






## Graphical Assistance for Determining Cutter Axis Directions in 3+2-Axis Machining

Masatomo Inui<sup>1</sup> , Shutaro Taguchi<sup>2</sup>  and Nobuyuki Umezu<sup>3</sup> 

<sup>1</sup>Ibaraki University, [masatomo.inui.az@vc.ibaraki.ac.jp](mailto:masatomo.inui.az@vc.ibaraki.ac.jp)

<sup>2</sup>Ibaraki University, [21nm459h@vc.ibaraki.ac.jp](mailto:21nm459h@vc.ibaraki.ac.jp)

<sup>3</sup>Ibaraki University, [nobuyuki.umezu.cs@vc.ibaraki.ac.jp](mailto:nobuyuki.umezu.cs@vc.ibaraki.ac.jp)

Corresponding author: Masatomo Inui, [masatomo.inui.az@vc.ibaraki.ac.jp](mailto:masatomo.inui.az@vc.ibaraki.ac.jp)

**Abstract.** Recently, the use of 3+2-axis machining, in which machining is performed by tilting the direction of the cutter spindle axis, has increased in the machining of parts with complex shapes, such as impellers and airplane parts. This paper describes a novel interactive software technique for assisting 3+2-axis machining. This software computes a range of cutter postures without interferences with the machine part for each point in the cutter path. The possible cutter postures for all points are examined, and the number of the machinable points is determined for each cutter posture. The obtained results are color-coded in the Gauss map. By referring to the color information of the map, cutter postures required for machining can be efficiently selected. By repeating the selection of the cutter posture based on the color display, the operator can semi-automatically generate the numerical control (NC) data for the 3+2-axis machining. Computational experiments were then performed to verify the feasible applicability of the software.

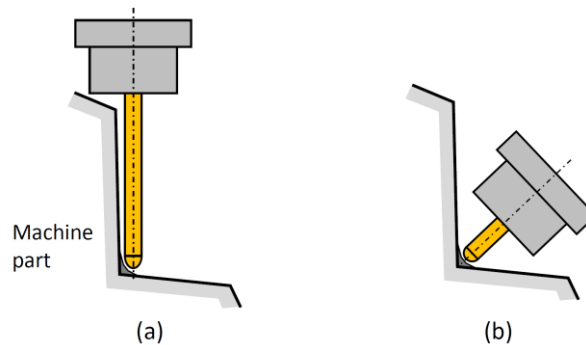
**Keywords:** 5-axis machining, Gauss map, RT cores, GPU, CAM

**DOI:** <https://doi.org/10.14733/cadaps.2023.689-703>

### 1 INTRODUCTION

Recently, the use of 3+2-axis machining, in which machining is performed by tilting the direction of the cutter spindle axis, has increased in the machining of parts with complex shape, such as impellers and airplane parts. In the 3+2-axis machining, the tool length can be shortened by setting the spindle direction properly. Accordingly, it is feasible to achieve stable machining with comparatively less tool deformation than that of the conventional 3-axis machining (Fig. 1). This type of machining enables the minimization of the number of mounting changes in the workpiece; hence, via this method, machining accuracy can be improved and machining cost minimized. Unlike simultaneous 5-axis machining with continuous changes in the spindle direction, the spindle direction is fixed in the 3+2-axis machining. Therefore, it is unnecessary to control the cutter

posture during the machining process, so that the introduction cost of the 3+2-axis milling machine is less than that of the simultaneous 5-axis machine.



**Figure 1:** (a) Conventional 3-axis machining and (b) 3+2-axis machining using a shorter tool.

The computation of the numerical control (NC) data for the 3+2-axis machining is usually performed in two steps.

**Step 1 Determine the cutter position:** A ball-end cutter is generally employed in the 3+2-axis machining. The center point of the ball-end cutter is on the surface offset from the part surface by the cutter radius. At this point, the cutter posture is yet to be determined.

**Step 2 Determine the cutter posture:** The posture of the cutting tool in the milling process is then determined for each cutter position, such that a certain clearance is ensured between the tool and workpiece surface, to prevent collision between them.

When machining a part with complicated shapes, the cutter's posture must be changed several times to complete the 3+2-axis machining operation. Owing to inevitable positional errors of the milling machine, the minute level difference is known to be generated in the part where the surfaces machined by the tool of different postures connect. Therefore, to realize an optimal good surface finish, it is desirable to minimize the cutter posture changes.

The determination of the proper combination of cutter postures for the 3+2-axis machining is a huge burden for machining engineers. In this study, we propose a novel software technique for assisting the interactive determination of cutter postures for the 3+2-axis machining. To properly select the cutter posture, the recognition and visualization of the surface region in the part that can be machined using the cutter in the designated posture are important. To minimize the cutter posture changes, it is generally desirable to select the cutter posture, such that the tool can machine the surface area exhaustively. To assist the selection of such cutter postures, a function is required to visualize the difference in machinable area for each cutter posture. Proper visualization of the surface area left unmachined by the tool is also necessary because such area becomes the next machine target area with the tool in a different posture. When interactive use is considered, the processing speed of the software is required to be sufficiently fast, to avoid interference with the user's thinking.

In our software, a tool path representing the locus of the center point of a cutter is provided as input data. Instead of classifying the surface area of the part by the cutter posture, our software classifies points constituting the tool path. For each point in the path, a range of cutter postures without interferences with the machine part is computed. The information on the possible cutter postures for all machining points is "superimposed" in a Gauss map, and the number of machinable points can be examined for each cutter posture. The obtained results are color-coded in the display. By referring to the display, the cutter posture in which several points can be machined is easily identified. When a user selects a cutter posture based on the display, points machinable by the cutter in that posture are automatically selected from the path data. New tool path data are obtained by reconnecting the selected points. Simultaneously, the color display is updated based on the cutter posture information of the remaining points. By repeating the

selection of the cutter posture based on the color display, the operator can efficiently generate the NC data required for the 3+2-axis machining.

In the next section, previous studies on the software assistance of the 5-axis and 3+2-axis machining are briefly reviewed. In Section 3, we present our interactive method for determining cutter postures. Details on the proposed method are provided in Sections 4 and 5. In Section 4, we present our detection method for interference-free cutter postures. The graphic interface for assisting the interactive selection of the cutter posture is discussed in Section 5, while the experimental computation results are presented in Section 6. Finally, we summarize and present our conclusions in Section 7.

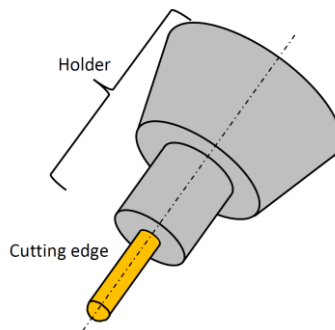
## 2 PRIOR STUDIES

In this section, we briefly discuss previous research results on the cutter posture determination in 5-axis machining. Several common subjects exist between the 5-axis machining and the 3+2-axis machining, and the research results for the 5-axis machining can be directly utilized for the automation of 3+2-axis machining.

In practice, 5-axis machining has been mostly used for manufacturing turbine blades and impellers. Several studies have been conducted to determine cutter postures specialized for machining these parts [1]. The basic NC-data computation method in 5-axis machining can be found in several studies, such as [2]. Most of the conventional studies on the cutter posture determination realize efficient computation by considering typical machining patterns [16], limiting the range of the posture based on the machinability of the tool [15], and determining the posture based on the geometric properties of the machining surface [3]. Although several commercial software packages that can generate NC data for 5-axis machining are available in the market, the technical details of the software are not published. Presently, these systems are limited in their utilization by the excessive amount of calculation time they incur.

Several research results are known to automatically determine the cutter posture without triggering collisions between the tool/holder and workpiece. Morishige *et al.* pioneered a determination algorithm for the collision-free cutter posture by adopting a two-dimensional configuration space [18, 19]. A trial-and-error-based method was proposed by Takeuchi *et al.* to compute collision-free cutter postures in 5-axis milling [23, 24]. Our research group developed a cutter posture calculation method using the depth buffer function and frame buffer objects [8]. Kang and Suh [13] employed a visibility cone to determine cutter accessibility in 5-axis milling. Spitz and Requicha developed a visibility cone computation method for a coordinate measurement machine using perspective projection and depth buffer [22]. Morimoto and Inui extended Spitz and Requicha's method for determining cutter accessibility in 3+2-axis machining [17].

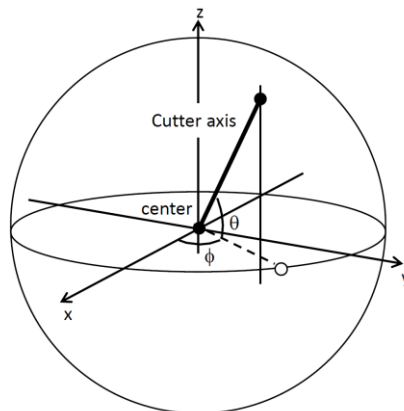
Recently, the parallel processing function of graphics processing unit (GPU) has been widely adopted as a technique for accelerating complicated geometric processing. For the GPU utilization in computer-aided manufacturing (CAM) software, known research results include the offset shape computation of polyhedral models [6], cutter location (CL) surface computation [4], cutter accessibility analysis [8, 12], Minkowski sum computation [14], and acceleration of the machining simulation [5]. In the field of 5-axis machining, another previous study realized milling simulation using a GPU [11]. In this research, a graphics library named OptiX was adopted to implement the software. The same library was used in this current study. In the CAM software, dextral modeling is often adopted as a shape representation method. The authors developed a conversion software for a polyhedral model into an equivalent dextral model using ray tracing (RT) cores of GPU [9]. The authors also realized technology which determines the tool posture without interfering with the part using the RT core [10]. This technology is utilized in this current study to accelerate computations.



**Figure 2:** Definition of cutter shape.

### 3 INTERACTIVE METHOD FOR DETERMINING CUTTER POSTURES

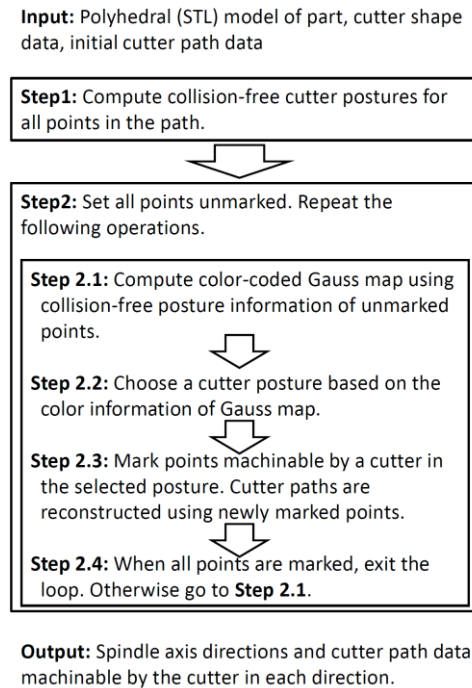
In this section, the outline of the proposed interactive method for determining cutter posture in the 3+2-axis machining is explained. In the following discussion, the milling operation with a ball-end cutter is assumed. A milling cutter comprises a cutting-edge part and cutter holder (Fig. 2). The shape of the holder can be considered a series of cylindrical shapes and/or truncated cone shapes coaxial with the cutting edge. In addition, the position of a ball-end cutter is usually represented by the center point of the spherical blade of the cutter. The cutter posture is given by the spindle axis direction of the cutter, which is specified by two rotational angles around two mutually perpendicular axes, A and B axes, of the milling machine. In this research, the azimuth  $\phi$  and elevation  $\theta$  angles in the world coordinate frame are adopted to define the cutter posture (Fig. 3); however, the posture can be specified based on the rotation around other axes, such as the A and B axes of a milling machine.



**Figure 3:** Definition of cutter posture using azimuth  $\phi$  and elevation  $\theta$  angles.

For the posture determination of the cutting edge, it is important to consider that the cutting edge is usually in contact with some parts of the surface for cutting and must not interfere with the other parts. In contrast, to determine the holder's posture, it is important that the holder and workpiece do not collide at any point during machining. The shape of the workpiece changes as machining progresses; hence, it is difficult to consider the effect of such shape change on the cutter posture at the time of NC data computation. Therefore, the determination of the holder posture is often performed, considering the collision avoidance between the holder and the part shape, and not the workpiece shape. Based on this concept, the cutter posture is determined in our study by considering the collision avoidance between the holder and the machine part with

sufficient space between them. In the future, we would like to realize the technology to determine the tool posture considering the change of the workpiece shape.

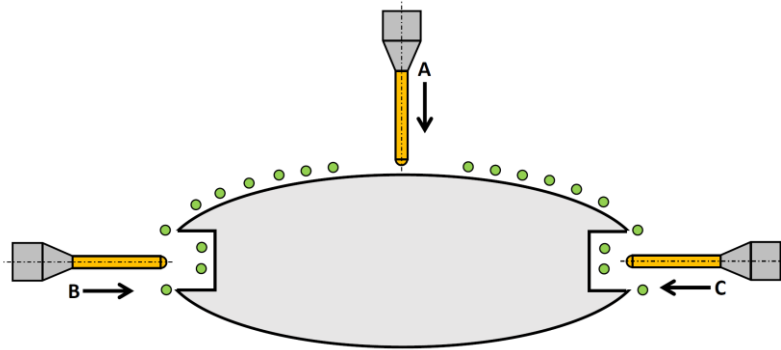


**Figure 4:** Processing flow for selecting the cutter postures interactively in 3+2-axis machining.

Figure 4 illustrates the flow of the selection process of the cutter posture. A polyhedral STL model of a machine part, shape data of a cutter, and path data representing the position change of the cutter are provided as input data to the software. The path data represent a series of coordinates of sufficiently close points. These data can be calculated using the conventional CAM software for 3-axis machining. Based on the given information, our software computes the range of collision-free cutter postures for each point in the path (Step 1). The cutter posture is then interactively determined in Step 2. Subsequently, our software determines the number of machinable points in the path for each cutter posture. The obtained results are color-coded and displayed on a Gauss map. When the user selects a cutter posture referring to the color information, the machinable point by the cutter in the posture is selected and path data are reconstructed using the machinable points. Considering the reduction of the tool posture changes, the spindle axis direction in which as many points as possible can be machined is generally chosen. The color information on the Gauss map is then updated based on the remaining points. This selection process for the cutter posture is repeated until all points in the cutter path become machinable. It finally outputs a combination of the spindle directions and cutter path data machinable by the cutter in each spindle direction. Although the software is slow in determining the available cutter postures for all points on the path, the subsequent interactive processing (Step 2) is sufficiently fast, and the user can select the required cutter postures with negligible awareness of latency.

Our software adopts color information on the Gauss map to guide the user to select a cutter posture that can machine as many points as possible. In general, this strategy can determine a feasible combination of cutter postures; however, the obtained results are not guaranteed to be the smallest set of cutter posture combinations. For this reason, we did not develop a system to automatically determine the tool postures based on this strategy. Such an example is illustrated in

Figure 5. This figure presents a section of a part, and the tool positions for machining its surface are depicted by green points. Although the maximum number of points can be machined by setting the cutter posture in the A-direction, further machining from the B- and C-directions is required to machine the remaining points, and a total of three directions are selected. However, if the B-direction can be selected first, all machining points can be machined by further selecting the C-direction.



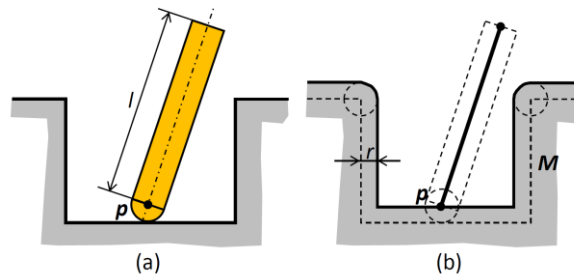
**Figure 5:** Section of a machine part and point representing the tool positions.

To obtain such a minimum combination, it is necessary to examine all combinations of the spindle axis directions in which all points can be machined. This problem is a typical set-covering problem and is known to be NP-complete. In the future, we hope to implement an algorithm that automatically and efficiently determines the optimal tool postures. We are currently researching various technologies required for the automation. This time, as one of such technologies, we have realized a technology to compute a range of tool postures capable of machining a point. Using this technique, we developed a color-coded display of the number of points that can be machined for each tool posture, and realized an interactive system for selecting tool postures by referring to this display.

#### 4 DETECTION OF CUTTER POSTURES WITHOUT INTERFERENCE

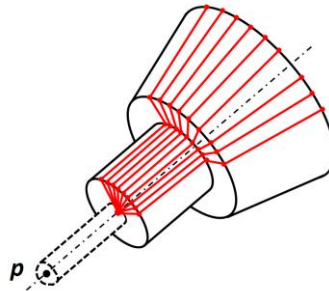
Our software first calculates the range of cutter postures that do not cause tool interference for each machining point. In this study, the cutter posture is considered a discrete combination of  $\phi$  and  $\theta$  angles. In total, 720  $\phi$  angles were considered in  $0.5^\circ$  intervals in the range of  $0-360^\circ$ , while 361  $\theta$  angles were considered in  $0.5^\circ$  intervals in the range from  $90^\circ$  to  $-90^\circ$ . Therefore,  $720 \times 361 = 259,920$  cutter postures are considered for each machining point, and the existence of the intersection of the cutter and machine-part shapes is checked and recorded for each posture of the machining point. We prepare a bit sequence 259,920 in length to record the cutter interference detection results for each machining point. Accordingly, "0" is assigned to a bit corresponding to the posture in which cutter interference occurs, and "1" is assigned for the posture bit without interference.

We divide the problem of the cutter interference detection into two sub-problems: interference detections of the cutting edge and holder part of the cutter. An offset shape  $\mathbf{M}$  is obtained by expanding the part shape by the cutter radius  $r$ . In a previous study, we developed a fast offsetting algorithm of a polyhedral object using the parallel processing function of GPU [7]. By adopting this algorithm, the polyhedral model of the offset shape can be obtained. The interference detection of the cutting edge whose center point is at point  $\mathbf{p}$  (Fig. 6(a)) is geometrically equivalent to the intersection detection of a line segment whose endpoint is at  $\mathbf{p}$  and a polyhedron  $\mathbf{M}$  (Fig. 6(b)).



**Figure 6:** Interference detection using offset shape  $M$ .

For each candidate direction, a straight line of length  $l$  is extended from point  $p$  to examine the intersection with  $M$ ; here,  $l$  denotes the value obtained by subtracting the cutter radius from the length of the cutting edge. The possible postures of the cutting edge are the directions in which the line segment and  $M$  do not intersect. Therefore, the posture determination problem of a cutting edge can be considered an intersection detection problem between a line segment and a set of polygons of the offset shape.



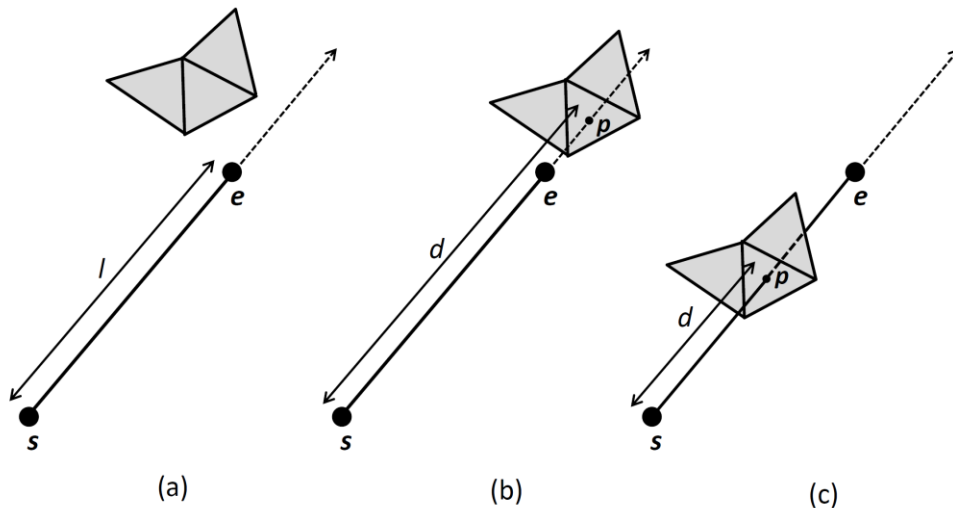
**Figure 7:** Holder representation using line segments.

For each candidate direction, a holder shape is oriented, such that the center axis of the holder is on the line originating from  $p$  in the cutter posture direction. To ensure sufficient clearance between the holder and the part, the cylinders and truncated cones, which constitute the holder shape, are defined to be slightly larger. A holder shape can be regarded as a swept surface obtained by rotating a series of line segments around the center axis of the cutter; therefore, we can represent the holder shape as a set of line segments (Fig. 7). The interference between a holder and a part shape can thus be ascertained by checking the intersection between line segments in the holder surface and polygons representing the part shape. Hence, the posture determination problem of a holder can be considered an intersection detection problem between a line segment and a set of polygons. A range of possible posture that can be taken by the cutting edge is obtained as a solution to the first problem. Another posture range of the holder is obtained as a solution to the second problem. Common solutions to both problems represent the appropriate postures of a cutter at point  $p$ .

Intersections between line segments and polygons can be computed at high speeds by employing the RT core hardware of GPU. The following is a summary of our algorithm represented in [10]. Current GPUs are equipped with a hardware called RT cores dedicated to an image processing technique called ray tracing in 3D computer graphics. In the ray tracing process, the computation of intersection points between half lines (rays) and polygons is frequently required. The RT core is a special hardware designed to speed up this process. We adopt this RT core technology to accelerate the intersection detection between a line segment and polygons.

NVIDIA Corporation provides an application programming interface (API) library called OptiX [21] that implements software for ray tracing. The function of the RT core is automatically available via its API functions. OptiX processing usually initiates the ray generation program. The starting point and direction of each ray are defined in this function and the ray tracing processing is initiated. By harnessing the parallel processing capability of GPU, thousands of threads for ray generation can be executed in parallel [20]. In the ray generation program, the `optixTrace()` function is called to initiate the ray tracing process. In this function, the closest hit or miss program is automatically invoked depending on whether the generated ray intersects the polygons.

- **Closest hit program:** This function is called when a ray collides with the nearest polygon during the ray tracing process. Information about the collision point is recorded using this function.
- **Miss program:** This function is called when a ray does not intersect with any polygons during the ray tracing process.



**Figure 8:** Intersection detection between a line segment and polygons using OptiX API.

To determine the possible posture of a cutter, it is important to detect the intersection between a line segment and a set of polygons. Let  $\mathbf{s}$  denote the coordinates of one endpoint of the line segment, and  $\mathbf{e}$ , the coordinates of the other end. The intersection detection of this line segment  $\mathbf{s}-\mathbf{e}$  and the polygon group can be realized using OptiX as follows. First, a ray generation program is employed to generate a ray whose starting point is at  $\mathbf{s}$  and whose direction is a vector from  $\mathbf{s}$  to  $\mathbf{e}$ . The collision between the ray and the polygon group is detected using the closest hit and miss programs as follows.

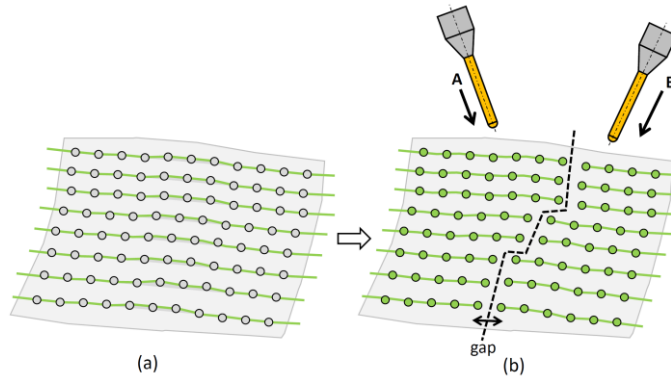
- If the miss program is invoked, the segments  $\mathbf{s}-\mathbf{e}$  and polygons do not intersect because the ray containing  $\mathbf{s}-\mathbf{e}$ , and the polygons do not collide (Fig. 8 (a)).
- If the closest hit program is invoked, the distance  $d$  between the obtained collision point  $\mathbf{p}$  and  $\mathbf{s}$  is examined. If  $d$  is longer than the segment length  $l$ , the segment and polygons do not intersect (Fig. 8(b)); otherwise, the segment and polygons exhibit an intersection (Fig. 8 (c)).

OptiX has a function for managing a group of polygons using acceleration structure. In this structure, polygons are recorded using the hierarchical bounding boxes. By adopting this structure, the collision detection between the ray and polygon groups can be further accelerated [21].





change the distance between the paths according to the cutter posture and the surface direction of the shape to be machined. Solving these issues is a direction for future research subjects.



**Figure 10:** Reconstruction of the tool path by connecting the points machinable using a cutter in the selected posture.

## 6 COMPUTATIONAL EXPERIENTS

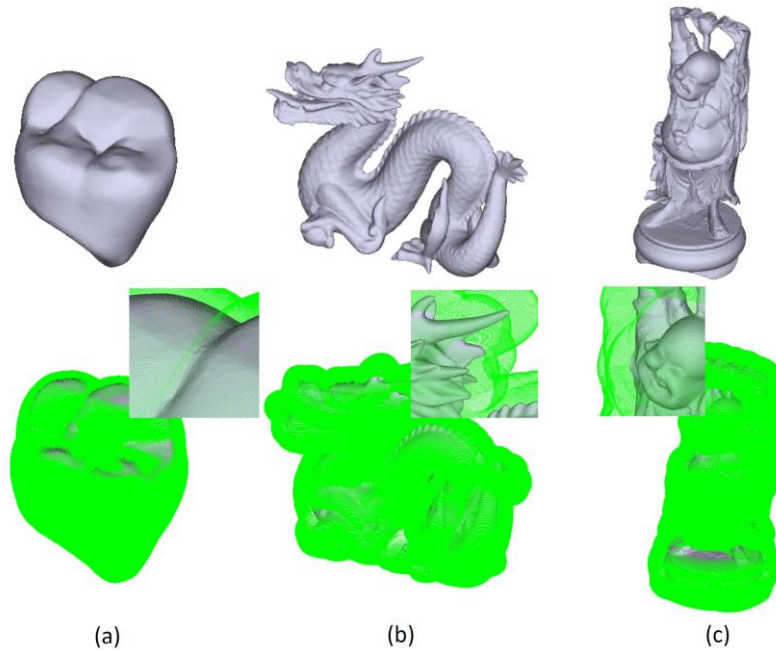
We implemented a software for assisting the cutter posture determination using the interactive aforementioned method. In the implementation, VisualStudio 2017, CUDA 10.2, and OptiX 7.2 were employed. A 64-bit PC with an Intel Core i9 Processor, 32-GB memory, and an nVIDIA GeForce RTX-3080 GPU was used in the experiments. Presently, we are developing a 5-axis machining CAM software for dental technicians, and in this case, a tooth model (Model A) was used as the machining object. We also prepared two small figure models (Models B and C) with complex shapes. Polyhedral models of the sample objects and the initial cutter paths for contouring the models are illustrated in Figure 11. Figure 12 presents the tool-shape data (ball-end cutter of 0.5-mm radius) used in our experiments. These data are fed into our system as input. In the current implementation, each cylinder and/or truncated cone of the holder shape were represented by 180 line segments covering its boundary. The number of polygons of the models and number of points in the paths are presented in Table 1.

	No. of polygons	No. of points in path	Offsetting time (s)	No. of polygons in offset shape	Time for computing collision-free postures (s)
Model A, tooth	15,000	102,841	5.22	1,266,656	1090.67
Model B, dragon	202,520	106,582	5.12	636,084	1207.53
Model C, happy buddha	67,240	80,064	4.66	769,220	914.47

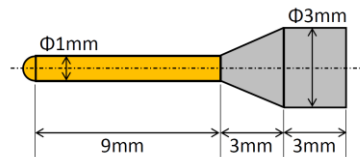
**Table 1:** Specifications of input data and necessary time for computing postures free from collisions for all points in the path.

The time required to use this system can be classified into three categories: (1) the time required to offset the part shape, (2) the time required to check cutter interferences for 259,920 cutter postures for each machining point, and (3) the time required to interactively generate cutter paths and update the color-coded display after the cutter posture selection. The time in (3) is usually 1 to 2 seconds for selecting each cutter posture, which is almost negligible. The time required to compute the offset shape of the models and time for detecting interference-free cutter postures are thus only presented in Table 1. In addition, 90% of the time required to determine the collision-free cutter postures is spent in detecting the tool postures without holder interference. It

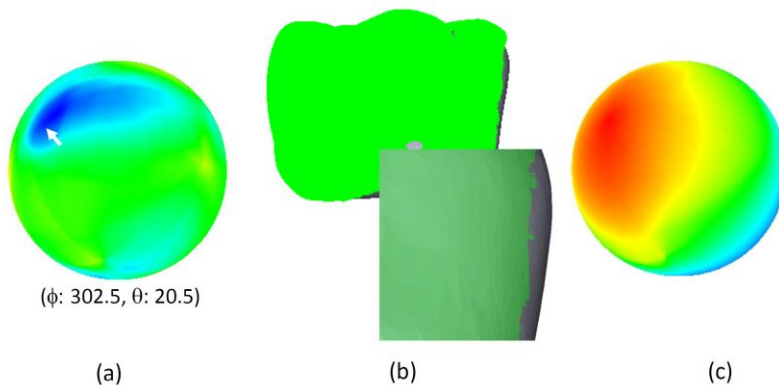
is considered that this is because the intersection detection of line segment and polygons is more often performed in the holder-interference avoidance.



**Figure 11:** Sample models and contour-type cutter paths: Models (a) A, (b) B, and (c) C.



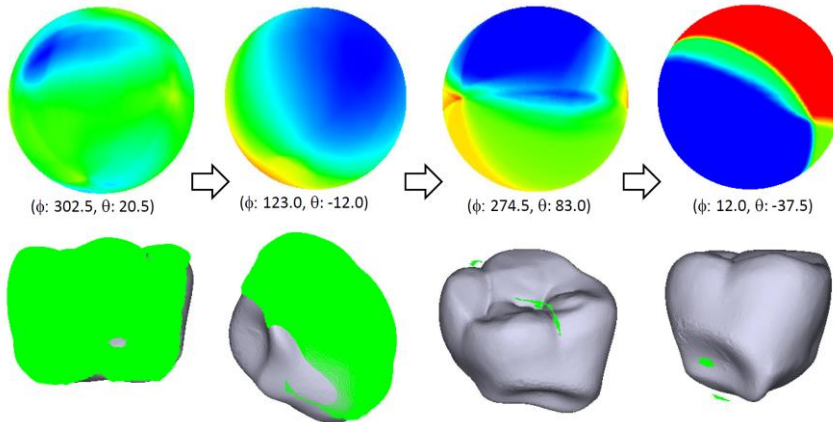
**Figure 12:** Shape data of ball-end cutter for milling the models.



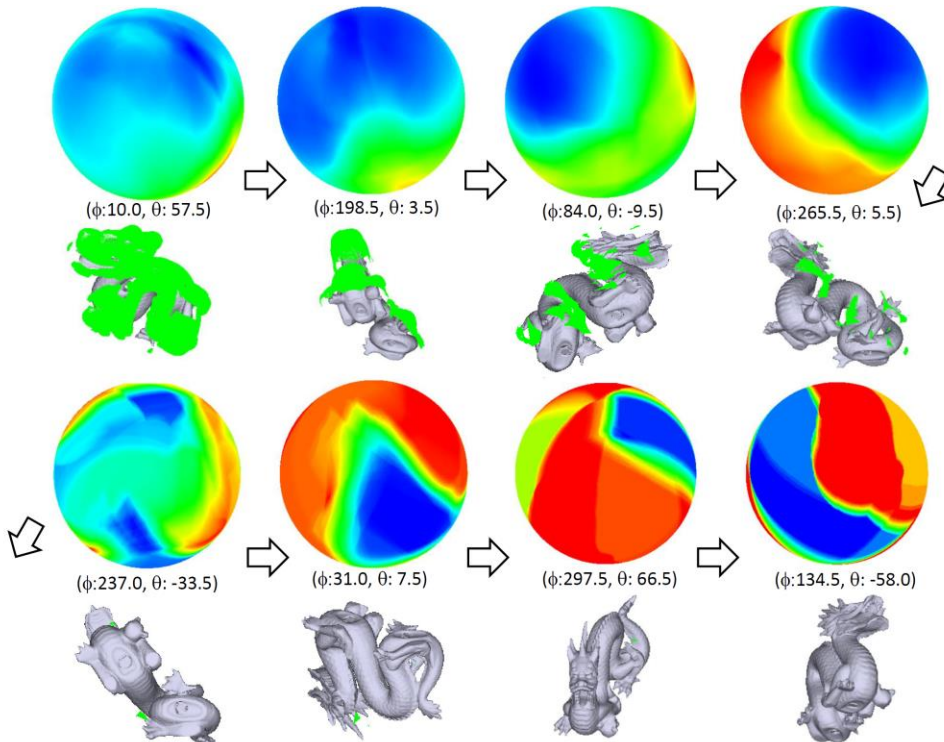
**Figure 13:** Snapshot of an interactive selection process of the cutter posture for Model A.

In Figure 13, a snapshot of an interactive selection process of the cutter posture is illustrated for Model A. Based on the computation results of the collision-free cutter postures, initial color-coded

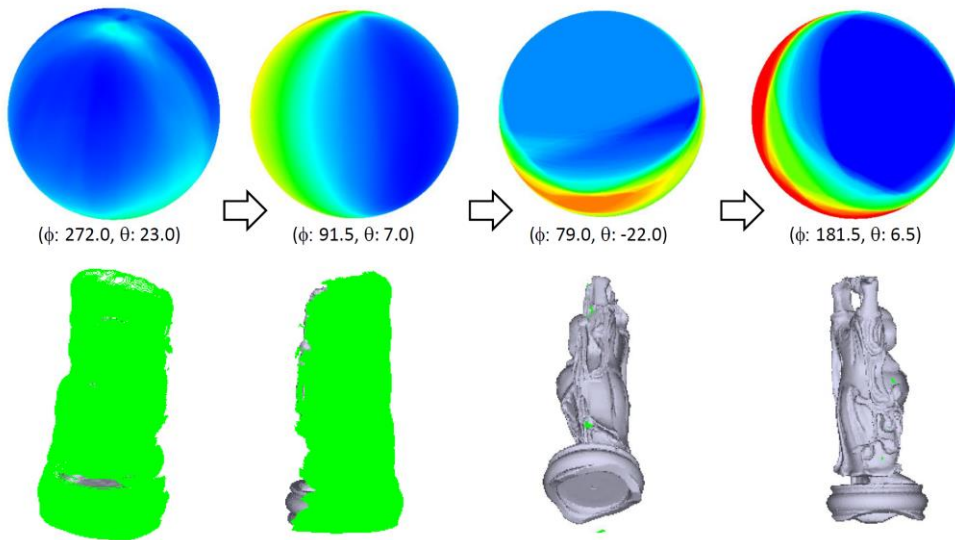
Gauss map is obtained, as illustrated in Fig. 13(a). The blue-colored region represents the cutter postures with several machinable points. When the bluest point in the Gauss map is selected by the mouse (white arrow in Fig. 13(a)), a contour-type cutter path using a cutter in the selected cutter posture ( $\phi = 302.5^\circ$ ,  $\theta = 20.5^\circ$ ) is reconstructed by properly connecting the machinable points (Fig. 13(b)). Simultaneously, the software recalculates the number of the machinable points for each cutter posture using the remaining points, and the color-coding results of the Gauss map are updated, as illustrated in Fig. 13(c).



**Figure 14:** Selection process of the cutter postures for Model A.



**Figure 15:** Selection process of the cutter postures for Model B.



**Figure 16:** Selection process of the cutter postures for Model C.

Fig. 14, 15, and 16 illustrate the selection processes of the cutter postures and cutter paths corresponding to each selected posture for all three models (Models A, B, and C). All the required cutter postures for 3+2-axis machining are 4, 8, and 4 for Models A, B, and C, as illustrated in Fig. 14, 15, and 16, respectively. Accordingly, we infer that the computation of the tool path for each selected tool posture and the update of the color information of the Gauss map after the selection were sufficiently fast, and a smooth interactive work was feasible. This time, we didn't have time to compare the performance of our system with other tool posture determination technologies. In the future, we will develop this technology to realize a system that automatically determines the tool postures. At that time, we plan to compare the necessary calculation time and results with other systems.

## 7 CONCLUSIONS

This paper describes a novel interactive software for assisting 3+2-axis machining. This software requires the polyhedral model of a machine part, shape data of a cutter, and path data representing the position change of the cutter. It computes a range of cutter postures without interferences with the machine part for each point in the path. The information on the possible cutter postures for all points were examined, and the number of machinable points was determined for each cutter posture. The obtained results were color-coded in the Gauss map. By referring to the color information of the Gauss map, cutter postures appropriate for the 3+2-axis machining of the part were efficiently selected, and the NC data for the machining operation were computed. We would like to provide this system to enterprises, conduct field tests, evaluate its effectiveness, and investigate necessary functional improvements. We also considered the development of the technology that can subdivide the part surface into several regions for each cutter posture and recalculate the tool path for each region. Development of a system for determining the optimal cutter postures for 3+2-axis machining without human intervention is another research target. Our proposed technology for determining all possible cutter postures for each point on the path is also effective in the automation of simultaneous 5-axis machining. Furthermore, the CAM software for simultaneous 5-axis machining based on this technology is considered our future research subject.

Masatomo Inui, <https://orcid.org/0000-0002-1496-7680>  
 Shutaro Taguchi, <https://orcid.org/0000-0003-2703-1731>  
 Nobuyuki Umezu, <https://orcid.org/0000-0002-7873-7833>

## REFERENCES

- [1] Bohez, E.L.J.; Senadhera, S.D.R.; Pole, K.; Duflou, J.R.; Tar, T.: A geometric modeling and five-axis machining algorithm for centrifugal impellers, *J. Manuf. Syst.*, 16(6), 1997, 422–436. [https://doi.org/10.1016/S0278-6125\(97\)81700-1](https://doi.org/10.1016/S0278-6125(97)81700-1).
- [2] Choi, B.K.; Jerard, R.B.: *Sculptured surface machining, theory and applications*, Kluwer Academic Publishers, Dordrecht, 1998.
- [3] Farouki, T.T.; Li, S.: Optimal tool orientation control for 5-axis CNC milling with ball-end cutters, *Comput Aided Geom Des*, 30, 2013, 226–239. <https://doi.org/10.1016/j.cagd.2012.11.003>.
- [4] Inui, M.: Fast inverse offset computation using polygon rendering hardware, *Comput. Aided Des.*, 35, 2003, 191–201. [https://doi.org/10.1016/S0010-4485\(02\)00052-0](https://doi.org/10.1016/S0010-4485(02)00052-0).
- [5] Inui, M.; Ohta, A.: Using a GPU to accelerate die and mold fabrication, *IEEE Comput. Graph. Appl.*, 27, January/February, 2007, 82–88. <https://doi.org/10.1109/MCG.2007.23>.
- [6] Inui, M.; Umezu, N.; Kitamura, Y.: Visualizing sphere-contacting areas on automobile parts for ECE inspection, *J. Comput. Des, Eng.*, 2(1), 2015, 55–66. <https://doi.org/10.1016/j.icde.2014.11.006>.
- [7] Inui, M.; Umezu, N.; Tsukahara, M.: Simple offset algorithm for generating workpiece solid model for milling simulation, *J. Adv. Mech. Des. Syst. Manuf.*, 11(4), 2017. <https://doi.org/10.1299/jamdsm.2017jamdsm0042>.
- [8] Inui, M.; Nishimiya, K.; Umezu, N.: Accessibility Map for Assisting Cutter Posture Determination in Five-Axis Mold Machining, 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), 2020, 432–437, <https://doi.org/10.1109/CASE48305.2020.9216975>.
- [9] Inui, M.; Kaba, K.; Umezu, N.: Fast dixelization of polyhedral models using ray-tracing cores of GPU, *Comput. Aided Des. & Appl.*, 18(4), 2021, 786–798. <https://doi.org/10.14733/cadaps.2021.786-798>.
- [10] Inui, M.; Kaba, K.; Umezu, N.: Fast Cutter Accessibility Analysis Using Ray Tracing Cores of GPU, *International Journal of Automation Technology*, 15(6), 2021, 842–851. <https://doi.org/10.20965/ijat.2021.p0842>.
- [11] Jachym, M.; Lavernhe, S.; Euzenat, C.; Tournier, C.: Effective NC machining simulation with OptiX ray tracing engine, *Vis. Comput.*, 35, 2019, 281–288.
- [12] Kaneko, J.; Horio, K.: Fast determination method of tool posture for 5-axis control machining using graphics hardware, *J. Japn. Soc. Precis. Eng.*, 72(8), 2006, 1012–1017, (in Japanese).
- [13] Kang, J.-K.; Suh, S.-H.: Machinability and set-up orientation for five-axis numerically controlled machining of free surfaces, *Int. J. Adv. Manuf. Tech.*, 13(5), 1997, 311–325.
- [14] Li, W.; McMains, S.: Voxelized Minkowski sum computation on the GPU with robust culling, *Comput. Aided Des.*, 43(10), 2011, 1270–1283. <https://doi.org/10.1016/j.cad.2011.06.022>.
- [15] Rao, M.; Ismail, F.; Bedi, S.: Tool path planning for five-axis machining using the principal axis method, *Int. J. Mach. Tools Manuf.*, 37(7), 1997, 1025–1040. [https://doi.org/10.1016/S0890-6955\(96\)00046-6](https://doi.org/10.1016/S0890-6955(96)00046-6).
- [16] Makhanov, S.S.: Adaptable geometric patterns for five-axis machining: a survey, *Int. J. Adv. Manuf. Technol.*, 47, 2010, 1167–1208. <https://doi.org/10.1007/s00170-009-2244-z>.
- [17] Morimoto, K.; Inui, M.: A GPU based algorithm for determining the optimal cutting direction in deep mold machining, *Proc. of IEEE Int. Symp. Assembly Manuf.*, 2007. <https://doi.org/10.1109/ISAM.2007.4288473>.
- [18] Morishige, K.; Takeuchi, Y.: 5-axis control rough cutting of an impeller with efficiency and accuracy, *Proc. 1997 IEEE Int. Conf. Robot. Autom.*, 1997, 1241–1247. <https://doi.org/10.1109/ROBOT.1997.614307>.

- [19] 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/10.1115/1.2830567>.
- [20] NVIDIA, CUDA C Programming Guide, 2018.
- [21] NVIDIA, OptiX™ ray tracing engine, <https://developer.nvidia.com/optix>
- [22] Spitz, S.N.; Spyridi, A.J.; Requicha, A.A.G.: Accessibility analysis for planning of dimensional inspection with coordinate measuring machines, *IEEE Trans. Robot. Autom.*, 15(4), 1999, 714–727. <https://doi.org/10.1109/70.782025>.
- [23] 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](https://doi.org/10.1016/S0007-8506(07)62028-9)
- [24] Takeuchi, Y.; Watanabe, T.: Generation of 5-axis control collision-free tool path and postprocessing for NC data, *CIRP Annals - Manuf. Tech.*, 41(1), 1992, 539–542. [https://doi.org/10.1016/S0007-8506\(07\)61263-3](https://doi.org/10.1016/S0007-8506(07)61263-3)