Title:
Contour-Type Cutter Path Computation Using Ultra High Resolution Dexel Model

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

Keywords:
NC Milling, Minkowski Sum, Solid Modeling, CAM

DOI: 10.14733/cadconfP.2019.353-357

Introduction:
In the machining of a mold part with complex curved surface, a contour-type cutter path with very small vertical intervals is necessary. Development of an algorithm for robust, accurate and fast cutter path computation is strongly requested by computer-aided manufacturing (CAM) software companies. In this paper, we propose a novel algorithm for computing a contour-type cutter path for a mold CAD model in the polyhedral representation. Our algorithm uses a Minkowski sum of the mold model and a cutter model in the inverted orientation in the path computation. Mikowski sum for a polyhedral object can be obtained by computing Minkowski sum shapes for the individual polygons (triangles) of the object, and by computing their Boolean union shape. In the conventional boundary representation modeling, the difficulty of the computation increases significantly when a complex model with many polygons is given, because topological reconstruction by trimming and reconnecting the surfaces into a closed model must be iterated many times.

To overcome this difficulty, a grid-based shape representation with no topological information is used in our Minkowski sum computation. Grid-based shape representation can be considered as sampling of the surface points according to the spatial cell structure. To reduce the shape error caused by the sampling interval, use of a cell structure based on an ultra-high-resolution grid is necessary. This approach causes inevitable increase of the memory usage and the processing time. There exists a multitude of contour-type cutter path computation algorithms. Some of them use the Minkowski sum of the mold shape and the cutter. However, due to problems with computation speed and the accuracy, there is no practical algorithm that uses a grid-based shape representation. In our study, two-directional dexel model is introduced for saving the memory usage. Parallel processing with many cores of GPU is also introduced for reducing the computation cost. Our cutter path computation software is robust, accurate and generally faster than the conventional CAM software.

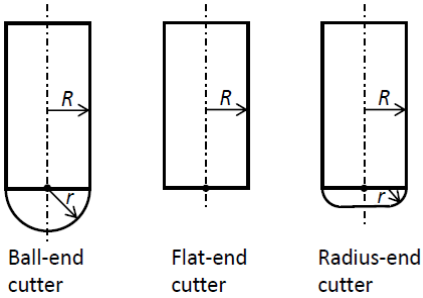

Fig. 1: Cutter shape specifications.

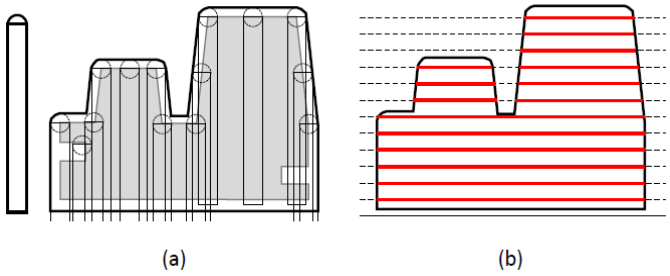

Fig. 2: Contour-type cutter path computation process.

Main Idea:
We consider the contour-type cutter path computation for a milling process with a ball-end, flat-end or radius-end cutter in a vertical spindle axis direction. The input data of our algorithm consists of a polyhedral STL model of a mold part, shape data of a cutter (tool radius R and corner radius r in Fig. 1), and a list of z coordinates specifying the height information of the contour-type path. It computes a Minkowski sum shape of the mold model and the cutter model in the inverted orientation (Fig. 2(a)). Obtained shape is sliced by horizontal planes positioned at the heights specified in the given list (Fig. 2(b)). Boundary curves of the cross-sectional figures correspond to the cutter path.

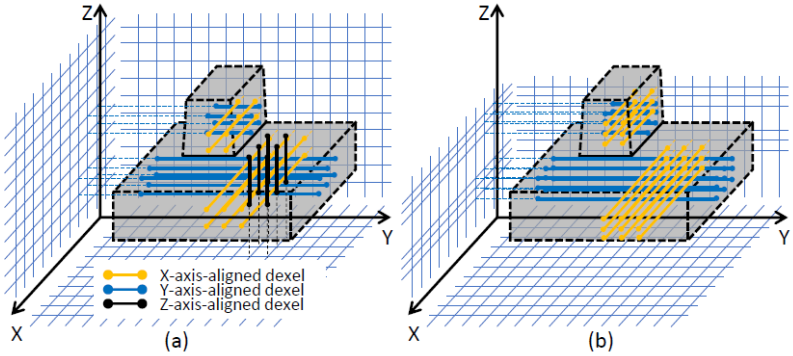

Fig. 3: Triple-dexel model (a) and dexels for representing sectional figures (b).

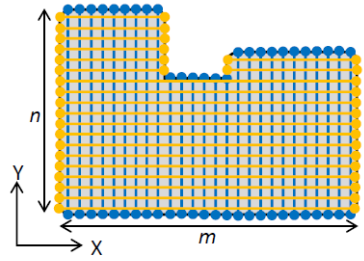

Fig. 4: Two-directional dexel model.

Two-Directional Dexel Model
In our study, two-directional dexel model is used for representing the Minkowski sum shape. In the original dexel model, a 3D object is represented by a series of vertical segments (dexels) defined at each grid point in a regular square grid in the xy-plane. Each segment corresponds to an overlapping range between the vertical ray that originates from each grid point and the object. In this method, near-vertical surfaces have inevitable staircase errors. A triple-dexel model was proposed to overcome this non-uniformity of the representation accuracy. In this representation, the 3D shape is not only defined by the z-axis-aligned (vertical) dexels, but also the x-axis-aligned dexels based on a grid in the yz-plane and y-axis-aligned dexels based on a grid in the zx-plane (Fig. 3(a)) [1].

In the contour-type path computation, the Minkowski sum shape is sliced by a horizontal plane in a specific height. Boundary curves of the cross-sectional figure correspond to the path at the height. To properly represent the section figures in the horizontal planes at the specific heights, we modify the grid definitions of the triple-dexel model. Horizontal lines of the grids in the yz- and zx-plane are arranged so that their z-coordinates become equal to the height specifications for the contour-type path. Since the z-axis-aligned dexels do not contribute the cross-sectional figures in the horizontal plane, they are removed from the representation (Fig. 3(b)). Removal of the z-axis-aligned dexels enables drastic reduction of the amount of memory required for representing the model.

Cross-sectional figures in each slicing plane are represented by x-axis-aligned dexels and y-axis-aligned dexels. We call this representation method two-directional dexel model (Fig. 4). Two-directional dexel model is equivalent to a 2D figure representation method using a regular square mesh. Consider a square mesh with resolution m in the x-axis direction and n in the y-axis direction. In the two-directional dexel model, the amount of the memory required for recording figures in the mesh is proportional to $m + n$ which is much smaller than mn necessary for recording the figures in the square mesh. m (or n) can be larger than 20,000 in our current implementation.

Minkowski Sum Computation Using Two-Directional Dexel Model
Our algorithm to explain is a modification of our offset computation algorithm using triple-dexel model [2]. Consider Minkowski sum computation of a single triangle and an inverted flat-end cutter of radius R and length l. We assume that l is sufficiently large (larger than height of the target mold part). The result shape is a composition of the following three types of surface elements (see Fig. 5).

- For each vertex v of the triangle, the inverted flat-end cutter is placed so that the reference point of the cutter and v become coincident.
- For each edge e, ruled surfaces generated by sweeping the inverted cutter along e are placed. Consider a vector $\boldsymbol{u}$ from one end point of e to the other end point. By using the vector $\boldsymbol{u}$, the surface of the cutter can be classified to a region whose normal vector faces to $\boldsymbol{u}$ and the other region whose normal vector is perpendicular to $\boldsymbol{u}$ or faces to the opposite direction of $\boldsymbol{u}$. We can obtain the ruled surfaces by linearly sweeping the border curves between these two regions along e. They are a slant cylinder connecting two end points of e, another slant cylinder locating l lower from the first slant cylinder, and two vertical parallelograms connecting two slant cylinders.
- Two triangles which are obtained by shifting the original triangle f by a vector $\boldsymbol{s}$ and its opposite vector – $\boldsymbol{s}$, where $\boldsymbol{s}$ is given by projecting the normal vector $\boldsymbol{n}$ of the triangle to the horizontal plane, then adjusting its length to be equal to R. The latter triangle is further moved in the negative z-axis direction by l.

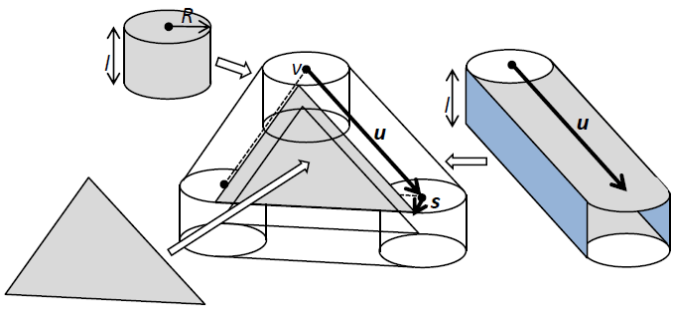

Fig. 5: Surface elements organizing Minkowski sum shape.

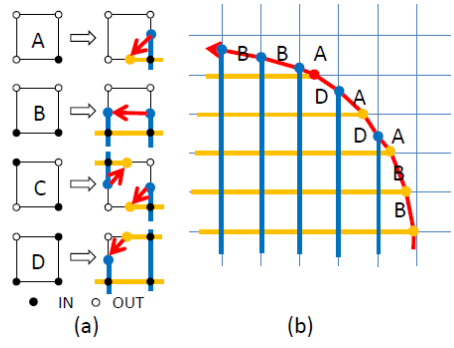

Fig. 6: Boundary curve tracing method.

For each line of the square mesh in the slicing plane, intersection points between the line and all surface elements mentioned above are calculated. Since the Minkowski sum of a triangle and an inverted flat-end cutter has a convex shape, points at both ends of the point sequence on the line correspond to the boundary of the overlapping range between the Minkowski sum shape and the line. This range is replaced to a dexel on the line. This operation is iterated for all grid lines in the slicing plane and a two-directional dexel model representing a cross-sectional figure of the Minkowski sum shape for a single triangle is obtained. Boolean union of the Minkowski sum shapes for all triangles of the mold CAD model represents the cross-sectional figure of the Minkoski sum shape of the mold model and the inverted flat-end cutter. This computation is iterated for all slicing planes. As a result, Minkowski sum shape for the mold model is obtained as a stack of two-directional dexel models.

In the computation process mentioned above, enormous number of intersection point calculations and Boolean union computations of dexels on the same line are executed. These computations on a line is independent of those on other lines. Thus, the intersection point calculation and the dexel-wise Boolean union computation can be done in a parallel manner. To implement parallel Minkowski sum computation software, we use the Compute Unified Device Architecture (CUDA). Current GPUs are designed to have thousands of streaming processors (SP) on a chip. By using CUDA, programmers can utilize a GPU as a general purpose parallel processor in which each SP executes a unit computation (or thread).

Ball-end cutter shape can be considered as a composition of a vertical cylinder and a sphere as shown in Fig. 1. Minkowski sum computation of a polyhedral CAD model and an inverted ball-end cutter of radius r thus can be realized by a Boolean union computation of following two 3D objects:

- A Minkowski sum shape of the CAD model and an inverted flat-end cutter of radius R (= r),
- Another Minkowski sum shape of the same model and a sphere of radius r.

Minkowski sum shape of a triangle and a sphere is equivalent to a composition of three spheres, three cylinders, and a thick plate (slab) placed on the triangle as follows:

- Three spheres placed on three vertices of the triangle.
- Three open cylinders placed along each edge e of the triangle, with the center axis of the cylinders coinciding with the edges.
- A slab with the area of the triangle and thickness $2r$ placed on the triangle with the center plane of the slab coinciding with the triangle.

Two-directional dexel model of the Minkowski sum shape of the mold CAD model and the sphere can be obtained by using a method similar to the Minkowski sum computation for the inverted flat-end cutter. The result shape is then combined to the Minkowski sum shape for a flat-end cutter of radius R. Minkowski sum computation for a radius-end cutter will be discussed later.

Contour-Type Path Computation Using Two-Directional Dexel Model

Contour-type cutter path is obtained by tracing the boundary of the cross-sectional figure in the two-directional dexel representation. Consider grid lines defining x-axis-aligned dexels and y-axis-aligned dexels in a slicing plane. These lines organize a regular square mesh in the plane. The two-directional dexel model is mapped on the square mesh. Each grid point of the mesh is then marked IN or OUT according to the following rule:

- If a grid point is within the range of a x-axis-aligned dexel or y-axis-aligned dexel, then the grid point is marked IN because it locates within the cross-sectional figure.
- Otherwise, it locates outside of the figure and it is marked OUT.

After marking all grid points, each square cell in the mesh is classified to four types according to the pattern of marks at its four corner points (Fig. 6(a)). Directed short segments are inserted in the cell so that they connect two dexel points in the same cell (short red arrows in the figure). The segments are traced and the boundary curve around the cross-sectional figure is obtained as a contour-type path (Fig. 6(b)).

In our boundary curve computation method, two points in the same cell are connected by a single segment, therefore sharp corners in the boundary curve cannot be reproduced appropriately. To improve reproducibility of the sharp corners, we modify the segment insertion method in the cell. Before connecting two points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ in a cell, tangent vectors $\boldsymbol{t}_0$ and $\boldsymbol{t}_1$ at the points are checked. Segment insertion method is changed according to the angle θ organized by $\boldsymbol{t}_0$ and $\boldsymbol{t}_1$ as follows:

- If θ is larger than θ_0 and smaller than θ_1, consider a tangent lines l_0 passing $\boldsymbol{p}_0$ and another tangent line l_1 passing $\boldsymbol{p}_1$, and compute their intersection point $\boldsymbol{p}$. Two segments $\boldsymbol{p}_0\boldsymbol{p}$ and $\boldsymbol{p}\boldsymbol{p}_1$ are inserted in the cell. $\boldsymbol{p}$ corresponds to a sharp corner.
- Otherwise, $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are connected by a single segment.

In our current implementation, we assign 5 degree to θ_0 and 170 degree to θ_1. To properly compute the tangent vector at the point in the cell, we modify the Minkowski sum computation software so that the normal vector at the dexel's end point is computed and recorded in the two-directional dexel model construction process. Since our model uses a ultra-high-resolution grid in the computation, the cell size is sufficiently small. This simple segment insertion method thus can reproduce the boundary curve accurately.

Minkowski Sum Computation for Radius-End Cutter

Shape of a radius-end cutter of radius R and corner radius r is equivalent to a Minkowski sum shape of a cylinder of radius R - r and a small sphere of radius r. Minkowski sum shape of a polyhedral mold model and an inverted radius-end cutter can be obtained in the following two step way:

Step 1: Compute Minkowski sum shape of the mold model and a sphere of radius r. The result shape is named M.

Step 2: Compute Minkowski sum shape of M and an inverted cylinder of radius $R - r$.

Since model M obtained in the first step is already in the two-directional dexel representation, we need a Minkowski sum computation algorithm for a model in that representation. Instead of developing a

new algorithm, we compute the contour-type path using the model *M* just after **step 1**, then compute the final cutter path for the radius-end cutter by horizontally expanding the path by *R* – *r*.

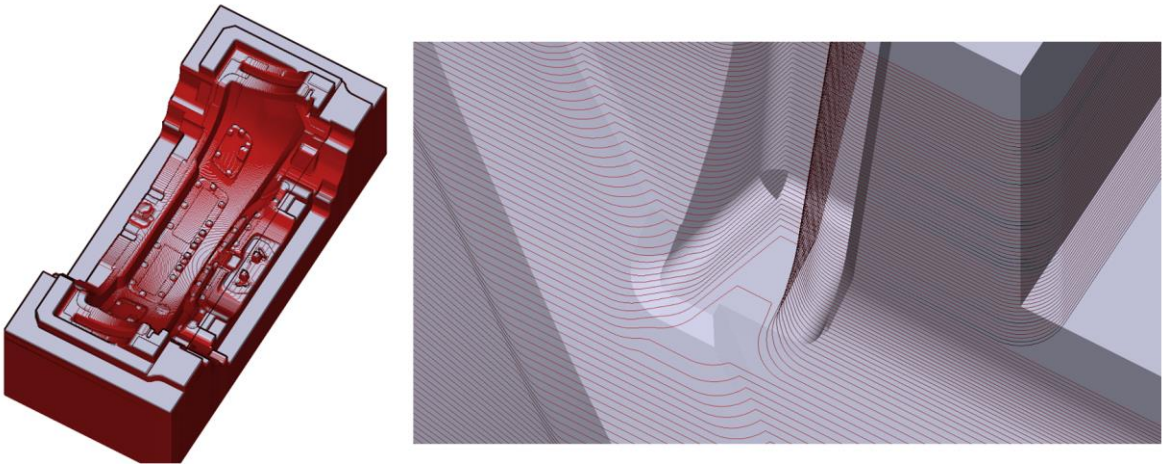

Fig. 7: Contour-type cutter path computation result and its closeup picture.

Computational Experiments

We implemented a contour-type cutter path computation software using VisualStudio 2010 and CUDA 7.5. A 64-bit PC with an Intel Core i7 Processor (2.8GHz), 16 GB memory, and an nVIDIA GeForce GTX-1050 GPU is used in the experiments. Fig. 7 illustrates a mold cavity CAD model of 2,460 x 1,085 x 892 mm size and a contour-type cutter path generated by our software. This CAD model has 844,180 polygons. The cutter path is for a flat-end cutter of 10 mm radius. In the computation, two-directional dexel model based on 20,480 x 9,346 resolution grid is used. The vertical interval between the slicing planes is set to 1.5 mm. Total number of the slicing planes is 598. 827 s is necessary for the cutter path computation. After the computation, a milling simulation is conducted to evaluate the quality of the cutter path. Cutter path obtained by our software achieves accuracy of 0.001 mm.

Conclusion:

In this paper, we propose a novel algorithm for computing a contour-type cutter path for a mold CAD model in the polyhedral representation. Our algorithm uses a Minkowski sum shape of the mold model and a cutter model in the inverted orientation. To realize robust Minkowski sum computation, a grid-based shape representation named two-directional dexel model is used. Two-directional dexel model is geometrically equivalent to a 2D shape representation method using a regular square mesh. It can represent the 2D object with much smaller amount of the memory than the regular square mesh-based method. Utilizing this advantage, we realized two-directional dexel models based on an ultra-high-resolution grid. A contour-type cutter path computation software was implemented to demonstrate the performance of the algorithm. It can compute a cutter path with sufficient accuracy for machining a large mold part.

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] Inui, M.; Umezu, N.; Kitamura, Y.: Visualizing sphere-contacting areas on automobile parts for ECE inspection, Journal of Computational Design and Engineering, 2(1), 2015, 55-66. http://dx.doi.org/10.1016/j.jcde.2014.11.006