the polygons of the model. Polygons in the AABB are sorted and classified to two groups with *n*/2 polygons. For each polygon group, a smaller AABB is formed and registered as a descendant of the original AABB. The process of defining descendant AABBs is iterated until the number of the polygons in a group becomes less than or equal to a predetermined number *nmax*, and a binary AABB tree is obtained. *nmax* is set to be 4 in our current implementation. This number is determined based on numerical experiments.

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 descendant 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. 12, ***p***0 and ***p***1 are 2 close viewing

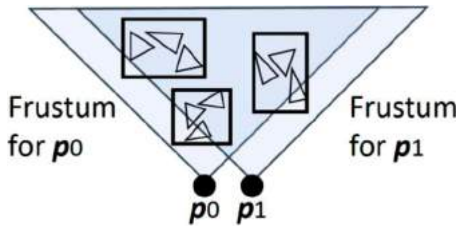

Figure 12. Viewing frustums for 2 close viewing points.

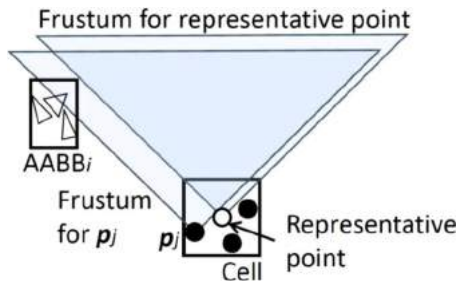

Figure 13. Culling using the representative frustum.

points. In the perspective projection with the same viewing direction, a viewing frustum for $\boldsymbol{p}0$ and another frustum for $\boldsymbol{p}1$ usually 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. 13 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.

5. Numerical experiments

A system for computing 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 seven cases of polyhedral models of mold parts and points representing the cutter positions for semi-finishing them. Tab. 1 shows number of polygons of the models, cutter radius, number of cutter positions, and the required time for computing the appropriate cutter postures (data given in the fifth column of the table) for the sample cases. First three cases (case A, B and C) are simple models for tests. Case D, E, F and G are actual mold cavity and core models and cutter position data for machining them. As shown in the table, more computation time is necessary for complex cases with many polygons and many cutter positions. Our system can determine the appropriate cutter postures for complex cases in a few minutes.

Table 1. Required time for computing appropriate cutter postures.

Case	Number of polygons	Cutter radius (mm)	Number of cutter locations	Computation with AABB and cells (s)	Computation without AABB and cells (s)
A	2,518	1.0	3,758	6.93	7.16
B	2,968	1.0	3,848	8.43	8.29
C	45,174	3.0	11,757	10.25	11.31
D	844,180	5.0	5,142	8.77	110.76
E	844,180	5.0	5,417	11.68	115.63
F	618,143	0.5	23,016	58.55	357.94
G	618,143	0.5	24,310	87.70	368.79

In the same table, necessary computation time without using the hierarchical AABB and the cell structure are also given (time given in the rightmost column). For simple cases (case A, B and C), performance difference between the system using the AABB and cell structure and the system without using them is small, however it becomes very large for complex cases (cases D, E, F and G) with many polygons and many cutter positions.

For the purpose of maintaining confidentiality, computation results for case C and E are only illustrated in Fig. 14 and Fig. 15, respectively. In these figures, (a) show sample parts. Red points in the figures represent the cutter position data. As shown in close-up figures (b), points are classified to a spatial cell structure. In each cell, 10 points are stored on the average. (c) in Fig. 14 and Fig. 15 show the computation result. In these figures, colored spherical 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 (ϕ, θ) common for all cutter positions. Red corresponds to cutter postures with 0.1 degree peak angle and blue corresponds to postures with the maximum peak angle allowed in the sample case.

As mentioned in section 4, the same rendering result is obtained by directly rendering the offset surface of the part for all cutter positions. Since the collecting operation of the visible surface points in the display and the rendering operation of the circular disks at the collected points are not necessary, some reduction of the computation

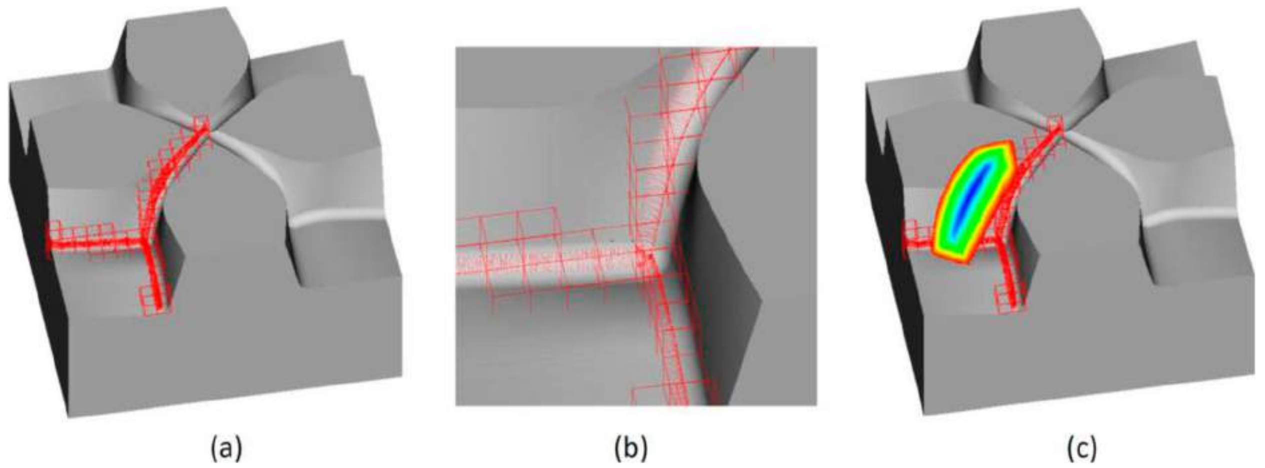

Figure 14. A simple mold model and cutter positions of case C (a), cell structure for recording the cutter positions (b), and appropriate cutter postures for executing 3 + 2 milling for all cutter positions (c).

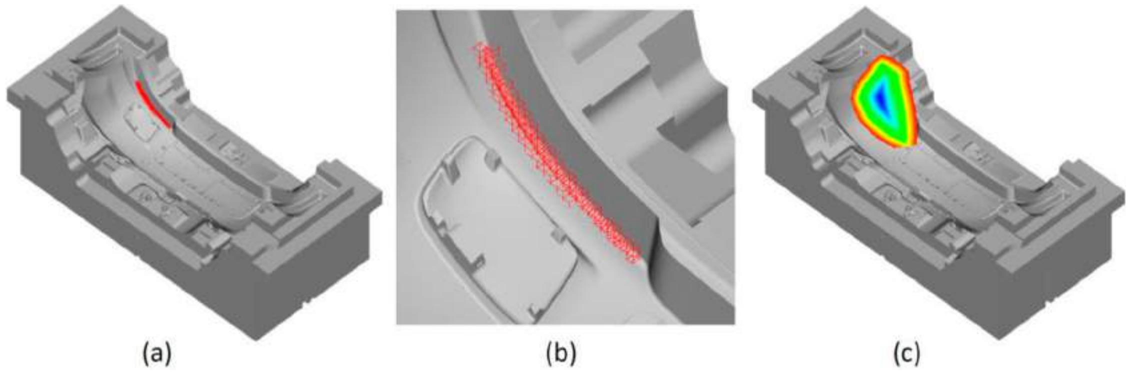

Figure 15. A cavity model and cutter positions of case E (a), cell structure for recording the cutter positions (b), and appropriate cutter postures for executing 3 + 2 milling for all cutter positions (c).

time can be expected in this method. The offset surface of a polyhedral object is generally obtained by shifting the polygonal faces in their normal vector directions by the offset radius and by inserting the cylindrical surfaces and spherical surfaces to fill the gaps between the shifted polygons. These curved surfaces are further tessellated to a set of small polygonal faces before the rendering. The result offset surface thus contains several times more polygons compare to the original polyhedral model. Increase of the number of polygons causes more computation time in the rendering operation especially for cases with many cutter positions.

Tab. 2 shows a comparison of the necessary computation time using our visible surface offsetting method and time using the tessellated offset surface in the rendering. Fig. 16(a) shows the offset shape used in the computation. This shape is obtained by offsetting the model for case C by 3 mm. The original model has 45,174 surface polygons and the offset model has 216,746 surface polygons. Cutter position data with different number of points are used in the experiments. These data are generated by duplicating the same position data shown in Fig. 14(a) multiple times. Appropriate cutter postures computed by using the offset surface are shown in Fig. 16(b). This result is the same to the result given in Fig. 14(c). Our visible surface offsetting method shows the better performance as the number of the cutter positions increases. The number of the polygons of the offset shape can be reduced by directly offsetting the curved surface of the original CAD model (not tessellated model) prior to the tessellation. Offsetting an object with curved surfaces is still a difficult problem

Table 2. Necessary computation time using part shape in the rendering operation and time using tessellated offset shape in the rendering operation.

Number of cutter locations	Computation using part shape (s)	Computation using offset shape (s)
11,757	10.25	7.15
23,514	12.05	11.45
47,028	15.30	19.32
94,056	22.30	36.91
188,112	33.59	70.94
376,224	62.69	137.50

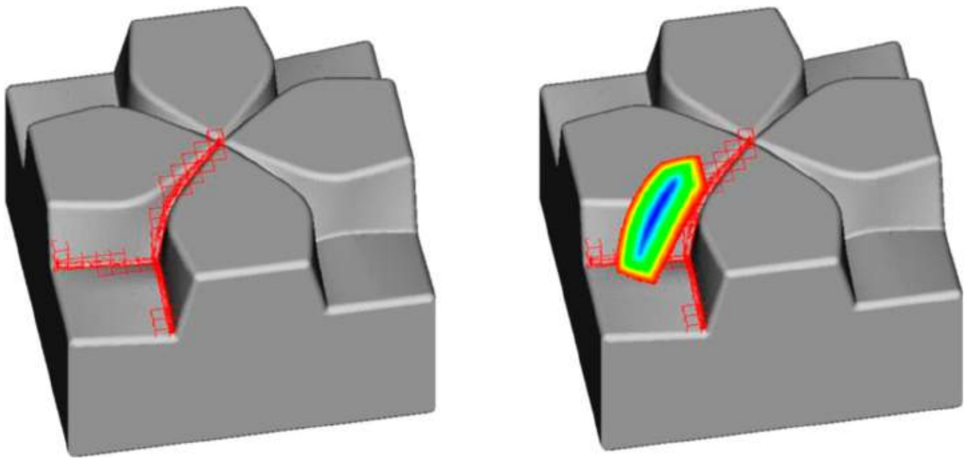

Figure 16. Computation result using the tessellated offset surface.

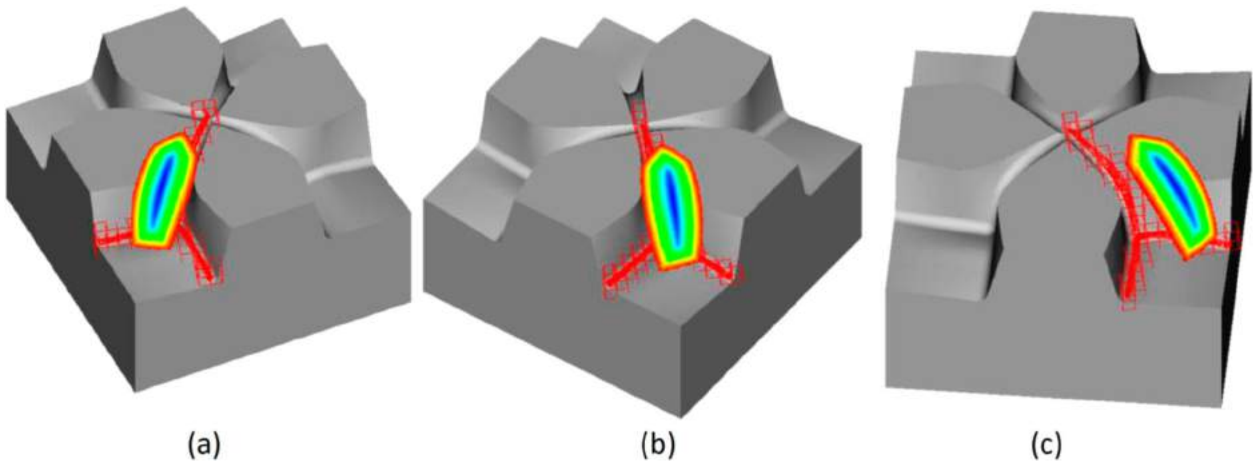

Figure 17. Computation result after rotating the part shape and cutter positions. (a) rotation by 30 degree, (b) rotation by 60 degree, and (c) rotation by 90 degree.

and it often needs large computation time in preparing the data.

Since our method uses the polygon rendering function in the computation, the result is affected by the tessellation accuracy of the object, polygonization accuracy of the circular disk in the visible surface offsetting, the viewing direction, and the pixel resolution of the display. In these parameters, we consider that the rendering direction has impact on the computation accuracy. To check the robustness of the computation by changing the viewing direction, we rotate the part shape and the cutter positions around the z-axis every 15 degrees and execute the computation. Fig. 17 illustrates the computation result for cases with rotation angles 30 degree, 60 degree and 90 degree, respectively. As shown in the figure, similar distributions of the peak angles of the AC are obtained for three cases. We checked the AC with the maximum peak angle for three cases and we found that the variation of the peak angles is less than 0.1 degree. This result is well acceptable for our cutter posture determination purpose.

6. Conclusions

In this paper, we discuss an algorithm for computing all appropriate cutter postures in the $3 + 2$ axis milling of the mold part. As a measure of the appropriateness of the cutter posture, the peak angle of the accessibility cone is used. Our algorithm realizes the fast computation using the polygon rendering hardware. Rendering operation of spheres in the offset shape consumes most part of the computation time. To reduce the number of rendering spheres, a novel method named visible surface offsetting is developed. Culling operation using the hierarchical AABB is introduced for further accelerating the rendering operation. Spatial coherency in the perspective projection enables the reduction of the number of the culling operations. An experimental system is implemented and some computation results are demonstrated. Our system can determine the cutter postures appropriate for machining the mold part using $3 + 2$ axis milling in a few minutes.

Acknowledgements

This research work was supported by JSPS KAKENHI Grant Number 17K06075.

ORCID

Masatomo Inui http://orcid.org/0000-0002-1496-7680
Shinji Nagano http://orcid.org/0000-0001-7197-4425
Nobuyuki Umezu http://orcid.org/0000-0002-7873-7833

References

[1] Forsyth, M.: Shelling and Offsetting Bodies, Proceedings of Third Symposium on Solid Modeling and Applications, 1995, 373-381. doi:10.1145/218013.218088

[2] Inui, M.: Fast inverse offset computation using polygon rendering hardware, Computer-Aided Design, 35 (2), 2003, 191–201. https://doi.org/10.1016/S0010-4485(02)00052-0

[3] Kaneko, J.; Horio, K.: Fast Determination Method of Tool Posture for 5-Axis Control Machining Using Graphics Hardware, Journal of the Japan Society for Precision Engineering, 72(8), 2006, 1012-1017 (in Japanese).

[4] Kang, J.-K.; Suh, S.-H.: Machinability and set-up orientation for five-axis numerically controlled machining of free surfaces, The International Journal of Advanced Manufacturing Technology, 13 (5), 1997, 311-325. https://doi.org/10.1007/BF01178251

[5] Makhanov, S.S.: Adaptable geometric patterns for five-axis machining: a survey, The International Journal of Advanced Manufacturing Technology, 47 (9), 1167-1208. https://doi.org/10.1007/s00170-009-2244-z

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

[7] 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. https://doi.org/10.1109/ISAM.2007.4288473

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

[9] Morishige, K.; Takeuchi, Y.; Kase, K.: Tool Path Generation Using C-Space for 5-Axis Control Machining, J. Manuf. Sci. Eng., 121 (1), 1999, 144-149. https://doi.org/10.1115/1.283056

[10] Rossignac, J.R.; Requicha, A.A.G.: Offsetting Operations in Solid Modelling, Computer Aided Geometric Design, 3(2), 1986, 129-148. https://doi.org/10.1016/0167-8396(86)90017-8

[11] Satoh, T.; Chiyokura, H.: Boolean Operations on Sets Using Surface Data, Proceedings of ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications, 1991, 119-126. https://doi.org/10.1145/112515.112536

[12] 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. https://doi.org/10.1109/70.782025

[13] 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

[14] Takeuchi, Y.; Watanabe, T.: Generation of 5-Axis Control Collision-Free Tool Path and Postprocessing for NC Data, CIRP Annals - Manufacturing Technology, 41(1), 1992, 539-542. https://doi.org/10.1016/S0007-8506(07)61263-3

[15] 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