Int J Adv Manuf Technol (2006) 27: 797–803
DOI 10.1007/s00170-004-2239-8

ORIGINAL ARTICLE

Dong-Yoon Lee · Su-Jin Kim · Hyun-Chul Kim · Sung-Gun Lee · Min-Yang Yang

Incomplete two-manifold mesh-based tool path generation

Received: 13 February 2004 / Accepted: 28 April 2004 / Published online: 16 March 2005

Abstract This paper presents a new paradigm for three-axis tool path generation based on an incomplete two-manifold mesh model; namely, an inexact polyhedron. When geometric data is transferred from one system to another system and tessellated for tool path generation, the model does not have any topological data between meshes and facets. In contrast to the existing polyhedral machining approach, the proposed method generates tool paths from an incomplete two-manifold mesh model. In order to generate gouge-free tool paths, cutter-location meshes (CL-meshes) are generated by offsetting boundary edges, boundary vertices, and facets. The CL-meshes are sliced by machining planes and the calculated intersections are sorted, trimmed, and linked. The grid method is used to reduce the computing time when range searching problems arise. The method is fully implemented and verified by machining an incomplete two-manifold mesh model.

Keywords CL-mesh · Grid method · Incomplete two-manifold mesh · Offset · Tool path generation

1 Introduction

Tool path generation is a procedure to compute a sequence of cutter-location points (CL-points) from the design-surface for given machining requirements such as allowance and tolerance. It is a difficult task that has been addressed by a number of researchers since the 1980s. Tool path generation methods are classified either as cutter-contact-based (CC-based) methods or CL-based methods [1]. In CC-based methods, tool paths are generated by converting the sampled cutter-contact (CC) points from the design-surface and the results are precise. Recently, high-speed machining (HSM) has been adopted in industrial fields, and new tool path generation methods for this machining strategy have been developed [2, 3]. However, interferences (gouges) between the tool and the design-surface may arise, and their detection and correction is difficult [4, 5]. In order to overcome this problem, cutter interference regions are searched and eliminated before tool paths are generated [6]. On the other hand, in CL-based methods, a cutter-location surface (CL-surface) is first calculated from the design-surface and tool paths are generated using the calculated CL-surface. The CL-surface represents a trajectory surface of the reference point of a cutter when the cutter slides over the design-surface. In three-axis milling, gouges can be avoided by computing the CL-surface prior to the path generation [7].

Recent commercialized CAM software systems use a tessellated (mesh) model to represent design-surfaces and CL-surfaces and to generate tool paths. Duncan and Mair's polyhedral machining was an early work utilizing this approach [8]. To make a CL-surface from a polyhedral model, the model is offset by a local-offsetting scheme, which uses the topological information and geometric characteristics of the model [9]. The approach is extended to the application of non-spherical cutters such as flat and filleted endmills [10]. An inverse tool offset method and a hidden-surface elimination mechanism of the polygon rendering hardware have been used to generate the CL-surface quickly [11]. However, all of the aforementioned methods can only be applied to polyhedral models that satisfy a two-manifold condition. This paper presents a new tool path generation method based on an incomplete two-manifold mesh model; namely, an inexact polyhedral model that does not have any topological data between surfaces and tessellated facets. The method consists of three major steps: a design-surface is first tessellated into triangular or quad meshes; cutter-location meshes (CL-meshes) are then generated and the desired tool paths are finally calculated by slicing the CL-meshes with machining (guide) planes. It is unnecessary to discuss the surface-tessellation step in detail since there are a number of available techniques, some of which are for trimmed surfaces [12, 13].

D.-Y. Lee (✉) · S.-J. Kim · H.-C. Kim · S.-G. Lee · M.-Y. Yang
Department of Mechanical Engineering,
Korea Advanced Institute of Science and Technology,
305-701, Daejeon, South Korea
E-mail: daniel.lee@kaist.ac.kr
Tel.: +82-42-8693264
Fax: +82-42-8693210

In Sect. 2, some problems that may occur during the geometric data transfer between different systems are briefly explained followed by an explanation of the incomplete two-manifold mesh. Our approach for the computation of the CL-mesh from the incomplete two-manifold mesh is given in Sect. 3. The grid method used to improve the tool path generation performance is explained in Sect. 4, and the proposed method is implemented and demonstrated by illustrative examples in Sect. 5.

2 Incomplete two-manifold mesh

Hundreds of commercial CAD/CAM software systems are used around the world in the metal working industry. These systems are developed based on different graphic kernels, bringing about the need for geometric data transfer. Generally, the geometry of a designed part is described by surfaces that are, in most cases, defined by non-uniform rational B-splines (NURBS). The mathematical formulas that describe these NURBS are very complex and each kernel implements those algorithms in its own way. Although many studies have been conducted in this area and data transfer techniques have become more advanced, the fundamental differences between the numerical descriptions and accuracy of geometry still yield numerous problems: missing surface patches, distortion of boundary curves, overlapping of patches, gaps between patches, trimming errors, tangent discontinuous, etc. [14]. Consequently, the transferred data violates the two-manifold condition; that is, points and curves shared by adjacent surfaces do not coincide exactly. As the model becomes more complex – i.e., the model has many trimmed surfaces and more complicated modeling history – more errors occur. When these errors appear in transferred data, an engineer heals the model manually, and sometimes this takes much more time in comparison with the tool path generation time. In order to generate tool paths from a tessellated model, all design-surfaces have to be converted to meshes, but the converting task of each surface is conducted independently without considering adjacent surfaces. As a result, the tessellated model satisfies the two-manifold condition in the inner region of a mesh, but violates the condition on the boundary region where the mesh meets neighboring meshes. Consequently, the tessellated model is composed of incomplete (bounded) two-manifold meshes, and thus this model is not a polyhedron.

For example, four patches are designed in a CAD system A, and transferred to another system B as shown in Fig. 1. By transferring, four points are generated from the one original point. The enlarged view of the tessellated patches shows that the two-manifold condition is satisfied in the inner region of a patch, but the condition is not satisfied in the boundary regions that are indicated by thicker lines. It is known that to make a polyhedron from incomplete two-manifold meshes, additional post-processes including topology construction and geometric healing are needed.

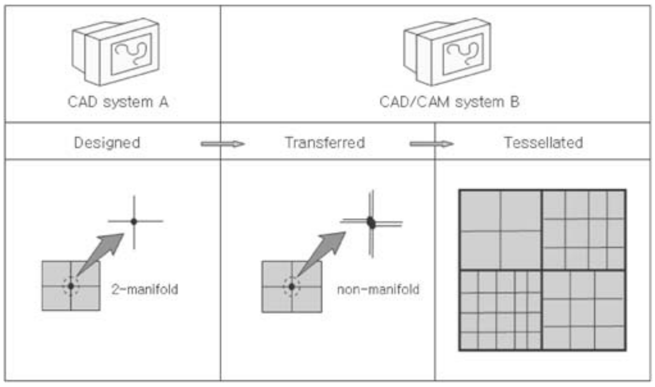

Fig. 1. Geometric data loss by transfer and tessellation

Figure 2 shows the difference between incomplete two-manifold mesh model and complete two-manifold mesh model. In the case of the former, vertices and edges that are located in the boundary region of a mesh do not coincide with vertices and edges of other meshes as shown in Fig. 2a, but in the case of the latter, all vertices and edges are shared by neighboring edges and facets as shown in Fig. 2b.

3 Cutter location mesh

In order to make a CL-mesh, we offset the tessellated model. An offset procedure is defined as making a new geometrical entity which is at constant distant d along the normal from a generator entity such as a curve, surface, or mesh [15]. As noted in the previous section, the tessellated model is not an exact polyhedron and this causes some serious problems in the boundary regions of the mesh. If the boundary region has a convex shape, gaps may exist between the offset meshes.

Figure 3a shows the two neighboring incomplete two-manifold meshes that meet each other in a convex shaped boundary region. If they are offset only with respect to the normal, gaps

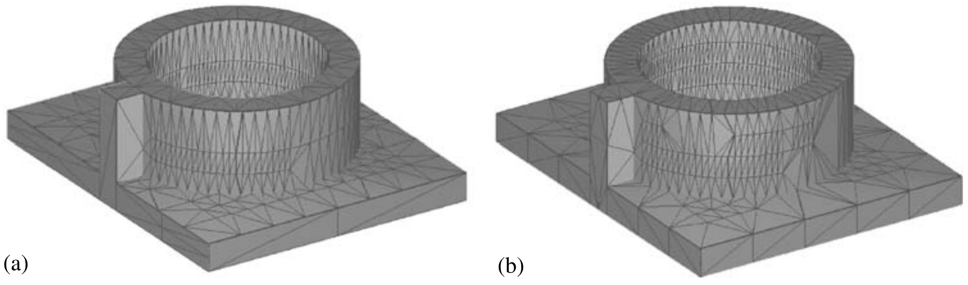

Fig. 2a,b. Difference between incomplete and complete two-manifold mesh model. **a** Incomplete two-manifold mesh model. **b** Complete two-manifold mesh model

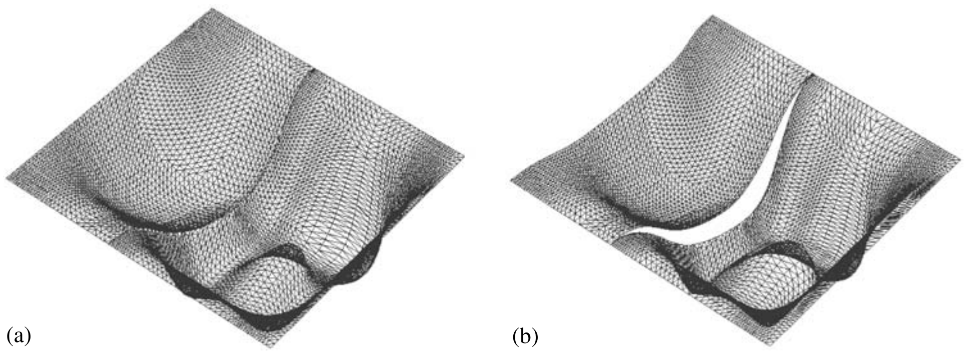

Fig. 3a,b. Offsetting of two incomplete two-manifold meshes. **a** Two incomplete two-manifold meshes. **b** Calculated offset meshes

are generated naturally as shown in Fig. 3b. In order to eliminate these gaps, the proposed method fills them with spherical or cylindrical meshes.

The offset method presented in this paper consists of three steps: all boundary edges are searched from the original meshes and they are offset using a cylindrical mesh; from the searched edges, all boundary vertices are gathered and offset using a spherical mesh; finally, facets in a mesh are offset by moving the vertex of the facet to the new position, which can be calculated with the normal vector of the vertex and the offset radius. The detailed algorithms employed in implementing the aforementioned steps are as follows:

Step 1: Boundary edge offset

A boundary edge is located on the outer boundary of a mesh. Compared with a normal edge, a boundary edge has only one facet and all of the boundary edges can be searched by utilizing this characteristic. When a boundary edge is found, a cylindrical mesh, which wraps around the edge, is appended to the CL-meshes. Given the tolerance, allowance, and the cutter radius, a cylindrical mesh that is composed of quad facets can be calculated as follows: from the two vertices that form the edge and the normal vector of the facet to which the edge belongs, the basis vector $\vec{m}$ is calculated and the interval $\delta\theta$ is determined by the tolerance ε and the offset radius r, which is determined by the

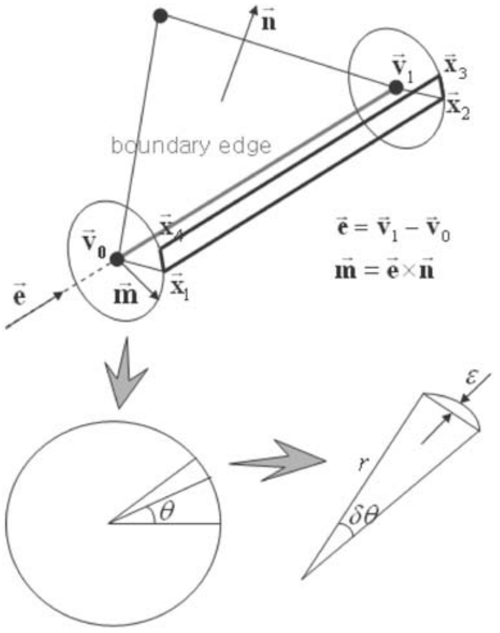

Fig. 4. Offsetting of the boundary edge

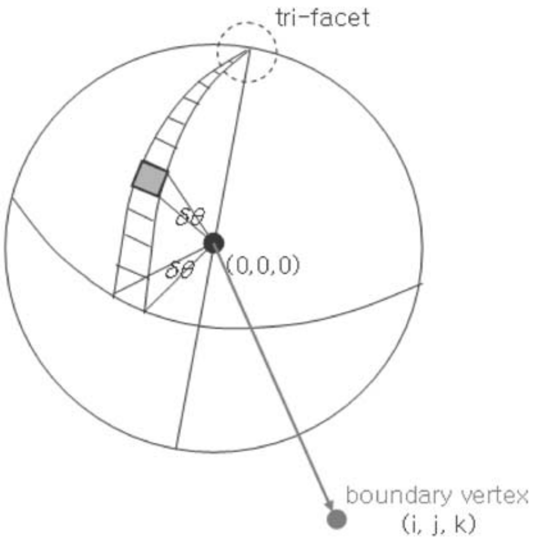

Fig. 5. Offsetting of the boundary vertex

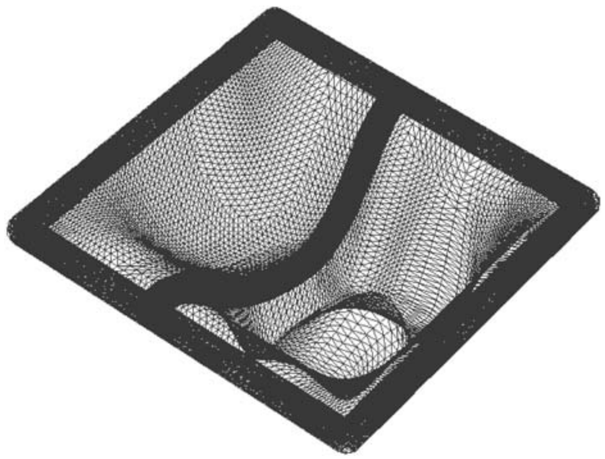

Fig. 6. Generated CL-meshes of Fig. 3a

cutter radius and the allowance (see Fig. 4).

$$\delta\theta = 2\cos^{-1}\left(1 - \frac{\varepsilon}{r}\right) \tag{1}$$

The four points that compose the quad facets of the cylindrical mesh are given by:

$$\begin{aligned}
\vec{x}_1 &= \vec{v}_0 + r\left(\vec{m}\cos\theta + \vec{n}\sin\theta\right)\\
\vec{x}_2 &= \vec{v}_1 + r\left(\vec{m}\cos\theta + \vec{n}\sin\theta\right)\\
\vec{x}_3 &= \vec{v}_1 + r\left(\vec{m}\cos(\theta+\delta\theta) + \vec{n}\sin(\theta+\delta\theta)\right)\\
\vec{x}_4 &= \vec{v}_0 + r\left(\vec{m}\cos(\theta+\delta\theta) + \vec{n}\sin(\theta+\delta\theta)\right)
\end{aligned} \tag{2}$$

Step 2: Boundary vertex offset

The boundary vertex belongs to the boundary edge noted above. A spherical mesh, which wraps around the vertex, is appended to the CL-meshes.

Figure 5 shows a spherical mesh where r and $\delta\theta$ are the same as those in the boundary edge case. As shown in Fig. 5, a spherical mesh is composed of quad facets except at the polar regions. Only one spherical mesh is made at the origin (0, 0, 0), and it is copied to all boundary vertices to reduce calculation time.

Step 3: Facet offset

After offsetting boundary edges and vertices, all facets are offset by moving a vertex to a new location along the normal

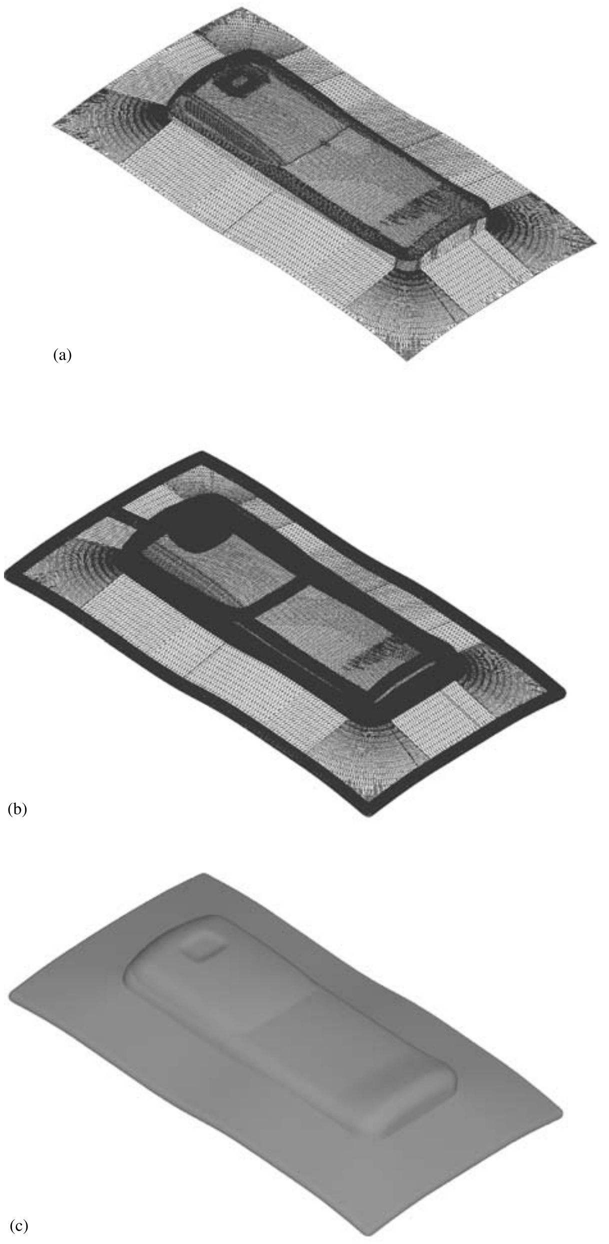

Fig. 7a–c. Examples of generating CL-meshes for the phone mold. **a** Tessellated model. **b** Generated CL-meshes. **c** CL-meshes (shading image)

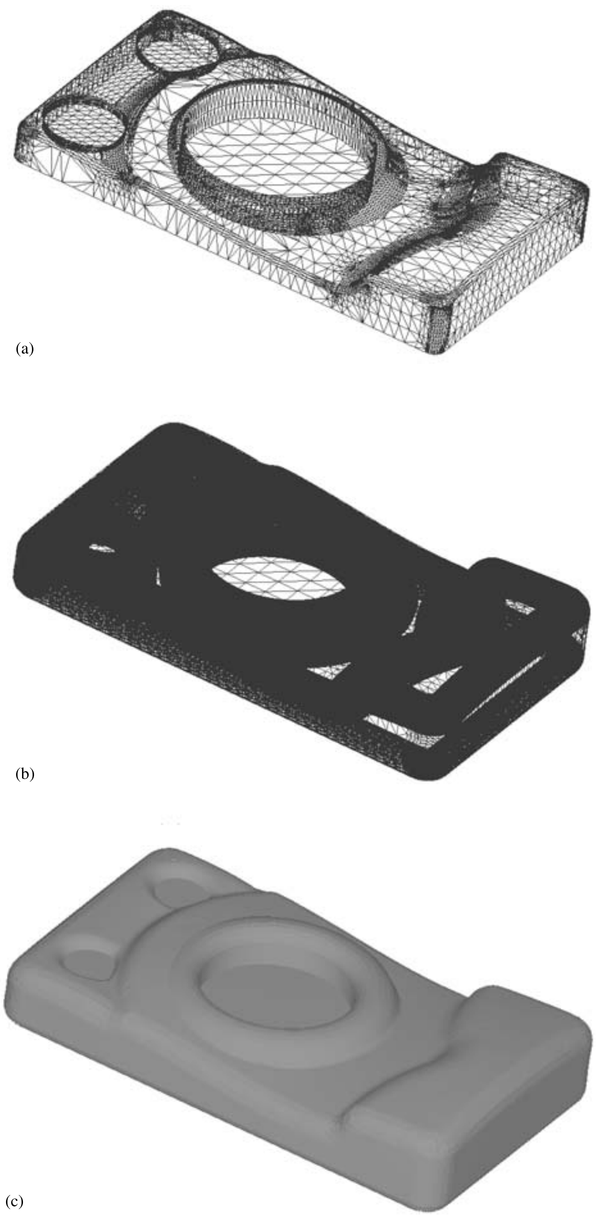

Fig. 8a–c. Examples of generating CL-meshes for the speaker mold. **a** Tessellated model. **b** Generated CL-meshes. **c** CL-meshes (shading image)

direction of the vertex. The new vertex is calculated using the following equation:

$$\vec{q} = \vec{p} + r\vec{n} \qquad (3)$$

For a smooth surface having a large curvature, this method works well, but if the surface is too sharp, unacceptable errors may be produced. To avoid this error, Qu suggested a method for calculating the offset vector for a vertex by using the weighted sum of the normal vectors of the facet [16].

The gap that is shown in Fig. 3b is filled by spherical or cylindrical meshes as shown in Fig. 6.

Figures 7 and 8 are more complicated examples of the proposed offset method for the phone mold, whose dimensions are $132.5 \times 227.5 \times 31.28$ mm, and the speaker mold, whose dimensions are $50 \times 92 \times 13.5$ mm.

4 Tool path generation from CL-meshes using grid method

When CL-meshes are generated, the tool paths are obtained by calculating the intersections between the generated CL-meshes and the machining (drive, guide) surfaces (see Fig. 9). In order to calculate the intersections efficiently, the candidate facets that can be intersected with a machining surface should be searched as rapidly as possible. As noted in Sect. 1, an incomplete two-manifold mesh model has no topological information, and consequently, it is very difficult to find a facet that is located in a specific region. Sometimes, all facets should be checked to find a desirable one, which is very time-consuming task. This problem naturally yields a range searching problem. A large variety of data structures have been proposed for use in collision detection algorithms, with the goal of speeding up query processing and response time [17]. Among those data structures, we use the grid method to store facets and query them, as its construction takes almost $O(n)$ time when n entities are stored, and a range query which reports r entities takes $O(r)$ if the entities are uniformly distributed.

4.1 Grid method

A spatial subdivision is a decomposition of some spatial domain into smaller pieces. Known also as partitions or simply subdivisions, they permit a problem involving the domain, or geometric objects lying in the domain, to be reduced to simpler or smaller subproblems [18]. A square domain is divided into a grid of small squares or cells – for each cell, a reference list of the entities that overlap the cell (in our study, facets) is built. The domain is set by a bounding box, which can include all entities of concern.

Figure 10 provides a flow chart for the procedure of storing entities by using the grid method.

Figure 11 shows an example of the grid method involving two entities. A star-shaped entity is stored in cells (4, 3), (4, 4),

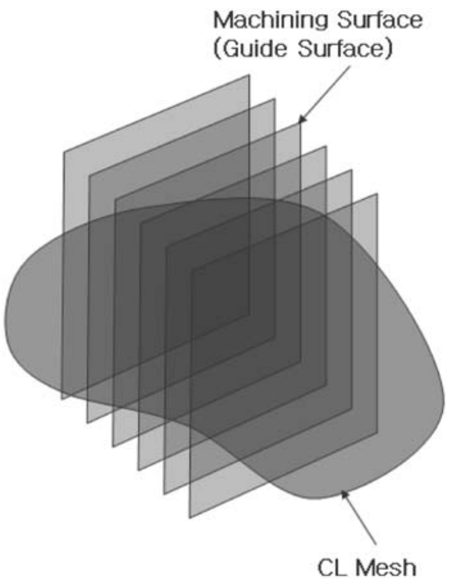

Fig. 9. Intersection between the CL-mesh and the machining surfaces

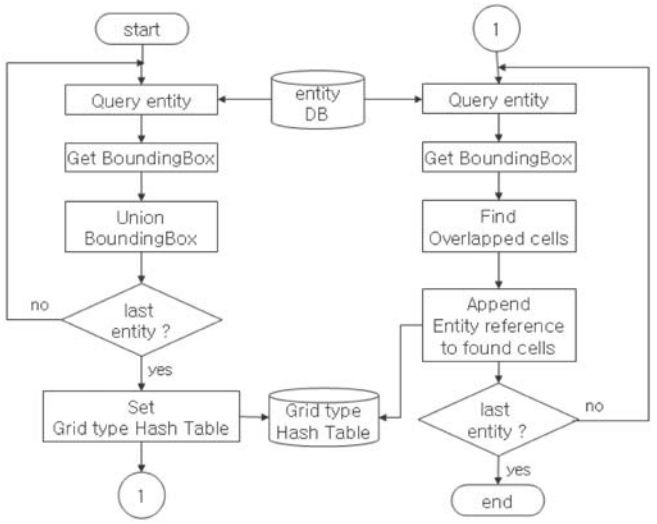

Fig. 10. Flow chart for the procedure of storing entities by using the grid method

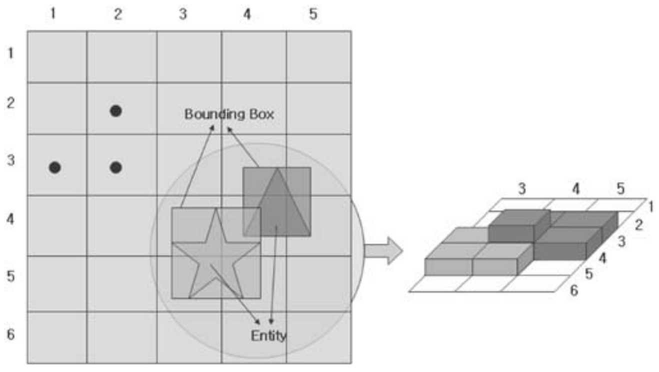

Fig. 11. Example of storing entities

(5, 3), and (5, 4), and a triangle-shaped entity is stored in cells (3, 4), (3, 5), (4, 4), and (4, 5).

When all entities are stored, the range searching process is conducted as follows: Fig. 12a shows a flow chart for the procedure of the range searching. When an entity is given, the bounding box of the entity is calculated and the cells that overlap with the bounding box can be gathered. The candidate entities that intersect with the given entity can be obtained by referencing the entity list of these cells. As shown in Fig. 12b, a line is given and the bounding box of the line is calculated. The bounding box overlaps with cells (4, 3), (4, 4), (4, 5), (5, 3), (5, 4), and (5, 5), and the entities which are stored in these cells are gathered. In this example, a star-shaped entity and a triangle-shaped entity are candidates which can intersect with the given line. This method is simple, but it shows good performance for a range search for tool path generation when there is no topological information.

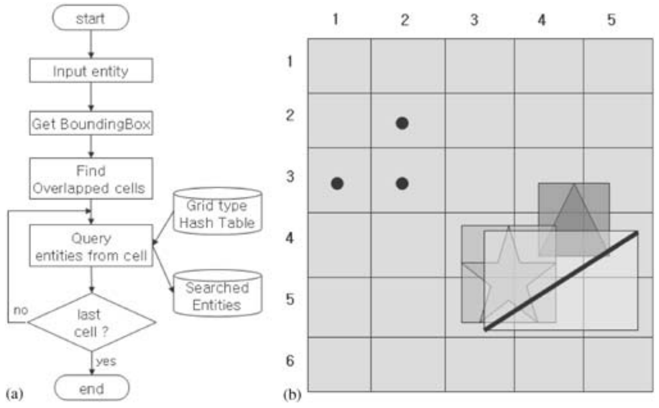

Fig. 12a,b. Querying of desired entities from grid. **a** Flow chart. **b** Example

4.2 Tool path generation

When machining surfaces are determined, the surfaces are divided into small planes and the intersection segments between CL-meshes and each divided plane are calculated using the aforementioned grid method. Because CL-meshes are composed of triangles or quad facets, the calculated intersection segments are lines. As shown in Fig. 6, the facets of CL-meshes overlap in the boundary region and the calculated lines are overlapped.

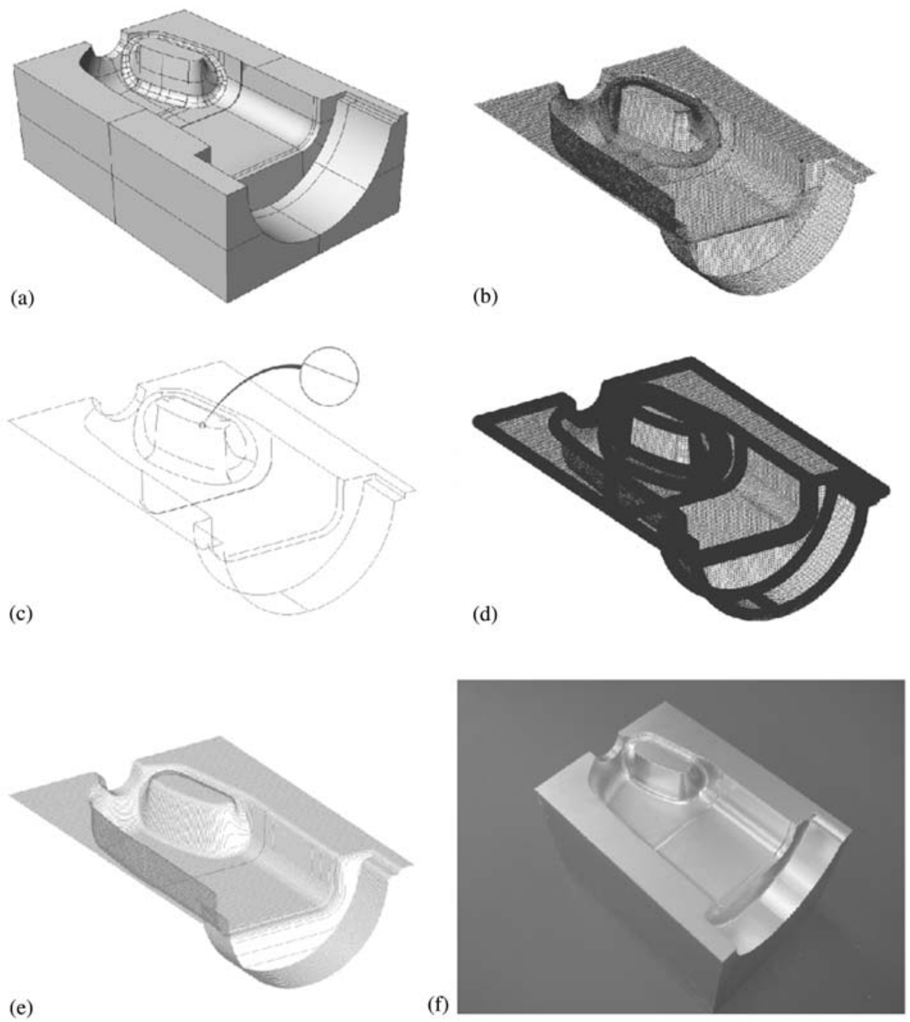

Fig. 13a–f. Examples of machining the cavity of a bottle mold. **a** Original model (IGES format, 16 surfaces). **b** Tessellated model. **c** Searched boundary edges. **d** Generated CL-meshes. **e** Calculated tool paths. **f** Machined part

To make gouge-free tool paths for three-axis machining, we apply the so-called "highest segment method." That is, the highest segments are chosen as tool paths, and if two or more of these segments intersect, the lower one is trimmed out.

5 Implementation and results

The proposed method of generating tool paths from an incomplete two-manifold mesh model is fully implemented on a PC. Figures 13a and b show the cavity part of a bottle mold and its tessellated model. It has 25 meshes and 42 247 facets and its dimensions are $140 \times 100 \times 50$ mm. The boundary edges are searched from the tessellated model, and the two-manifold conditions are violated in these boundaries, as shown in Fig. 13c. In the upper right corner of the figure, an enlarged partial view of the boundary edges marked with a circle shows that the adjacent two edges do not coincide with each other. As shown in Fig. 13d, the CL-meshes for the example model are generated by applying the proposed offset methods. Figure 13e shows the calculated tool paths, and Fig. 13f shows the actual machined part with the generated tool paths.

6 Conclusions

This study attempted to develop effective algorithms for tool path generation for incomplete two-manifold mesh models. Compared to existing polyhedral machining, the proposed method can be used to generate tool paths from an incomplete two-manifold mesh model, which is generally obtained when a designed part is transferred to another system and the transferred surfaces are tessellated. In the proposed approach, CL-meshes were generated and then sliced by the vertical machining planes in order to generate one-way tool paths. The intersecting segments were sorted, trimmed and linked using the highest segment method. To reduce computing time, a grid method was used to store and query the geometric entities. The proposed algorithms were fully implemented and demonstrated by illustrative examples.

References

1. Choi BK, Jerard RV (1998) Sculptured surface machining. Kluwer, Dordrecht
2. Lee SG, Yang SH (2002) CNC tool-path planning for high-speed high-resolution machining using a new tool-path calculation algorithm. Int J Adv Manuf Technol 20(5):326–333
3. Lee EK (2003) Contour offset approach to spiral toolpath generation with constant scallop height. Comput Aid 35(6):511–518
4. Choi BK, Jun CS (1989) Ball-end cutter interference avoidance in NC machining of sculptured surfaces. Comput Aid 21(6):371–378
5. Zhou L, Lin YJ (2001) An effective global gouge detection in toolpath planning for freedom surface machining. Int J Adv Manuf Technol 18(7):461–473
6. Han Z, Yang D (1998) Optimal tool selection for interference-free sculptured surface machining. In: Machining Impossible Shapes – International Conference on Sculptured Surface Machining, pp 247–262
7. Choi BK, Kim DH, Jerard RB (1997) C-space approach to tool-path generation for die and mold machining. Comput Aid 29(9):657–669
8. Duncan J-P, Mair SG (1983) Sculptured surfaces in engineering and medicine. Cambridge University Press, Cambridge
9. Jun CS, Kim DS, Park SH (2002) A new curve-based approach to polyhedral machining. Comput Aid 34(5):379–389
10. Hwang JS, Chang TC (1998) Three-axis machining of compound surfaces using flat and filleted endmills. Comput Aid 30(8):641–647
11. Inui M (2003) Fast inverse offset computation using polygon rendering hardware. Comput Aid 35(2):181–201
12. Piegl LA, Richard AM (1995) Tessellating trimmed NURBS surfaces. Comput Aid 27(1):6–26
13. Hamann B, Tsai PY (1996) A tessellation algorithm for the representation of trimmed NURBS surfaces with arbitrary trimming curves. Comput Aid 28(6):461–472
14. Goult RJ, Shearar PA (1990) Improving the performance of neutral file data transfers. Springer, Berlin Heidelberg New York
15. Maekawa T (1999) An overview of offset curves and surfaces. Comput Aid 31:195–173
16. Qu X, Stucker B (2003) A 3D surface offset method for STL-format models. Rapid Prototyp J 9(3):133–141
17. Ar S, Chazelle B, Tal A (2000) Self-customized BSP trees for collision detection. Comput Geom 15(1-3):91–102
18. Laszlo M-J (1996) Computational geometry and computer graphics in C++. Prentice-Hall, New York