Development of Reverse Engineering System for Machine Engineering Using 3D Bit-map Data

Tatsuro Yashiki* and Tarou Takagi*

*Power & Industrial Systems R&D Laboratory, Hitachi, Ltd.

Abstract

In this paper, the authors describe a reverse engineering system, named BitCAT, for machine engineering using 3D bit-map data obtained from CT digitizers. BitCAT can compose B-Rep CAD models by fitting parametric primitives, such as flat planes and cylinders, onto 3D bit-map surfaces. To find boundaries of the primitives effectively, a novel interactive "growing surface method" is developed and applied in the system. BitCAT is also provided with a method to determine geometric constraints between the primitives during composition processes. BitCAT can make very useful CAD models for manipulation and NC machining, because they are properly attached with geometrical features. A demonstration verifies the faculty and usefulness of the system.

1. Introduction

In many areas of machine engineering, reverse engineering techniques can be used to create CAD models of real objects for which no CAD model exists. Reverse engineering typically starts with digitizing real objects so that surface or solid models can be deduced in order to exploit the advantages of CAD/CAM technologies. To digitize real objects, high energy X-ray computed tomography (CT) is very useful, because they can digitize whole shapes including interior regions accurately and automatically in a very short time [1].

X-ray CT apparatuses obtain sectional images as two-dimensional (2D) bit-map data. After real objects have been digitized by X-ray CT, they are represented as piled-up X-ray CT sectional images, that is three-dimensional (3D) bit-map data. On the other hand, surface or solid models that conventional CAD systems handle are represented as boundary representation (B-rep) data [2]. Figure 1 shows the difference between B-rep and 3D bit-map formats. A 3D B-rep datum is represented as a set of geometrical and topological definitions of the shape elements like vertices, edges, faces, loops and shells. On the other hand, a 3D bit-map datum is represented as a three-dimensional array of tiny cells called "voxels" (volume picture cells). Each voxel holds inside/outside discrimination or relative density at its position. Compatibility is poor between bit-map data and B-rep data.

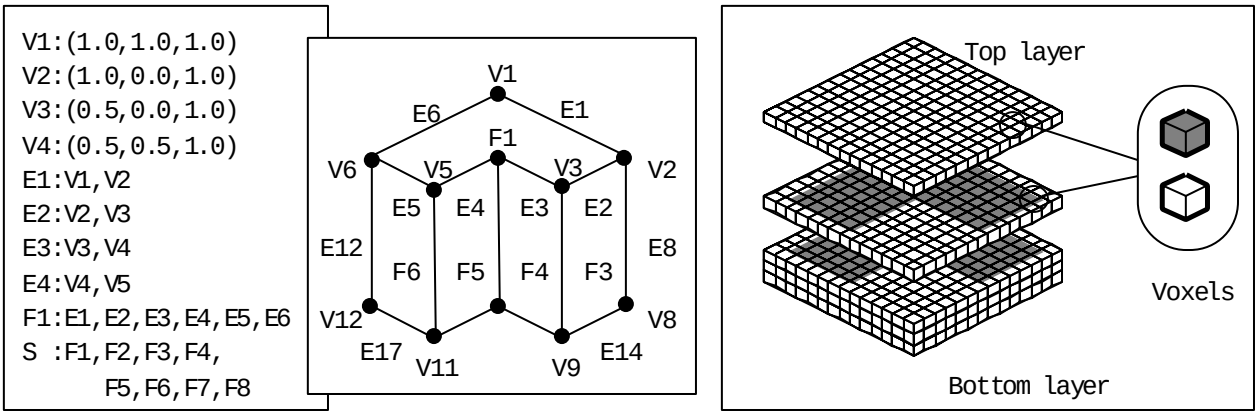


Figure 1. B-rep and bit-map formats

The purpose of this study is to construct B-rep CAD models from 3D bit-map data obtained from X-ray CT apparatuses. For this purpose, the authors developed a reverse engineering system for machine engineering, named “BitCAT”, which can make very useful B-rep models for CAD manipulation and NC machining. In this paper the authors describe the algorithm for creating B-rep CAD models and give results of an example conversion.

2. Related work

Many commercial software packages for reverse engineering create B-rep CAD models from 3D point cloud data obtained from 3D digitizers by the following procedures: (i) constructing triangulated meshes from 3D point cloud data; (ii) constructing a curve network, which divides a geometry defined by triangulated meshes into four-sided patches; and (iii) fitting NURBS surfaces onto each patch.

Many mechanical parts include parametric primitives, such as flat planes, cylinders and swept surfaces. Because the B-rep CAD models created according to the above approach are composed of NURBS surfaces only, parametric primitives in mechanical parts are represented as NURBS surfaces. This causes the following problems.

1. Because the data size required for handling a NURBS surface in CAD systems is much larger than that of a parametric primitive, CAD operations for such B-rep models may need a very long processing time.
2. Making modifications for NURBS surfaces are much more difficult than for parametric primitives. For example, to change the diameter of a cylindrical hole represented as NURBS surfaces, the user must move many control points suitably.
3. It is more difficult to generate NC tool paths for NURBS surfaces than for parametric primitives.

To overcome these problems, the authors suggest the following approach: (i) extracting

regions from 3D bit-map data to which appropriate parametric primitives can be fitted; (ii) fitting parametric primitives to the extracted regions; and (iii) fitting NURBS surfaces to the regions where no appropriate parametric primitives can be fitted. Because parametric primitives are represented just as they are in the B-rep CAD models created by the authors' approach, the B-rep CAD models are very useful for CAD manipulation and NC machining.

3. Algorithm for creating B-Rep CAD models

In this study, the authors developed a reverse engineering system for machine engineering, BitCAT, which can construct B-rep CAD models from 3D bit-map data. The algorithm for creating B-Rep CAD models is shown in Figure 2. First 3D bit-map surfaces are created from 3D bit-map data. Next regions are extracted from 3D bit-map surfaces to which appropriate parametric primitives can be fitted. Parametric primitives are fitted to the extracted regions. Then NURBS surfaces are applied on the regions where no appropriate parametric primitives are fitted. Then topological information of the B-Rep CAD models is created. Finally, the operator specifies geometric relationships among parametric primitives and refits them so that the specified geometric relationships can be satisfied. In some steps, interactive operations are required. Details of the algorithm in each step are described below.

3.1 Creating 3D bit-map surfaces

In this stage, each voxel of 3D bit-map data is converted to a 1-bit voxel, which holds inside/outside discrimination, after thresholding has been applied. With a geometry represented as

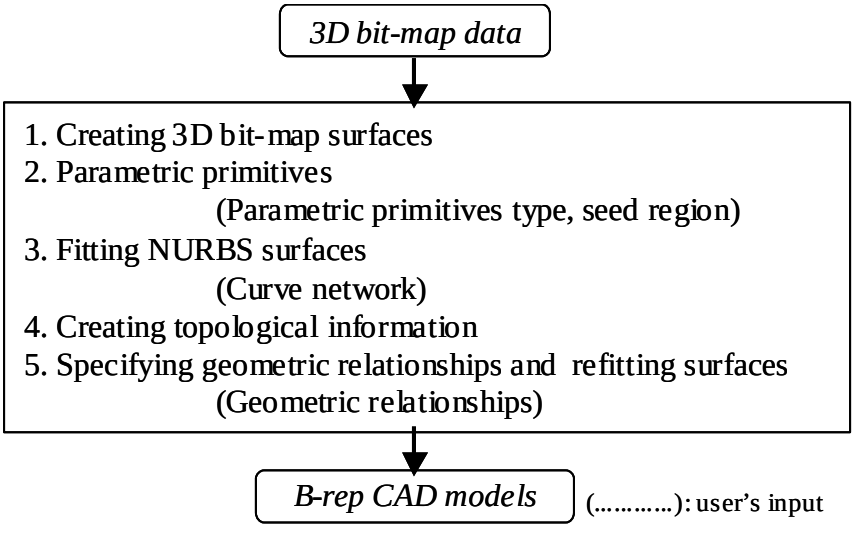


Figure 2. Algorithm for creating B-Rep CAD models

the 3D bit-map data of the 1-bit voxel, the surfaces are composed of pixels on which an inside

voxel is adjacent to an outside voxel (Figure 3). Such pixels are called “surface pixels”. Surface pixels are extracted from 3D bit-map data and 3D bit-map surfaces are created.

3.2 Fitting parametric primitives

After 3D bit-map surfaces have been created, regions are extracted from 3D bit-map surfaces to which appropriate parametric primitives can be fitted. Then parametric primitives are fitted to the extracted regions. To accomplish fitting parametric primitives, the following three problems must be solved at the same time: segmentation of 3D bit-map surfaces, determination of parametric primitive type and parametric primitive fitting [3].

To solve these three problems, the authors developed a novel interactive “growing surface method”. In the growing surface method, a user specifies the parametric primitive type and seed region on 3D bit-map surfaces interactively using the BitCAT GUI interface. Then segmentation and fitting proceed automatically with the following procedures.

First a surface of given parametric primitive type is fitted to the seed region. This surface which is called the “reference surface” is used to estimate how well a surface pixel matches the regions. Next the current region is growing by adding a surface pixel if the following requirements are satisfied.

1. The surface pixel is adjacent to the current region.
2. The perpendicular distance between the surface pixel and the reference surface is within the threshold of ***D***.
3. The angle between the surface normal at the surface pixel and reference surface normal at the surface pixel position is within the threshold of ***S***. The surface at the surface pixel is estimated by least-squares fitting of a plane to the surface pixel and the neighboring ones.

The region growing procedure is continued until no more surface pixels can be added to the current region. Then the reference surface is refitted to the current region. The above two procedures of region growing and surface refitting are iterated mutually until the current region do not grow any further. Finally the parametric primitive is fitted to the current region.

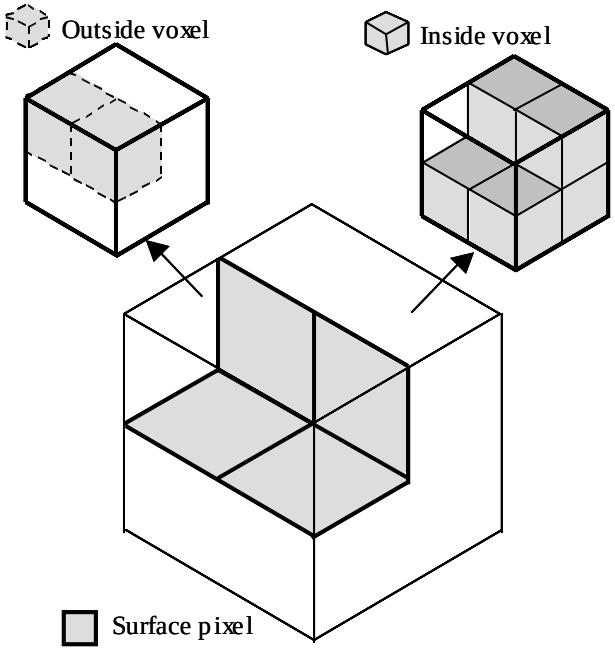


Figure 3. Definition of surface pixels

In the authors' current implementation, flat planes and cylinders are possible parametric primitive types. To fit parametric primitives, a least-squares surface fitting is performed [4].

3.3 Fitting NURBS surfaces

If there are regions where appropriate parametric primitives are not fitted, NURBS surfaces are applied on the regions instead of them. This task is done by the following procedures: (i) constructing a curve network interactively using the BitCAT GUI interface, which divides regions into four-sided patches; and (ii) fitting NURBS surfaces on each patch [5].

3.4 Creating topological information

In this stage, topological information of B-Rep CAD models is created. First boundaries of the divided regions on the 3D bit-map surfaces are extracted. By tracing the divided regions along the boundaries, it is known how the divided regions are connected to each other. In the object shown in Figure 4(a), for example, *region 1* has one outer boundary and one inner boundary. By tracing *region 1* along the two boundaries, it can be found that *region 1* is connected to *region 2*, *region 3*,

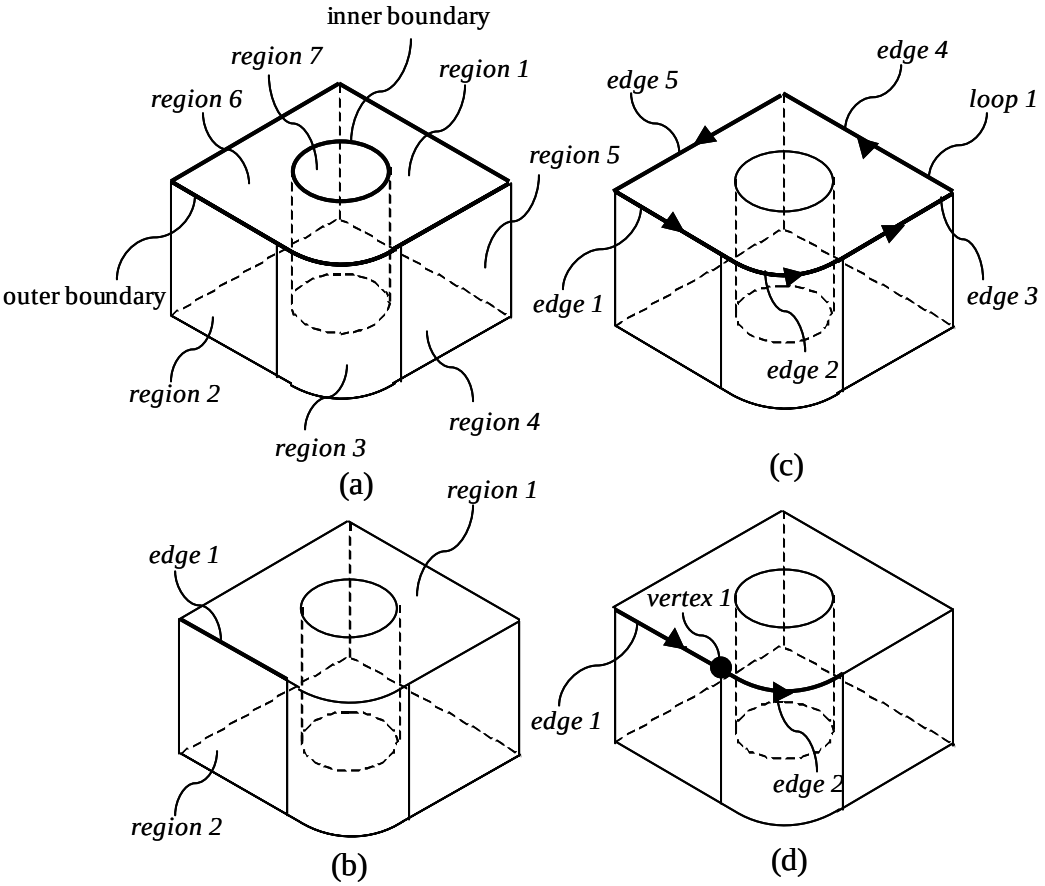


Figure 4. Example of creating topological information

region 4, *region 5* and *region 6* around the outer boundary and *region 7* around the inner boundary.

The edges can be defined by calculating intersecting curves between two adjacent regions. This calculation is done according to information on the connection among the divided regions. In the above example, *edge 1* can be defined by calculating the intersecting curve between *region 1* and *region 2* (Figure 4 (b)).

Then by linking the edges in the counterclockwise direction along the outer boundaries and in the clockwise direction along the inner boundaries, the loops can be defined. As shown in figure 4 (c), for example, *loop 1* can be defined by linking *edge 1*, *edge 2*, *edge 3*, *edge 4* and *edge 5* in the counterclockwise direction along the outer boundary.

Next the vertices can be defined by calculating intersecting points between two linked edges along the loops. For example, *vertex 1* can be defined by calculating the intersecting point between *edge 1* and *edge 2*, which are linked along *loop 1* (Figure 4 (d)).

3.5 Specifying geometric relationships and refitting surfaces

In many cases, mechanical parts have some geometric relationships among parametric primitives. For example, in the object shown in Figure 5, *cylinder 1* is connected to *plane 1* and *plane 2* smoothly (C1 continuity). *Plane 3* is perpendicular to *cylinder 1*, *plane 1* and *plane 2*.

Because each parametric primitive is fitted individually in the stage described in section 3.2, there are no geometric relationships among them yet. Then geometric relationships among parametric primitives are specified and the parametric primitives are refitted so that the specified geometric relationships can be satisfied.

In the authors' systems, a user specifies geometric relationships among parametric primitives interactively using the BitCAT GUI interface. Then parametric primitives are refitted. Geometric relationships among parametric primitives can be represented as constraint functions for the parameters defining the shapes. For example, the relationship that two flat planes are perpendicular to each other can be represented as a constraint function for the flat planes' normal vectors as follows:

$$\vec{n}_1 \cdot \vec{n}_2 = 0$$

where $\vec{n}_1$ and $\vec{n}_2$ are the normal vectors of the two flat planes. Then parametric primitives can be gotten which satisfy the

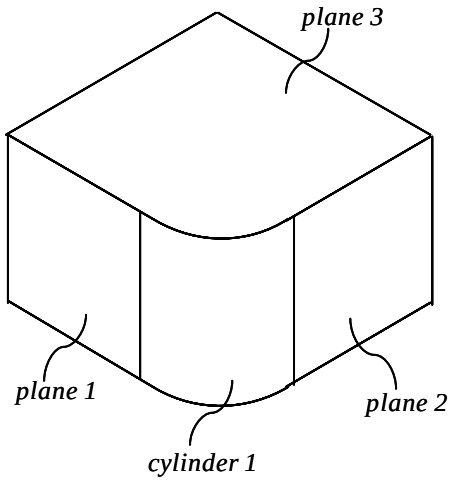
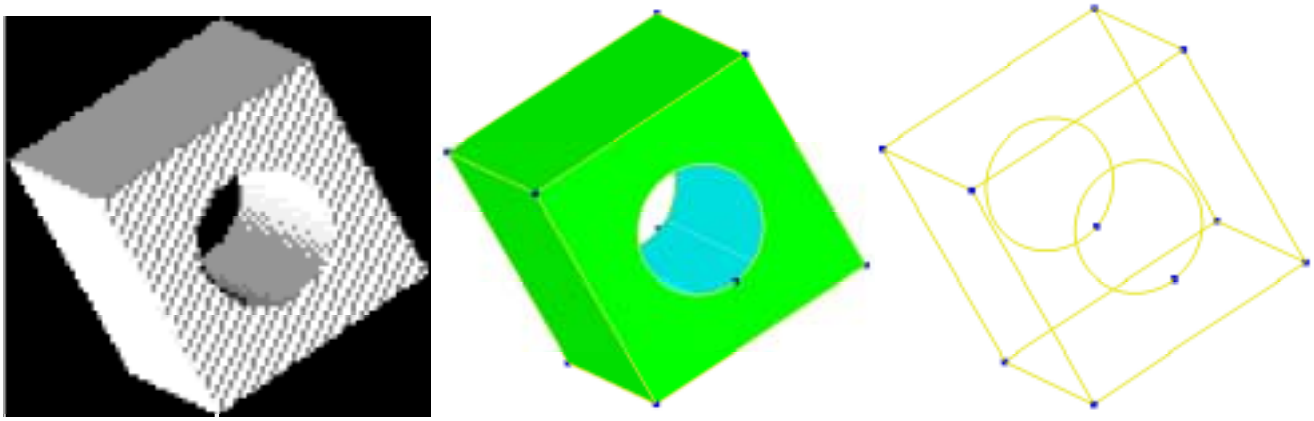
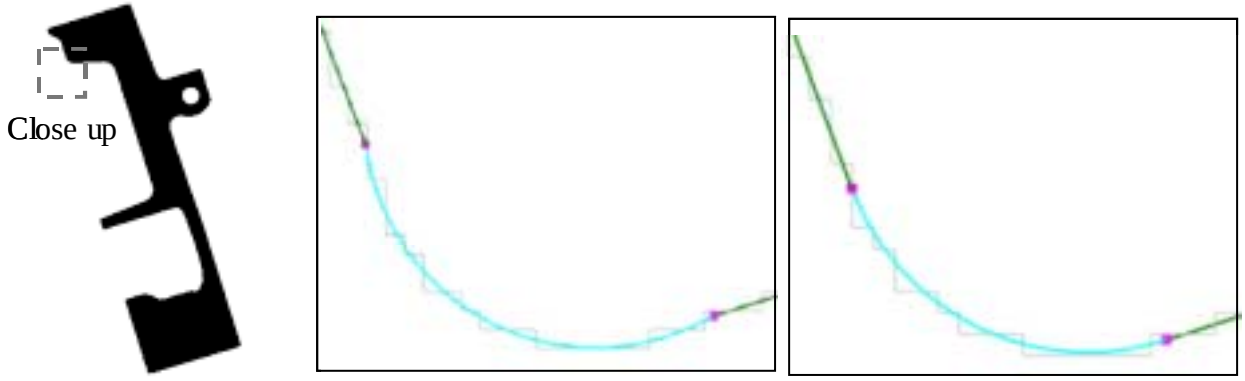


Figure 5. Example of geometric relationships among parametric primitives



(a) (b) (c)

Figure 6. Example of BitCAT applied to 3D bit-map datum



(a) (b) (c)

Figure 7. Example of BitCAT applied to 2D bit-map datum

specified geometric relationships by fitting them with some constraint functions. Mathematically this task is equivalent to dealing with a constrained non-linear optimization problem. In this study, a quasi-Newton method [6] and Lagrange multiplier method [7] are used to solve this optimization problem.

4. Example

The authors have applied the BitCAT system to some 2D / 3D bit-map data in order to confirm the validity of the algorithms for creating B-Rep CAD models. The results are shown in this section.

First to test fitting parametric primitives and creating topological information procedures, the 3D bit-map datum shown in Figure 6 (a) was used. This 3D bit-map datum was created by adding the standard deviation of the noise to the synthetic data. The data size was 100x100x100 voxels. Figure 6 (b) shows the complete B-Rep CAD models created by BitCAT and Figure 6 (c) shows the edges and vertices.

Next to test specifying geometric relationships and surfaces refitting procedures, the 2D bit-map datum shown in Figure 7 (a) was used. This 2D bit-map datum was obtained from X-ray CT apparatuses. Figure 7 (b) shows parametric primitives (lines and circles) fitted to the 2D bit-map datum, where no geometric relationships were specified. The authors specified C1 continuity connections between parametric primitives. The results are shown in Figure 7 (c). Every parametric primitive was connected with C1 continuity. This result is in 2D, but the algorithms can be extended to 3D with no difficulty.

5. Conclusion

The conclusions of this paper are summarized below;

1. The authors have developed a reverse engineering system for machine engineering, named "BitCAT", in order to construct B-rep CAD models from 3D bit-map data obtained from X-ray CT apparatuses.
2. BitCAT can compose B-Rep CAD models by fitting parametric primitives, such as flat planes and cylinders. BitCAT is also provided with a method to determine geometric constraints between the primitives during composition processes. By using these functions, BitCAT can make very useful CAD models for manipulation and NC machining.
3. BitCAT has been applied to some 2D / 3D bit-map data and the validity of the algorithms was confirmed.

References

[1] J. H Stanley and R. N. Yancey, CT-assisted Agile Manufacturing, Proceedings of SPIE, Nondestructive Evaluation for Process Control in Manufacturing (1996).

[2] J. D. Foley, A. Dam, S. K. Feiner and J. F. Hughes, Computer Graphics, Addison-Wesley (1990)

[3] T. Varady, R. R. Martin and J. Cox, Creating Geometric Models in Reverse Engineering, RECCAD Deliverable Document 1(1996)

[4] G. Lukacs, A. D. Marshall and R. R. Martin, Geometric least-squares fitting of spheres, cylinders, cones and tori, RECCAD Deliverable Documents 2 and 3(1997)

[5] L. Piegl and W. Tiller, The NURBS Book, Springer (1997)

[6] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, Numerical Recipes in C, Cambridge (1992)

[7] D. P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Academic Press (1982)