www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Visualization of 3+2 Axis Machining Result by Combining Multiple Z-map Models

Masatomo Inui[1], Qi Chen[2] and Nobuyuki Umezu[3]

[1]Ibaraki University, masatomo.inui.az@vc.ibaraki.ac.jp
[2]Ibaraki University, 20nm470t@vc.ibaraki.ac.jp
[3]Ibaraki University, nobuyuki.umezu.cs@vc.ibaraki.ac.jp

Corresponding author: Masatomo Inui, masatomo.inui.az@vc.ibaraki.ac.jp

Abstract. Geometric NC milling simulation is usually performed before the actual cutting process to visualize the machining result and detect potential errors during the machining process. Recently, 3+2 axis machining, in which machining is performed by tilting the direction of spindle of the tool, has been used increasingly in the mold and die fabrication. In this paper, a novel method to visualize the milling result of the 3+2 axis machining operations is proposed by combining multiple Z-map models into a single triple-dexel model. This method can correctly visualize the result of 3+2 axis machining operations at high speed by using the rapidity of the Z-map based milling simulation and the expressiveness of the triple-dexel model. The effectiveness of the technology is demonstrated using software and via computational experiments.

Keywords: Model conversion, triple-dexel modeling, milling simulation.
DOI: https://doi.org/10.14733/cadaps.2022.825-837

1 INTRODUCTION

Geometric NC milling simulations are generally performed before the actual cutting process to visualize the machining result and detect potential errors such as cutter gouges in the machining process. A milling operation is geometrically equivalent to a Boolean subtraction of the swept volume of a cutter moving along a path from a solid model representing the stock shape. The NC cutter path for machining a curved surface is a series of short line segments. The swept volume of the cutter is defined for each line segment, and its subtracting operation from the workpiece model is repeated in the milling simulation. Because the cutter path for milling a complex surface often is several tens of meters long, tremendous times of the subtracting operations are necessary for the simulation. Because Boolean set operation such as the subtraction is a significantly complicated process, numerous repetitions often affect the computation time and stability of the processing.

The milling simulation method based on the Z-map representation of the workpiece shape is most commonly used because of its implementation ease, speed, and robustness. In this method, the workpiece shape is represented as a collection of vertical line segments, which are extended

from each grid point of a square mesh placed on a horizontal reference plane (Figure 1(a)). For each linear motion of the cutter along the path, the intersection between the segments and the cutter swept volume is computed and removed from the workpiece model. This process is iterated for all the cutter motions, and the final shape of the workpiece is obtained. Z-map based NC milling simulation can be accelerated using the depth buffer function of the graphics processing unit (GPU) [9, 10, 12]. Using the latest GPU, it is possible to visualize the result of complicated machining within dozens of seconds. To represent the machining results accurately, the reference plane for the Z-map must be set perpendicular to the spindle direction of the cutter. Therefore, machining simulation using the Z-map is basically performed only for 3-axis machining, in which the spindle direction is fixed in the z-axis direction.

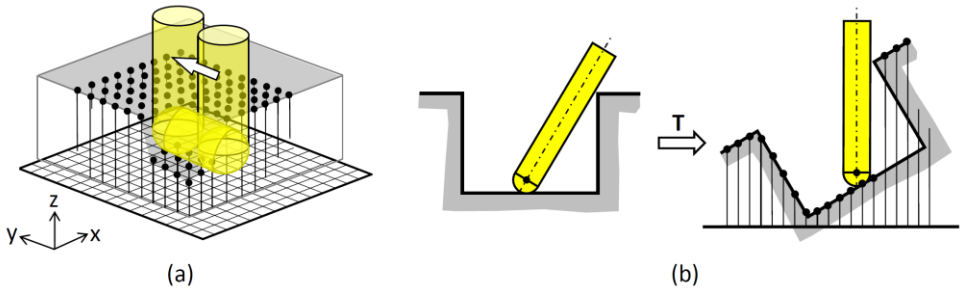


Figure 1: (a) Z-map based milling simulation. (b) 3+2 axis machining simulation using Z-map.

Recently, the use of 3+2 axis machining, in which machining is performed by tilting the direction of the tool spindle, has increased in the mold and die fabrication. In the 3+2 axis machining, the tool length can be shortened by properly setting the spindle direction. Furthermore, stable machining with comparatively less tool deformation than that of the conventional 3-axis machining is possible. This type of machining enables the minimization of the number of mounting changes of the workpiece. Using this method, both the machining accuracy and machining cost can be simultaneously improved. When the direction of the spindle is limited to one direction, 3+2 axis machining can be geometrically simulated using a Z-map by applying a coordinate transformation **T** to the workpiece shape and the cutter path so that the spindle direction of the cutter is parallel to the z-axis (Figure 1(b)). After the simulation, the resultant shape is transformed back to the original coordinate frame using $\mathbf{T}^{-1}$ (the inverse transformation of **T**). When machining a complex mold, the milling operation can be performed by switching multiple spindle directions, which cannot be handled by a single Z-map model. Therefore, in the geometric simulation of the 3+2 axis machining, a time-consuming simulation technique using B-reps, voxels, or triple-dexels is generally used. At present, the simulation method which satisfies all of speed, accuracy, and stability of processing has not been realized for the 3+2 axis machining.

We propose a novel method to visualize the result of the 3+2 axis machining operations. The proposed method decomposes a complex 3+2 axis machining operation into a set of single-spindle-direction-machining operations. Each machining operation is simulated using a Z-map. Subsequently, all Z-map models representing the machining results for each spindle direction are combined into one solid model, which represents the total machining result. In our software, a triple-dexel model was used to represent the resultant shape. This simulation method can correctly visualize the 3+2 axis machining result at a high-speed using the rapidity of the Z-map based milling simulation and the expressiveness of the triple-dexel model. The effectiveness of the technology was verified using software and by performing computational experiments. In Section 2, the existing studies on milling simulation are briefly reviewed. In Section 3, the basic concept of our conversion algorithm from multiple Z-maps to one triple-dexel model is illustrated. The details of the algorithm are explained in Section 4. Experimental computation results are presented in Section 5. Finally, we summarize our conclusions in Section 6.

2 BACKGROUND

The existing literature on milling simulation can be classified into two groups, namely, geometric milling simulation and cutting force simulation in the milling process. Because the result of the geometric simulation is used as the basic data in the cutting force analysis [23], geometric milling simulation can be considered fundamental to any milling simulation technology. The main objective of the geometric milling simulation is to visualize the workpiece shape change based on the milling operation. A majority of systems realize the simulation based on the Boolean subtraction of the cutter swept volume from the workpiece shape model. One of the major differences among the systems is the representation scheme of the workpiece shape. CSG, B-reps, Z-maps, voxels, dexels, triple-dexels, octree, and vector representations have been used in the exiting studies [2-4, 6, 8, 16, 18, 21, 22, 25, 26]. The use of the parallel processing capability of a GPU for accelerating the simulation has also been studied, for example, in [9, 10, 12].

Recently, the focus of machining simulation research has shifted to the simulation of 3+2 axis machining and simultaneous 5-axis machining. Gong and Feng developed a simulation method using the cutter swept volume and cutter removal volume in the triangle mesh representation [7]. Erdim and Sullivan realized milling simulation software using the composite adaptively sampled distance field [5]. In the power skiving of gear machining, the cutter exhibits a significantly complex motion. Tapoglou developed a cutting simulation software of power skiving using a B-reps solid modeling system [24]. We have developed a geometric simulation system of the 5-axis machining using workpiece model in the triple-dexel representation [11]. We applied the simulation system to the power skiving also [13]. One study realized the 5-axis machining simulation using GPU [15]. Although the future developments in the field can be expected, the existing simulation method needs to be improved.

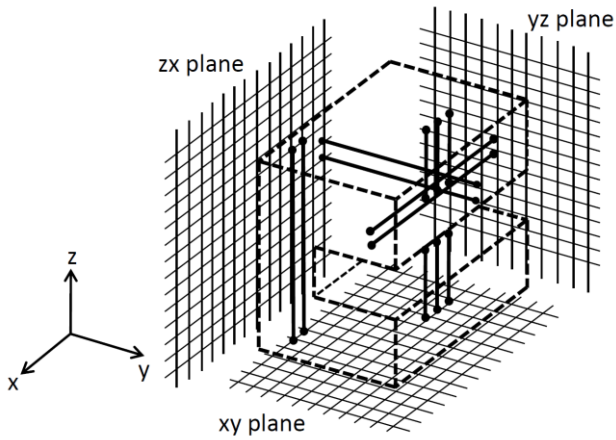


Figure 2: Triple-dexel model.

We use the Boolean set operation of solid models when combining several machining simulation results. To perform the set operation stably and at a high speed, the triple-dexel (triple rays) model [1] is used as a shape representation method of the simulation result. Dexel modeling is known as an extension of the Z-map. In this method, the object shape is represented using a bundle of z-axis-aligned segments defined for each grid point of a square mesh in the xy plane [25]. With dexel modeling, near-vertical surfaces are prone to inevitable large shape errors caused by a finite grid resolution. The triple-dexel model was proposed to overcome this non-uniformity of the representation accuracy. In this representation, the solid shape is not only defined by a z-axis-aligned dexel model but also by an x-axis-aligned dexel model based on a square mesh in the yz plane and a y-axis-aligned dexel model based on a mesh in the zx plane (Figure 2). Because a triple-dexel model defined based on properly aligned square meshes in the xy, yz, and zx planes is

equivalent to a voxel model, it can be easily converted to a polyhedral model using boundary evaluation techniques such as Marching Cubes algorithm [17].

3 BASIC IDEA

In this paper, we propose a simple and fast milling simulation method for 3+2 axis machining. This method is specialized for visualizing the machining result. Currently, it is not suitable for handling the dynamic shape changes to the workpiece in the machining process, for example, animation display of the machining process.

3.1 Outline of the Algorithm

Figure 3 illustrates the outline of the algorithm. The input data of the algorithm consist of a polyhedral STL model representing the initial shape of the workpiece and a set of cutter path data and their corresponding cutter data for 3+2 axis machining. The data of each cutter path are defined for one cutter with a specific spindle direction. The output of the algorithm is a solid model representing the resultant shape after milling. In our implementation, the triple-dexel model is used for representing the resultant shape; however, our method is not limited to this model representation and can be applied to any solid shape representation method.

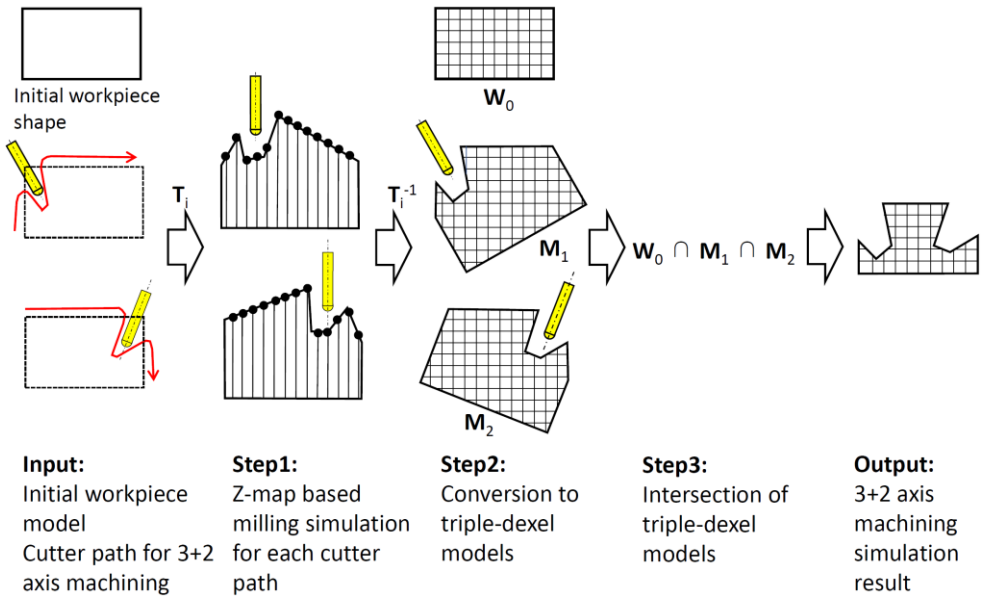


Figure 3: Outline of the algorithm.

Consider a single 3+2 axis machining operation with a cutter in a certain spindle direction. By performing a proper coordinate transformation $\mathbf{T}_i$ on the initial workpiece model and each cutter path, and by executing Z-map based milling simulation with the transformed data, a machined shape in the Z-map representation is obtained, as mentioned in Step1 in Figure 3. The machined shape is converted to an equivalent polyhedral model. The polyhedral model is transformed into the original coordinate frame by applying $\mathbf{T}_i^{-1}$ on it. The model is then converted to a triple-dexel model (Step2). When 3+2 axis machining is performed for multiple spindle directions, the same number of machined shapes can be obtained by repeating the above-described processing for each spindle direction. A resultant workpiece model reflecting all 3+2 machining operations is finally obtained by combining multiple machined shapes using the Boolean intersection operation (Step3). In this method, the simulation time can be reduced significantly, because a simple conversion of

the polyhedral model into a triple-dexel model and several Boolean intersection operations of dexel models are necessary only after high-speed milling simulation using the Z-map.

3.2 3+2 Axis Machining Simulation Using Z-map

Consider N cutter path data and the corresponding specifications of the cutter and spindle directions. Using these data, we can obtain N cutter swept volumes $\{\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \ldots, \mathbf{V}_N\}$. The workpiece shape **R** representing the machining result is obtained by the following Boolean subtraction operations:

$$\mathbf{W}_0 - \mathbf{V}_1 - \mathbf{V}_2 - \mathbf{V}_2 - \ldots - \mathbf{V}_N = \mathbf{R} \tag{3.2.1}$$

where W_0 represents the initial workpiece shape.

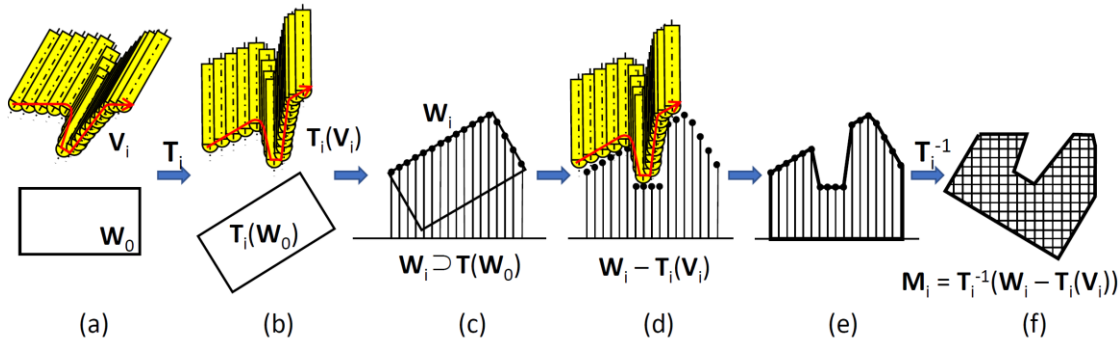


Figure 4: Model conversion process.

The same result can be obtained using the Z-map based milling simulation. Figure 4(a) illustrates an initial workpiece model $\mathbf{W}_0$ and a swept volume $\mathbf{V}_i$ of an inclined cutter moving along a given cutter path. By applying a coordinate transformation $\mathbf{T}_i$, which sets the spindle direction of the cutter parallel to the z-axis, we obtain $\mathbf{T}_i(\mathbf{W}_0)$ and $\mathbf{T}_i(\mathbf{V}_i)$, as depicted in Figure 4(b). $\mathbf{T}_i(\mathbf{W}_0)$ is then converted to a Z-map model $\mathbf{W}_i$. We prepare a horizontal reference plane sufficiently below $\mathbf{T}_i(\mathbf{W}_0)$ and place a square mesh on it. For each grid point of the mesh, a vertical line segment is defined. The height of the top surface of the workpiece model $\mathbf{T}_i(\mathbf{W}_0)$ at the grid point is set as the z coordinate of the top point of the segment. The height of the reference plane is used as the z coordinate of the bottom side point of the segment so that the obtained Z-map model $\mathbf{W}_i$ satisfies $\mathbf{W}_i \supset \mathbf{T}(\mathbf{W}_0)$, as depicted in Figure 4(c).

A Z-map based milling simulation is then executed using $\mathbf{W}_i$ and $\mathbf{T}_i(\mathbf{V}_i)$ and the resultant Z-map model $\mathbf{W}_i - \mathbf{T}_i(\mathbf{V}_i)$ is obtained (Figure 4(d)). This model is converted to a polyhedral model using the method described in the following section (Figure 4(e)). By applying $\mathbf{T}_i^{-1}$ to the model and by further converting the transformed polyhedral model to a triple-dexel model, a model $\mathbf{M}_i$ equivalent to $\mathbf{T}_i^{-1}(\mathbf{W}_i - \mathbf{T}_i(\mathbf{V}_i))$ is obtained (Figure 4(f)). Using N set of the cutter path, cutter shape, and spindle direction specifications, N Z-map based milling simulations are executed. Consequently, we obtain N Z-map models representing the machined shape. They are converted to equivalent triple-dexel models $\{\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \ldots, \mathbf{M}_N\}$ by using the algorithm mentioned above.

Consider the following Boolean intersection computations:

$$\mathbf{W}_0 \cap \mathbf{M}_1 \cap \mathbf{M}_2 \cap \mathbf{M}_3 \cap \ldots \cap \mathbf{M}_N \tag{3.2.2}$$

Because $\mathbf{M}_i = \mathbf{T}_i^{-1}(\mathbf{W}_i - \mathbf{T}_i(\mathbf{V}_i))$, we can derive

$$= \mathbf{W}_0 \cap (\mathbf{T}_1^{-1}(\mathbf{W}_1 - \mathbf{T}_1(\mathbf{V}_1))) \cap (\mathbf{T}_2^{-1}(\mathbf{W}_2 - \mathbf{T}_2(\mathbf{V}_2))) \cap (\mathbf{T}_3^{-1}(\mathbf{W}_3 - \mathbf{T}_3(\mathbf{V}_3))) \cap \ldots \cap (\mathbf{T}_N^{-1}(\mathbf{W}_N - \mathbf{T}_N(\mathbf{V}_N)))$$

$$= \mathbf{W}_0 \cap (\mathbf{T}_1^{-1}(\mathbf{W}_1) - \mathbf{V}_1) \cap (\mathbf{T}_2^{-1}(\mathbf{W}_2) - \mathbf{V}_2) \cap (\mathbf{T}_3^{-1}(\mathbf{W}_3) - \mathbf{V}_3) \cap \ldots \cap (\mathbf{T}_N^{-1}(\mathbf{W}_N) - \mathbf{V}_N) \quad (3.2.3)$$

Because $\mathbf{T}_i^{-1}(\mathbf{W}_i) \supset \mathbf{W}_0$, the equation can be modified to

$$= \mathbf{W}_0 - \mathbf{V}_1 - \mathbf{V}_2 - \mathbf{V}_2 - \ldots - \mathbf{V}_N \quad (3.2.4)$$

and the milling simulation result **R** is derived. In the actual implementation, for each spindle direction, Z-map based milling simulations with multiple cutters are collectively executed, and the resultant Z-map model is converted into a triple-dexel model.

4 CONVERSION FROM MILLING RESULT Z-MAP TO TRIPLE-DEXEL MODEL

To realize the simulation method, the following three steps are necessary:

- Z-map based milling simulation (Step1).
- Conversion from a Z-map model to a triple-dexel model (Step2).
- Boolean intersection computation of triple-dexel models (Step3).

We used our developed Z-map based milling simulation software [10] in this study. This system can visualize the milling result in a short time using the depth buffer function of the GPU. Because the calculation of the Boolean intersection of two triple-dexel models can be decomposed to a set of simple Boolean intersection calculations of segments on the same line, its implementation is straightforward. Therefore, in the following, only the method for converting the machined shape model in the Z-map representation to a triple-dexel model is explained.

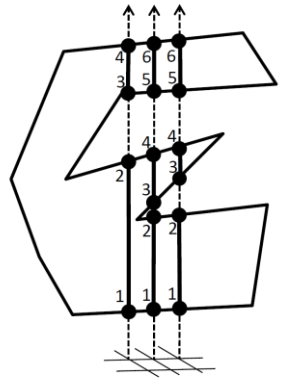


Figure 5: Ray algorithm for converting a polyhedron into a dexel model.

4.1 Polyhedrization of Z-map Model

A Z-map model can be recognized as a set of vertical line segments. It is difficult to convert such a shape representation directly into a triple-dexel model. A closed polyhedron can be easily converted into an equivalent triple-dexel model using the ray algorithm [19]. Therefore, we first convert a Z-map model into a closed polyhedron. Then we further convert it into a triple-dexel

model using the ray algorithm. Before the conversion to the triple-dexel model, a coordinate transformation $\mathbf{T}_i^{-1}$ is applied to move the polyhedron to the original coordinate frame. In the ray algorithm, a square mesh in the xy plane is prepared. Corresponding to each grid point, a ray perpendicular to the xy plane is extended along the z-axis, and its points of intersection with the surface polygons of the object are computed, as depicted in Figure 5. These intersection points are sorted along the direction of the ray. By connecting odd-numbered intersection points with even-numbered intersection points using lines, a set of z-axis-aligned dexel model equivalent to the polyhedral model is obtained. Other x-axis-aligned dexel model and y-axis-aligned dexel model are obtained by repeating the above process using square meshes on the yz plane or the zx plane.

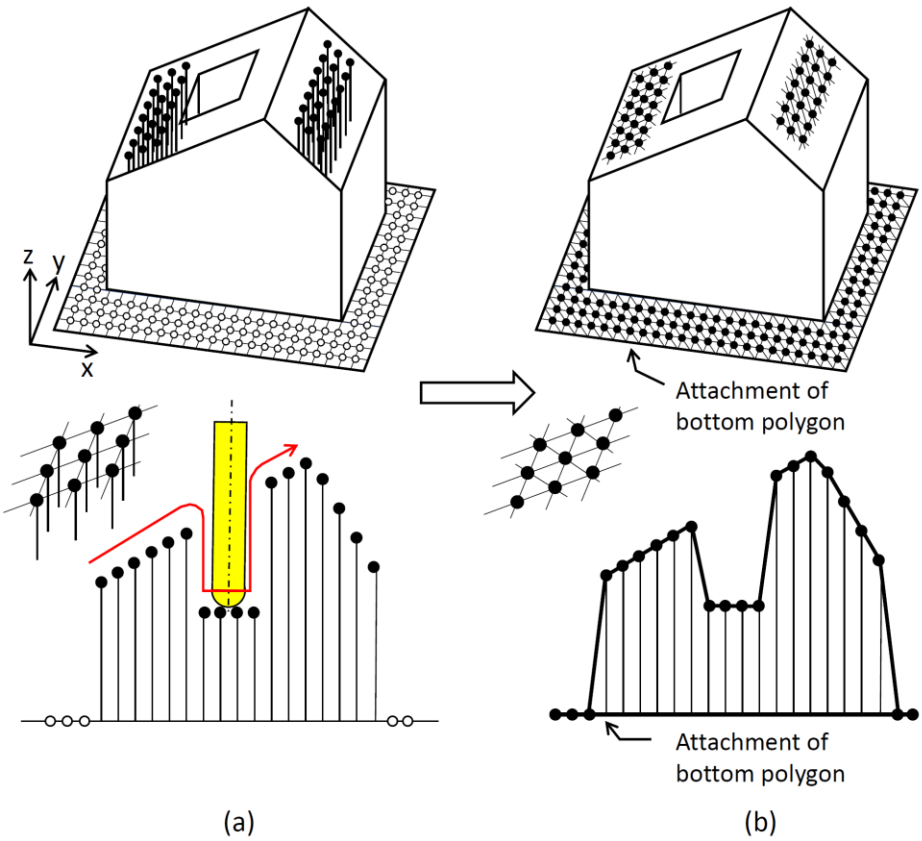


Figure 6: Model conversion process from a Z-map model to a closed polyhedron.

Figure 6(a) illustrates a Z-map based milling simulation result using a workpiece model and a cutter path. These data were obtained by applying transformation $\mathbf{T}_i$ to the original workpiece model and cutter path. Before the simulation, transformed workpiece model was converted into a Z-map model using the method explained in Section 3.2. Z-map model of the workpiece was defined based on a square mesh given in the horizontal reference plane. The square mesh was aligned with respect to the x- and y-axis of the world coordinate frame. The size and position of the mesh were determined so that its boundary completely contained the projection of the workpiece model to the reference plane within. In the milling simulation using the transformed cutter path and vertical cutter, the z coordinate of the top side points of the vertical Z-map segments were modified and a milling result shape was obtained.

The top surface of a Z-map model can be recognized as a group of points arranged according to a two-dimensional (2D) grid structure. Z-map model thus can be converted into a polygon mesh

by appropriately connecting the adjacent points. Since the polygon mesh obtained in this method is an open shape, the ray algorithm cannot be applied as it is. To realize the conversion, we developed a method for obtaining a closed polygon mesh based on a Z-map model. When the square mesh for defining the Z-map is given sufficiently wide on the reference plane, there exist grid points without height information (white points in Figure 6(a)) around the grid points with proper height information (black points) representing the machining result shape. By giving a sufficiently small height value to the white grid points, it is possible to define three-dimensional (3D) points for all grid points and define a polygon mesh that completely covers the workpiece shape by properly connecting the points. A closed polyhedron is finally obtained by attaching a rectangle of the same size as that of the mesh in the bottom side of the Z-map, as depicted in Figure 6(b). An inverse transformation $\mathbf{T}_i^{-1}$ is applied to the obtained closed polyhedral model, and a triple-dexel model representing the machined shape is computed by applying the ray algorithm to the model.

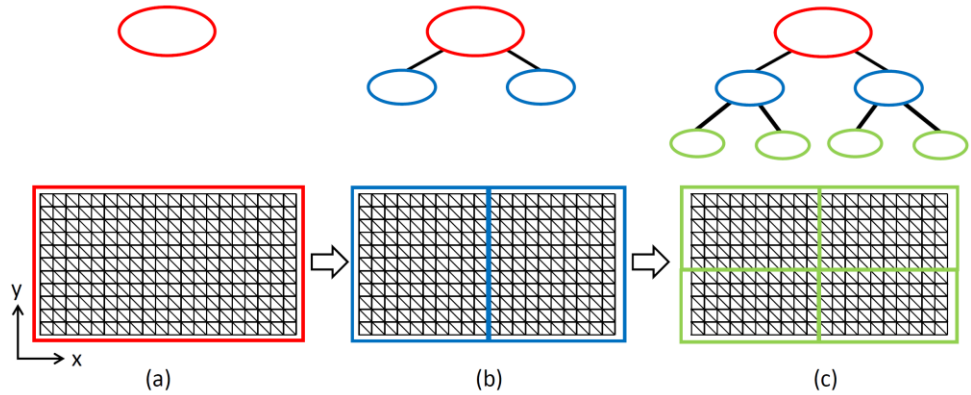


Figure 7: Hierarchical bounding boxes for managing surface polygons of the machined shape.

4.2 Use of Hierarchical Bounding Boxes

The most critical process in our algorithm is to convert a polyhedral model representing the machined shape into a triple-dexel model. In the ray algorithm, the intersection calculation between a straight line and polygons is executed repeatedly. The Z-map model used in the milling simulation is usually defined based on a high-resolution square mesh of approximately 5,000 × 5,000 resolution. When this Z-map is converted into a polyhedron using the above-mentioned algorithm, the number of polygons exceeds fifty million. To efficiently compute intersection points between so many polygons and a straight line, polygons are recorded and managed by a hierarchical bounding box structure in our software.

Polygons obtained by the conversion from a Z-map are defined based on a 2D square mesh. For each square cell, two triangular polygons are defined (see Figure 6(b)). First, we define an axis-aligned bounding box (AABB) [20], which contains polygons of the whole mesh within and make it the root node of a tree structure, as depicted in Figure 7(a). Although a 2D rectangle is drawn in the figure, the rectangular parallelepiped including the polygon group is actually defined. We then divide the mesh into two sub-meshes of equal size in the direction of the x-axis. For each sub-mesh, an AABB is defined so that it contains polygons relating to the sub-mesh. Two obtained AABBs are assigned as child nodes of the root node (Figure 7(b)). A mesh corresponding to each child AABB is further divided into two sub-meshes of equal size in the direction of the y-axis. Two AABBs are defined and registered as child nodes of the node corresponding to the mesh before the subdivision (Figure 7(c)). The mesh dividing operation and child AABB defining process are iterated in the directions of x- and y-axis alternately, and a binary AABB tree structure is defined. In the current implementation, this process is terminated when the mesh size becomes 4 x 4 or less.

An efficient selection of triangles having potential intersections with a straight line can be achieved by traversing the AABB tree from the root node to the leaf nodes in a depth-first manner. At each tree node in the traversal, the intersection between the line and the AABB corresponding to the node is checked. The child node of the node is followed if the line has an intersection with the box; otherwise, further traversal from the node is canceled. When the leaf node of the AABB tree is reached, the intersection points between the line and the triangles recorded in association with the box at the node are computed and recorded. After the tree traversal, the obtained intersection points are sorted along the line direction to obtain the dexels on the line.

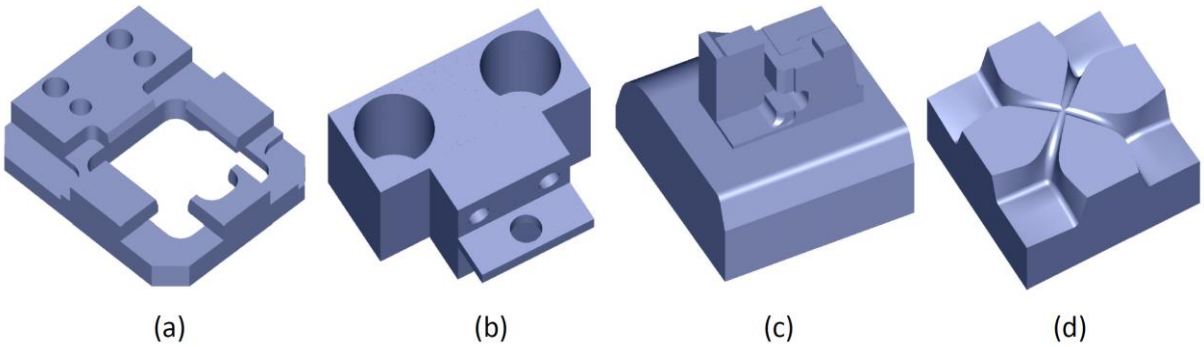


Figure 8: Four sample models used in the cutter path computation for 3+2 axis machining: (a) model A (29.5 × 24.0 × 6.0 mm), (b) model B (30.0 × 26.0 × 17.0 mm), (c) model C (51.0 × 45.0 × 43.0 mm), and (d) model D (130.0 × 130.0 × 80.0 mm).

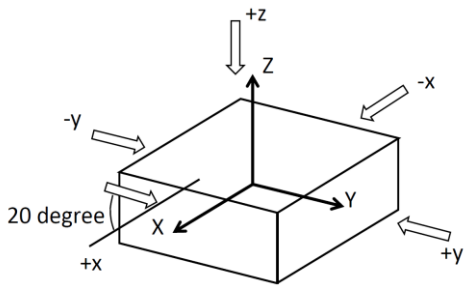


Figure 9: Five spindle directions for 3+2 axis machining used in our experiments.

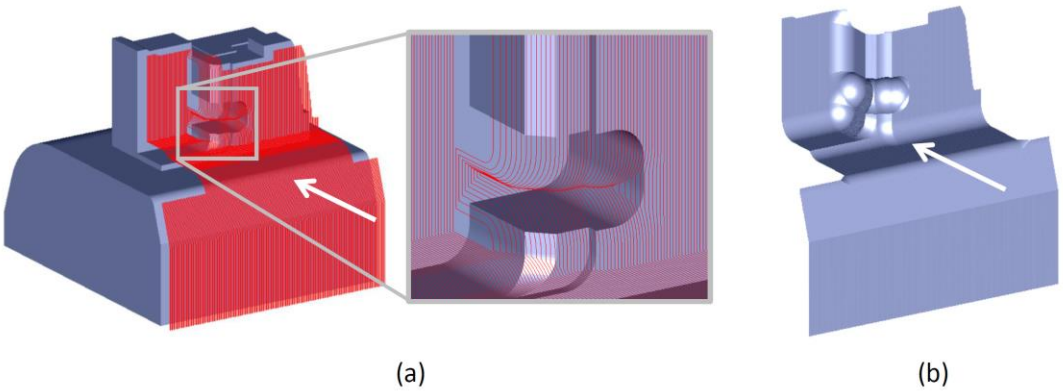
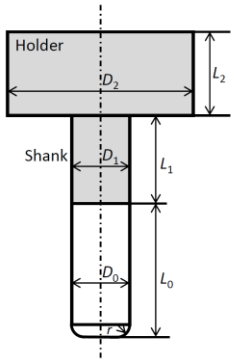


Figure 10: (a) Cutter path computation example using our CAM software for model C with a horizontal ball-end cutter. (b) Simulation result of a machined surface obtained using our Z-map

based milling simulator. White arrows in the figures represent the direction of the spindle of the cutter.

5 COMPUTATIONAL EXPERIMENTS

Using the algorithm, we implemented software for visualizing the 3+2 axis machining result. VisualStudio 2017 and OpenGL were used in the implementation. The effectiveness of the software was verified by performing computational experiments. A 64-bit PC with Intel Core i7 Processor, 32 GB memory, and nVIDIA GeForce RTX-2080 SUPER GPU was used. Figure. 8 illustrates four machine part CAD models A, B, C, and D used in our experiments. Cutter paths for 3+2 axis machining these parts were computed. In the cutter path computation and Z-map based milling simulation, our custom developed CAM software [10] was used. In the path computation, we specified 5 spindle directions, namely, +z, +x, +y, -x, and –y, as depicted in Figure 9. For directions +y, -x, and -y, cutters were set horizontal. For spindle direction +x, the cutters were tilted 20 degrees from the horizontal. Figure 10(a) illustrates an example of the computed cutter path for milling part C using a horizontal ball-end cutter in the -y direction. Figure 10(b) illustrates a final machined surface obtained using our Z-map based milling simulator. For each spindle direction, we specified 4 cutters (1 flat-end cutter and 3 ball-end cutters). Figure 11 lists the specification of the cutters used in the path generation and milling simulation.



Cutter type | D_0 | L_0 | r | D_1 | L_1 | D_2 | L_2 (*mm*)

Cutter type	D_0	L_0	r	D_1	L_1	D_2	L_2
Flat	12.0	24.0	0.0	12.0	30.0	25.0	50.0
Ball	6.0	10.0	3.0	6.0	40.0	25.0	50.0
Ball	2.0	5.0	1.0	2.0	10.0	25.0	50.0
Ball	0.5	4.0	0.25	0.5	5.0	25.0	50.0

Figure 11: Specification of the cutters used in path computation and simulation.

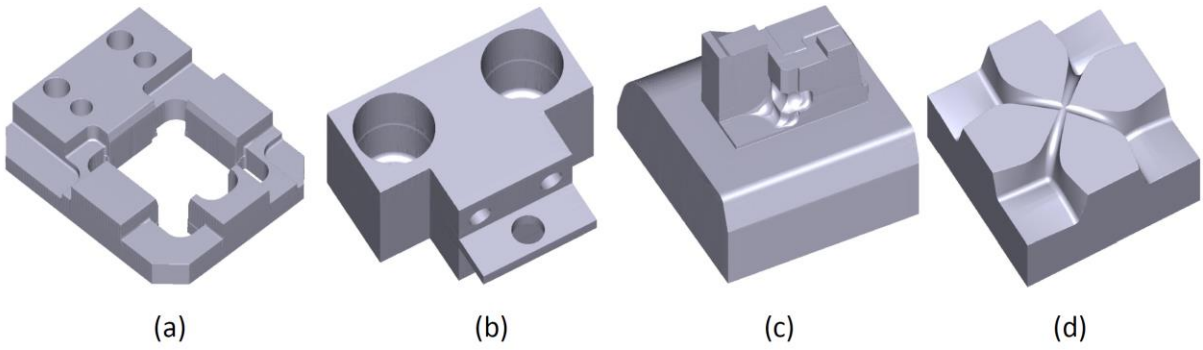


Figure 12: Simulation results of four models.

For each model, a block model of a slightly larger size was defined as its initial workpiece model. Z-map based milling simulations using four types of cutters were performed for each spindle direction. In the milling simulation, Z-map model based on a 5,000 × 5,000 resolution square

mesh was used (resolution varied depending on the model size and spindle direction). The obtained Z-map model representing the machining result was converted into a triple-dexel model. These processes were iterated for five spindle directions, and five triple-dexel models were obtained. The Boolean intersection of these five models and initial workpiece shape was computed and one triple-dexel model representing the total 3+2 machining result was finally obtained. The resultant model was further converted into a polyhedron using Marching Cubes algorithm. Figure 12 depicts the result workpiece models after the 3+2 axis machining. The machining result shape is correctly visualized. The resolutions of the triple-dexel model, total number of line segments in the cutter path, and necessary computation time are listed in Table 1. The computation time is described in terms of processing steps. From the table, it is evident that the results of complex 3+2 axis machining can be visualized within minutes. It is also evident that a significant amount of processing time is required in Step2 for converting the Z-map model into a triple-dexel model.

Sample model	Res. of triple-dexel model	Num. of G01 segments in cutter path	**Step1** Comp. time for Z-map simulation (s)	**Step2** Comp. time for Z-maps to triple-dexels (s)	**Step3** Comp. time for Boolean intersection (s)
A	1,123 x 932 x 309	1,457,908	23.24	89.89	0.45
B	1,092 x 959 x 661	1,585,724	27.30	227.70	0.71
C	1,083 x 967 x 928	2,873,793	40.38	375.63	0.76
D	1,023 x 1023 x 665	5,171,895	52.85	419.48	0.67

Table 1: Computation times for 3+2 axis machining simulations.

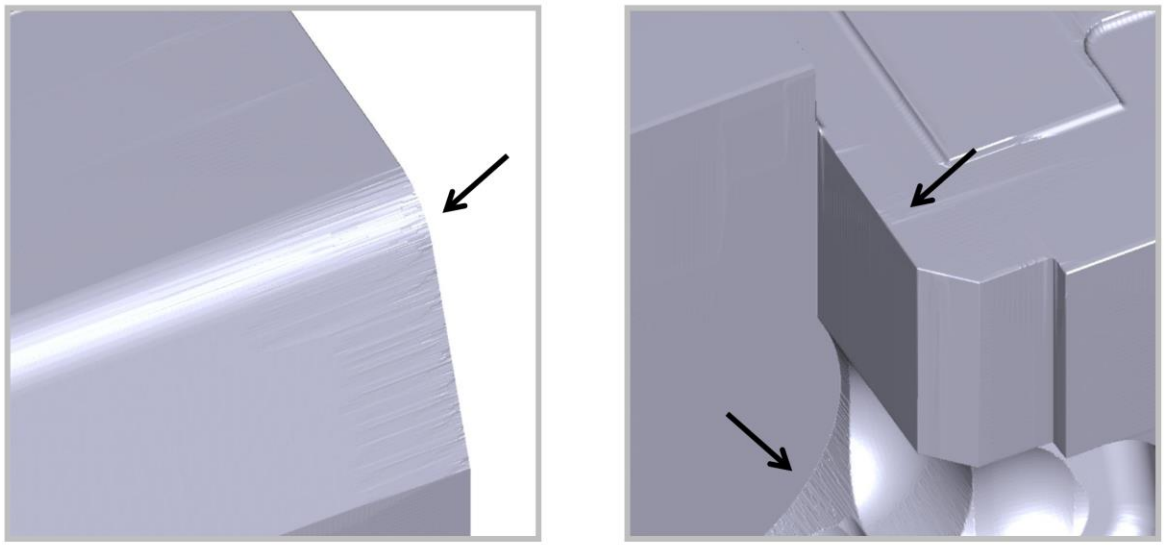


Figure 13: Shape errors appearing in the workpiece surface obtained by the simulation.

The simulation quality using this method depends considerably on the quality of the Z-map model used in the milling simulation. Because the Z-map is a shape representation based on the square mesh in the horizontal plane, a step-like shape error caused by the mesh structure is inevitable, particularly in the vertical wall shape. Such an error is transferred as-it-is to the triple-dexel model, thereby lowering the simulation quality. Figure 13 depicts two examples of shape errors observed

in the simulation results. These shape errors can be reduced by increasing the resolution of the Z-map model used in the simulation. When the resolution of the Z-map is increased, the number of polygons converted into polyhedron in the model increases. Consequently, the time required for converting the polyhedron into the triple-dexel model appears to increase. In our previous research, we developed a fast dexelization method for a polyhedron using the ray tracing (RT) core of GPU [14]. Using this technology, a polyhedron of several million triangles can be converted into a high-resolution dexel model within 1 s. By incorporating this technology, it is considered that the problems of the shape error and processing time in Step2 can be resolved.

6 CONCLUSIONS

In this paper, we propose a novel algorithm to realize geometric milling simulation of 3+2 axis machining. Using this algorithm, fast and accurate simulation is achieved by combining the Z-map based milling simulation technology and triple-dexel modeling. For each spindle direction, a Z-map based milling simulation is performed. The obtained machined shape is converted to a closed polyhedral model. Subsequently, it is further converted into a triple-dexel model using the ray algorithm. To accelerate the polyhedron-to-triple-dexel conversion, surface polygons are recorded and managed by a hierarchical bounding box structure, which is constructed based on the grid structure of the Z-map. The effectiveness of the method is verified by computational experiments. The computation accuracy of the software depends on the resolution of the Z-map used in the simulation. Although the use of higher-resolution Z-maps enables better simulation accuracy, it increases the computation time, particularly in the dexelization process of the polyhedral model. In our previous study, we realized a fast conversion from a polyhedral model to a dexel model using GPU's RT core technology. The dexelization problem of a high-resolution polyhedron is considered to be solvable using this technology.

Masatomo Inui, https://orcid.org/0000-0002-1496-7680
Qi Chen, https://orcid.org/0000-0001-8575-3420
Nobuyuki Umezu, https://orcid.org/0000-0002-7873-7833

REFERENCES

[1] Benouamer, M.O.; Michelucci, D.: Bridging the gap between CSG and brep via a triple ray representation, Proceedings of ACM Symposium on Solid Modeling and Applications, 1997, 68–79. https://doi.org/10.1145/267734.267755
[2] CGTech, http://www.cgtech.com/products/about-vericut/optipath/.
[3] Choi, B.K.; Jerard, R.B.: Sculptured Surface Machining, Theory and Applications, Kluwer Academic Publishers, 1998, 255–258.
[4] Drysdale, R.L.; Jerard, R.B.; Schaudt, B.; Hauck, K.: Discrete simulation of NC machining, Algorithmica, 4(1), 1989, 33–60. https://doi.org/10.1007/BF01553878
[5] Erdim, H.; Sullivan, A.: Cutter Workpiece Engagement Calculations for Five-axis Milling Using Composite Adaptively Sampled Distance Fields, Procedia CIRP, 8, 2013, 438–443. https://doi.org/10.1016/j.procir.2013.06.130
[6] Fridshal, R.; Cheng, K.P.; Duncan, D.; Zucker, W.: Numerical control part program verification system, Proceeding of Conference on CAD/CAM Technology in Mechanical Engineering, 1982, 236–254.
[7] Gong, X.; Feng, H.-Y.: Cutter-workpiece engagement determination for general milling using triangle mesh modeling, Journal of Computational Design and Engineering, 3(2), 2016, 151–160. https://doi.org/10.1016/j.jcde.2015.12.001
[8] Huang Y.; Oliver, J.H.: NC milling error assessment and tool path correction, SIGGRAPH 1994 Proc. 21st Annual Conference on Computer Graphics and Interactive Techniques, 1994, 287–294. https://doi.org/10.1145/192161.192231

[9] Inui, M.; Kakio, R.: Fast visualization of NC milling result using graphics acceleration hardware, in Proc. IEEE International Conference on Robotics and Automation, 2000, 3089–3094. https://doi.org/10.1109/ROBOT.2000.845138
[10] Inui, M.; Ohta, A.: Using a GPU to Accelerate Die and Mold Fabrication, IEEE Computer Graphics and Applications, January/February 2007, 82–88. 10.1109/MCG.2007.23
[11] Inui, M.; Umezu, N.: GPU Accleration of 5-Axis Milling Simulation in Triple Dexel Representation, Proc. of 2010 International Symposium on Flexible Automation, July 12, 2010.
[12] Inui, M.; Umezu, N.; Shinozuka, Y.: A Comparison of Two Methods for Geometric Milling Simulation Accelerated by GPU, Transactions of the Institute of Systems, Control and Information Engineers, 26(3), 2013, 95–102. https://doi.org/10.5687/iscie.26.95
[13] Inui, M.; Huang, Y.; Onozuka, H.; Umezu, N.: Geometric simulation of power skiving of internal gear using solid model with triple-dexel representation, Procedia Manufacturing, 48, 2020, 520-527, https://doi.org/10.1016/j.promfg.2020.05.078
[14] Inui, M.; Kaba, K.; Umezu, N.: Fast Dexelization of Polyhedral Models Using Ray-Tracing Cores of GPU, Computer-Aided Design & Applications, 18(4), 2021, 786-798. https://doi.org/10.14733/cadaps.2021.786-798
[15] Jachym, M.; Lavernhe, S.; Euzenat, C.; and Tournier, C.: Effective NC machining simulation with OptiX ray tracing engine, The Visual Computer, 35, 2019, 281–288. https://doi.org/10.1007/s00371-018-1497-7
[16] Jerard, R.B.; Hussaini, S.Z.; Drysdale, R.L.; Schaudt, B.: Approximate methods for simulation and verification of numerically controlled machining programs, The Visual Computer, 5(6), 1989, 329–348. https://doi.org/10.1007/BF01999101
[17] Lorensen, W.E.; Cline, H.E.: Marching Cubes: A High-Resolution 3D Surface Construction Algorithm, Computer Graphics, Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987, 163–169. https://doi.org/10.1145/37401.37422
[18] Menon, J.P.; Robinson, D.M.: Advanced NC verification via massively parallel ray-casting: extensions to new phenomena and geometric domains, ASME Manufacturing Review, 6(2), 1993, 141–154.
[19] Menon, J.; Marisa, R.J.; Zagajac, J: More powerful solid modeling through ray representations, IEEE Computer Graphics and Applications, 14(3), May 1994, 22-35. https://doi.org/10.1109/38.279039
[20] Moller, T.; and Haines, E.: Real-Time Rendering, A K Peters, 1999.
[21] Oliver, J.H.; Goodman, E.D.: Direct dimensional NC verification, Computer-Aided Design, 22(1), 1990, 3–10. https://doi.org/10.1016/0010-4485(90)90023-6
[22] Saito, T.; Takahashi, T.: NC machining with G-Buffer method, in SIGGRAPH 1991 Proceedings of 18th Annual Conference on Computer Graphics and Interactive Techniques, 25(4), 1991, 207–216. https://doi.org/10.1145/122718.122741
[23] Takata, S.; Tsai, M.D.; Inui, M.; Sata, T.: A cutting simulation system for machinability evaluation using a workpiece model, CIRP Annals, 38(1), 1989, 417-420. https://doi.org/10.1016/S0007-8506(07)62736-X
[24] Tapoglou, N.: Calculation of non-deformed chip and gear geometry in power skiving using a CAD-based simulation, International Journal of Advanced Manufacturing Technology, 100, 2019, 1779-1785. https://doi.org/10.1007/s00170-018-2790-3
[25] Van Hook, T.: Real-time shaded NC milling display, SIGGRAPH 1996 Proceedings of 13th Annual Conference on Computer Graphics and Interactive Techniques, 20(4), 1996, 15–20. https://doi.org/10.1145/15922.15887
[26] Wang, W.P.; Wang, K.K.: Geometric modeling for swept volume of moving solids, IEEE Computer Graphics and Applications, 6(12), 1987, 8–17. https://doi.org/10.1109/MCG.1986.276586