International Journal of Image and Graphics
© World Scientific Publishing Company

ENUMERATION OF CONTOUR CORRESPONDENCE

SHIGERU OWADA
Department of Information Science, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654 Japan
ohwada@is.s.u-tokyo.ac.jp

YOSHIHISA SHINAGAWA
Beckman Institute and Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign
405 North Mathews Avenue Urbana, Illinois 61801 U.S.A.
sinagawa@uiuc.edu

FRANK NIELSEN
Sony Computer Science Laboratories, Inc.
3-4-13 Higashi-Gotanda, Shinagawa-ku, Tokyo 141-0022 Japan
nielsen@csl.sony.co.jp

Received (29 June 2002)
Revised (2 December 2002)
Accepted (15 January 2003)

With the recent advances in computed tomography and magnetic resonance devices, cross-sectional images are now commonly used for diagnosis. However, how contours between cross-sections should be connected is often ambiguous. In this paper, we propose an algorithm that enumerates all possible cases of the correspondence of contours. This is useful for achieving fully automatic interpolation of contours, although our current implementation still requires some degree of manual interaction.

Keywords: Contour interpolation; reconstruction; Reeb graph; singular points; topology.

1. Introduction

Computed tomography (CT) and magnetic resonance (MR) devices enable us to easily obtain cross-sectional images from physical objects. To visualize such data in three dimensions (3D), volume rendering is often used.[12] Volume rendering, however, usually requires quite dense cross sections and a large amount of storage. Therefore, simpler methods, such as surface rendering, are still used for visualization.

When polygonal meshes are constructed from sparse cross-sectional contours, each interspace between adjacent cross-sectional planes has to be filled by interpolation. If the shape of the object is simple, the interpolation is trivial. However,

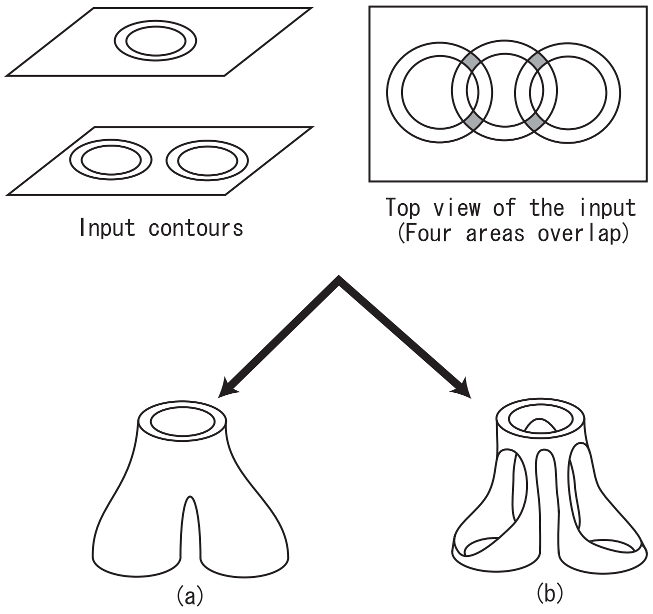


Fig. 1. Ambiguity in interpolating contours where the interpolated object could be either a bifurcating pipe (a) or four pillars (b)

handling real-world medical data, such as human skeleton or brain, becomes difficult as the spacing between slices increases and when the numbers of contours in adjacent cross sections differ.

There has been much previous work in this field.[3][4][5] However, to the best of the authors' knowledge, the existing work focuses mainly on how to obtain a smooth transition rather than on the correspondence of topological structures. For example, in Fig.1, it is not easy to determine the correct topological structure for the interspace. Suppose that the six contours are obtained from a blood vessel. One may speculate that two streams meet between the two slices as shown in Fig.1(a). However, if *shape-based interpolation*[4] is used, for example, the result is four pillars, as shown in Fig.1(b) (arising from the four areas of overlap in a top view).

It is usually difficult to determine topological structure without additional knowledge about the object. In Fig.1, the four pillars could actually be the correct interpolation for some other physical object.

Although topological structure is important, a user's knowledge is not limited to topology. The topology of an object describes just one part of the full shape information - the skeleton information. Other knowledge may be specified by means of the integral of the surface curvature, volume of the object, or its interference with other objects' surfaces, but these quantities cannot be calculated and compared with possible cases until the surface has been reconstructed.

However, the strategy of first determining the topological structure, then constructing the surface, and finally optimizing the shape to match the user's knowledge tends to find local minima, because the optimization is usually performed to improve the shape of the surface, rather than to change the topology. Even if topological change is allowed, without scanning every possible case, it is hard to find an answer that is sufficiently close to the global optimum, because topological changes strongly and unpredictably affect the above-mentioned quantities.

In this paper, we propose a method that enumerates all possible topological structures. A polygonal mesh is then constructed for each result in the enumeration. This algorithm forms a basis for calculating the global optimum, namely the shape that best matches the user's knowledge.

2. Background

Over two decades have been devoted to research on interpolation of contours.[63]

Although this work did not deal with bifurcation, an important idea was introduced by Fuchs et al.[3] Namely, the problem of finding the correspondence between points on contours can be reduced to the problem of finding the minimum cost path in a directed toroidal graph. Later work has taken bifurcation into account. The original toroidal graph was extended to deal with bifurcations and holes by Shantz.[7] Shinagawa et al. extended the toroidal graph to be continuous.[8] Christiansen et al. proposed an algorithm based on the connection of the nearest points.[9] Ekoule et al. proposed another method that handles highly complex bifurcations and convex contours.[10] The Delaunay triangulations can also be used for solving the correspondence problem.[11] These methods typically use contours defined by pieces of straight lines as input, calculate the correspondence between junction points, and output triangulated meshes.

There is yet another approach where a 2D function is defined for each cross section and is interpolated in 3D. In this framework, each cross-sectional contour is first converted to a binary image and then converted to a grayscale image where the intensity of a pixel is computed as the distance from the contours.[412] It calculates the gray value as the shortest signed distance from the contours (a positive value for the interior of the contours and a negative for the exterior) and is linearly interpolated in 3D. The final surface is then extracted as the isosurface of the 3D field distance function. Recently, shape-based interpolation has been further extended by using information about the correspondence of contours.[5] Distance field manipulation[13] is similar to these approaches.

If the spacing between slices is quite narrow, it is possible to solve the contour interpolation problem as the interpolation of unorganized points.[1415] In this scenario, each contour is converted to a set of unorganized points having no connectivity information. There is some work involved in the reconstruction from this set of unorganized points, mainly in the context of approximating the object surface obtained by range scanners or stereo matching algorithms.

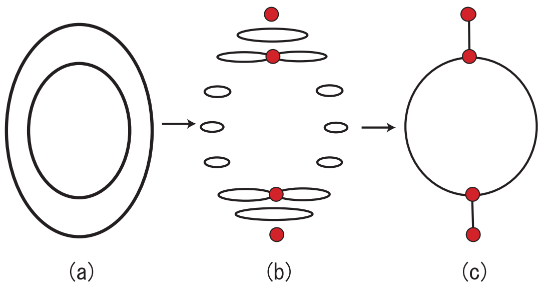


Fig. 2. Construction of a Reeb graph where the original object (a) is sliced (b) and the corresponding Reeb graph (c) is constructed by handling each contour as a point

Other approaches include work by Cong et al., in which a numerical solver was used that directly calculates the functional value in 3D.[16] Yet another method uses singularity and the distance between contours in determining bifurcation,[17] which has similarities to our algorithm. However, there is still a possibility with this method of rejecting a correct result, since it only uses the distances between contours and the genus as the knowledge of an object. Our method just outputs possible results, giving further information about an object such as the curvature limit or the volume of the closed area.

The remainder of the paper is as follows. Notation is defined and our algorithm to enumerate all the possible connectivity patterns is presented in Section 3. In Section 4, we show several results and Section 5 concludes the paper.

3. Algorithm

3.1. *Notation*

3.1.1. *Reeb graphs*

The first tool we use is the so-called *Reeb graph*. A Reeb graph is a graph that gives a simple representation of the topology (bifurcation status) of a continuous function defined onto an object. One of the simplest functions is the height function h. The height function h simply returns the height (e.g. the z value) of the point of the surface.

A Reeb graph of an object is built as follows. As depicted in Fig.2, the object is sliced and each cross-sectional contour is represented as a point of the Reeb graph. The points where two (or more) contours meet or a contour disappears are called the *singular points* (see Fig.3). Mathematically, the singular points are defined as the points where

$$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial y} = 0 \tag{1}$$

holds (see Morse theory).

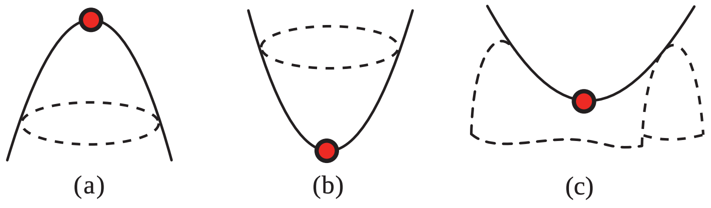


Fig. 3. Singular points used in this paper: a peak point (a), a pit point (b), and a saddle point (c)

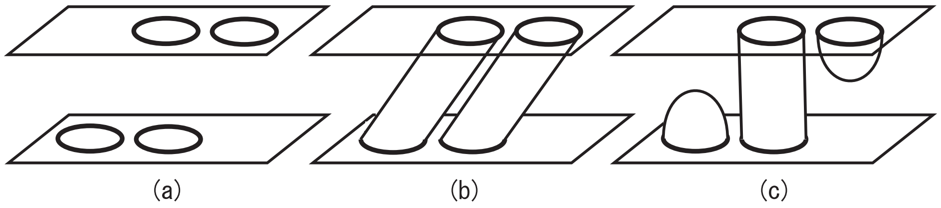


Fig. 4. An example in which the number of singular points is important where the input contours(a) are interpolated either using general assumptions only(b), or with knowledge about the number of singular points(c)

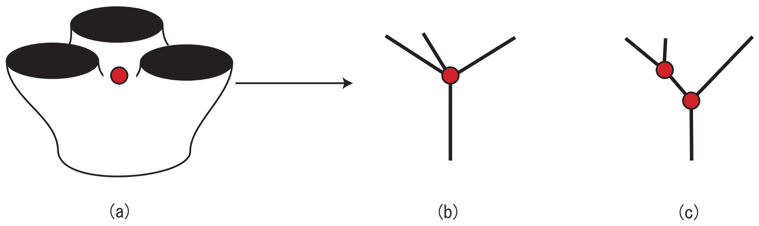


Fig. 5. An example of a degenerate singular point (a) where the singular point connects three contours to one contour (b) which can also be interpreted as the combination of two 2-to-1 singular points that coincide at this point (c)

The importance of the number of singular points is as follows. If the input contours are as shown in Fig.4(a), then from traditional assumptions of bifurcations and from the distance between contours, the topology shown in Fig.4(b) results. However, if the user specifies knowledge about the number of singular points, (in this case, “two”,) the topology shown in Fig.4 (c) should be the correct answer.

The singular points are represented as the nodes of the Reeb graph.[17] Singular points where more than two contours meet are said to be *degenerate* (see Fig.5). These degenerate singular points can be decomposed into a sequence of simple 2-to-1 singular points. Although it is possible to use other functions,[18] the Reeb graph of the height function is sufficient in this paper.

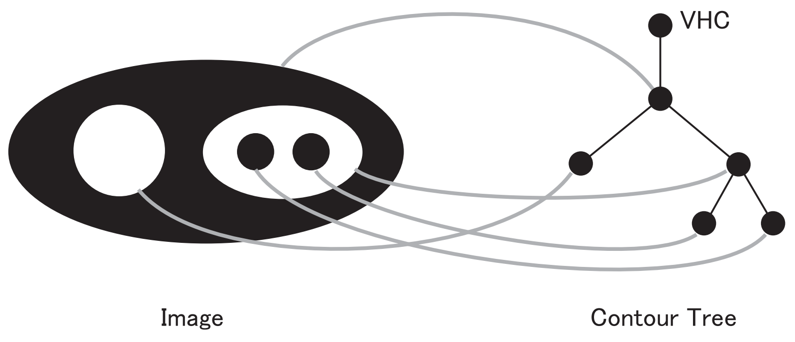


Fig. 6. An example of a contour tree where each contour in an image corresponds to a node in the contour tree and a parent-child relationship in the contour tree implies a circumscription relationship in the image. The root node of the contour tree is the VHC

3.1.2. *Contour trees*

A *contour tree* represents the circumscription relationship of contours in a slice, and similar notions have been used previously [19]. Each node in a contour tree represents one contour and each edge represents the circumscription relationship between two contours. A contour circumscribes all contours which are its descendants in the contour tree. In general, there is more than one separate object in an image. In this case, more than one connected component exists in a contour tree. For convenience, we assume that there is a virtual contour that circumscribes all the contours in the image and the corresponding root node is added to the contour tree. Such a contour is called a *Virtual Hollow Contour (VHC)* (see Fig.6). A singular point exists at the height where the topology of the contour tree changes.[20]

3.2. *Outline of the proposed algorithm*

We assume that contours are already extracted from input images and the contour trees are built. The outline of the algorithm is as follows.

- Transform a contour tree of each slice by the operations that merge or eliminate nodes. This corresponds to the transformation of the Reeb graph. Each elementary transformation generates a singular point. At this stage, all possible transformation patterns are enumerated. We limit ourselves to the cases where nodes are only merged or eliminated rather than divided or added because the increase in the number of nodes in these cases leads to an infinite number of solutions. This restriction is further discussed later on.
- Two of the transformed contour trees are compared, each of which is one of the enumerated contour trees from adjacent slices. If both trees have the same structure, the transformation operations applied in the previous stage are validated.

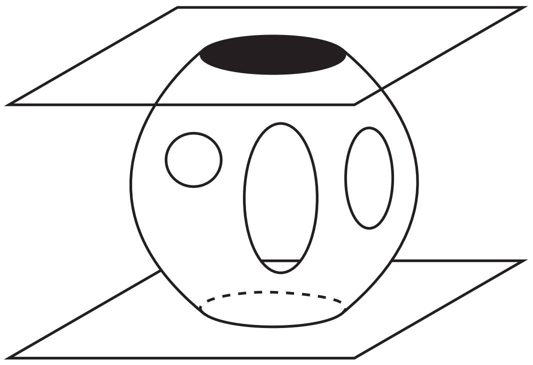


Fig. 7. An example of an unnatural result caused by an increase in the number of contours

3.2.1. *Transformation of a contour tree*

According to the Morse theory, there are three types of non-degenerate singular points. In what follows, we denote them by e^0,e^1 and e^2 using the same notations as the cells corresponding to the singular points.[20] There are twelve types of Morse operators defined.[20] If operations that increase the number of contours are allowed, the number of possible Reeb graphs becomes infinite. Moreover, it causes an unnatural result as in Fig.7, where topological structure in the interspace is complicated (more specifically, when the object is sliced by a plane at the middle of the original upper and lower cross sections, the number of cross-sectional contours is larger than that on either of the original planes.)

For this reason, we assume the number of contours will never increase at the enumeration stage. Thus, we adopt only three types of Morse operators. The singular point e^0 is considered to be the point where a contour disappears as shown in Fig.8(a) (if seen from top to bottom); i.e., this removes a node in a contour tree. The corresponding operator is **E0**. The singular point e^1 is a point where two contours are connected. Since there are two types of operations in the case of the singular point e^1 (that is, it connects either sibling contours as shown in Fig.8(b) or the parent and child contours as shown in Fig.8(c)), the corresponding two operators (**E1_SI** and **E1_PC**) are defined. Note the difference between the singular points and the Morse operators. E0 is the operator that corresponds to the singular point e^0, whereas E1_SI and E1_PC correspond to e^1.

When one of the three operators is applied to a part of a contour tree, the remaining part of the contour tree is affected accordingly.[20] If E0 is applied, all the descendant nodes must be removed simultaneously to avoid self intersection that never occurs in the case of natural solid objects. When E1_SI is applied, the descendants of the connected two contours are simply merged and become the descendants of the newly created node (see Fig.9). The E1_PC case is more complicated (see Fig.10). In this case, contour **a**(parent) and contour **b**(child) are connected. The

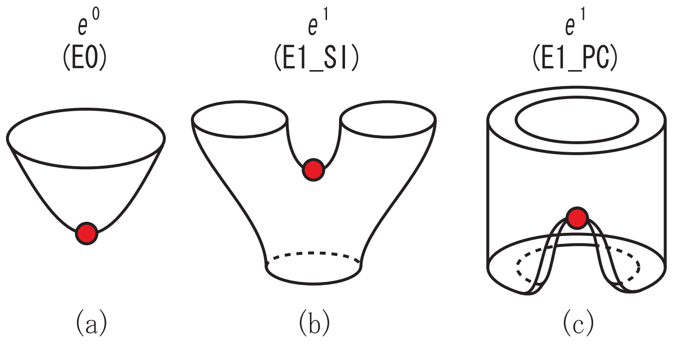


Fig. 8. Three types of Morse operators which correspond to the singular points e^0 and e^1 where e^0 removes a contour and e^1 connects two contours

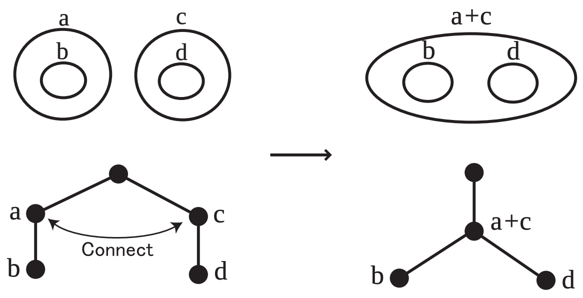


Fig. 9. An example of E1_SI where descendant nodes **b** and **d** are merged after the connection

newly created contour **a+b** and the descendants of **b** (**c**,**d**) become siblings. The general case is shown in Fig.11.

3.2.2. *Enumeration*

Enumerating every possible correspondence of contours amounts to enumerating all the possible transformations of contour trees and matching them, which in turn amounts to seeking all possible combinations of the aforementioned operators. A problem is the complex behavior of the remaining parts of the tree when the operators are applied. We propose the following algorithm.

(1) Applying E0
At first, we enumerate all the combinations of E0. To do this, we traverse the contour tree from the root(VHC) and set a flag at each node indicating whether to apply E0 in breadth-first order or not. As an exception, the root node (VHC) is never marked. If a node is marked as "apply E0", all the descendants are deleted. An example of the result at this stage is shown in Fig.12, where the

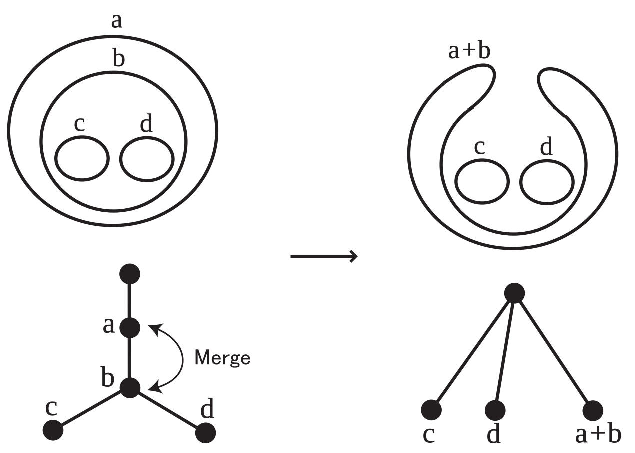


Fig. 10. An example of E1_PC where after the contour **a** and **b** are merged, **c**,**d** and **a+b** become sibling nodes

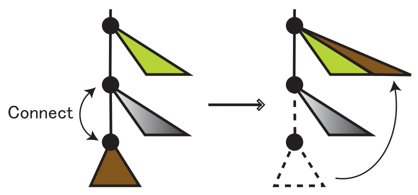


Fig. 11. When a parent and a child are connected by E1_PC, the descendant nodes of the child become sibling nodes of the parent node.

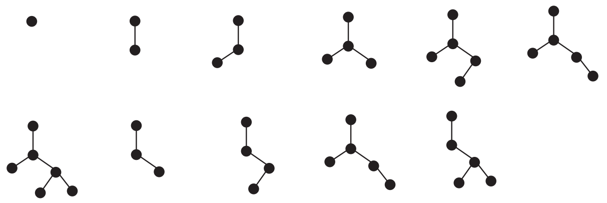


Fig. 12. All the enumerated patterns by the application of E0 to the contour tree shown in Fig.6

contour tree shown in Fig.6 is used as input.

(2) Applying E1_SI and E1_PC

Next, E1_SI and E1_PC are applied. These operators merge nodes in the contour tree. As we stated previously, E1_PC creates a complex structure in the remaining part of the tree. To handle this, we again mark the nodes. At this stage, both operators connect nodes several times and eventually a certain num-

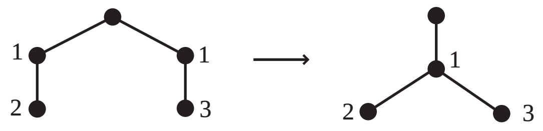


Fig. 13. An example of E1_SI application by indexing, where both children of VHC have index "**1**" and these two nodes are connected by E1_SI

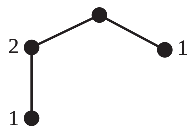


Fig. 14. Impossible assignment of indices where two nodes with index "**1**" are contained and these nodes cannot be connected because they are neither siblings nor parent-child nodes

ber of nodes remain, regardless of the applied operators. To describe this, we assign the same index number to the original nodes which merge to one node. Fig.13 depicts an example of the correspondence between indexing and the actual application of the operators. In this case, nodes that have index "**1**" are connected. The applied operator is E1_SI because the original two nodes with index "**1**" are siblings. Every time a merge operation is applied, it is necessary to move the other part of the tree according to the type of the operator. This is repeated until no two neighboring nodes have the same index. If any two nodes still have the same index, the index assignment is invalid (e.g., Fig.14). This test is performed for every combination of index assignments.
By converting the combination of operators to the indexing problem, the order of operators is lost; e.g., the difference between Fig.15(a) and Fig.15(b) is not distinguished. However, we do not regard this difference as important because in many cases, the height of the singular points depends on what kind of smoothing algorithm is used in the final mesh reconstruction. In this paper, the constructed Reeb graph is just used for determining the initial mesh. The final surface shape is not necessarily the same as the original Reeb graph.

3.2.3. *Matching the contour trees*

After the possible contour trees are enumerated, the two trees enumerated from adjacent cross sections are compared. If they have identical structure, the two contour trees can be matched. In this case no more transformation of contour trees (singular points) is necessary to connect the contours.

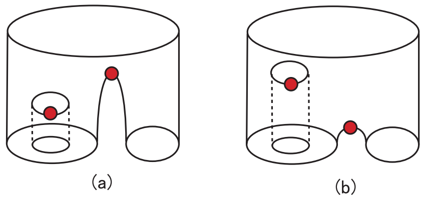


Fig. 15. Examples of singular points at different heights

4. Implementation

4.1. *Initial mesh construction*

Once the correspondence of contours is determined, the surface of the object is constructed. Although construction of the surface is not essential in this paper, we propose a simple procedure here. At first, the position of each singular point is calculated by the following two steps.

(1) Determine the two dimensional positions of the singular points on the cross-sectional plane. If the singular point is e^0, the position is simply the center of gravity of the contour connected to the singular point. If the type of singular point is e^1, the position is calculated as the center of the closest two points[9] (see Fig.16).
(2) Determine the height of each singular point within the corresponding interspace. The height order of singular points corresponds to the application order of the Morse operators (see section 3.2). The singular points are located evenly in the height direction retaining the height order (see Fig.17). Note the height difference between singular points is calculated for each connected component of the Reeb graph within an interspace.

Finally, the mesh is built. It is an easy and well-studied problem because all the bifurcation points and the corresponding contour shapes are already calculated. It means that the topological shape is already known and only the one-to-one correspondence of contours should be considered. In addition, because newly created contours have similar shapes to the neighboring contours as shown in Fig.16, it is trivial to find the correspondence of points between the new contour and the neighbors. When the contours on adjacent slices vary widely in shape, we use the toroidal graph[3] to calculate the correspondence.

4.2. *Experimental Results*

We have implemented an interactive topology selector as shown in Fig.18. When the contour data is input (Fig.19 (a)), possible Reeb graphs are automatically enumerated. This software also has an ability to sort the enumerated results using an objective function. This function is designed to minimize the number of singular

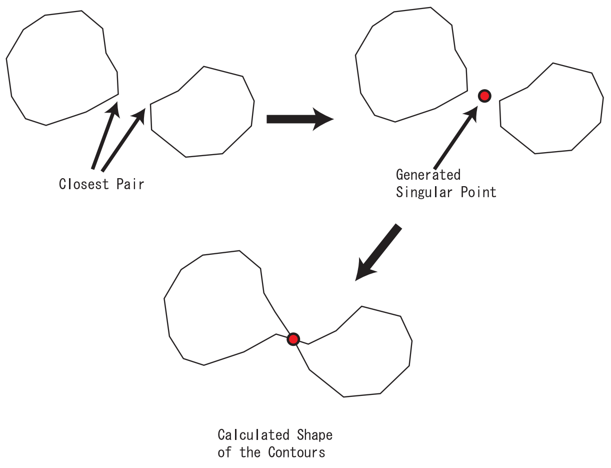


Fig. 16. Calculation of contour shapes where a new singular point is generated at the center of the closest pair of contours

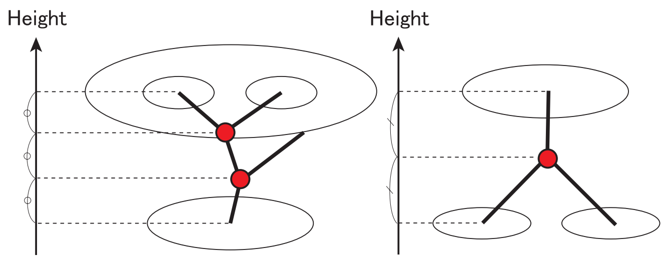


Fig. 17. Calculation of the height of each singular point, where each singular point is located evenly in the height direction retaining the height order within the connected component

points[21] and the distance between connected contours. If the Reeb graph at the top of the list is not satisfactory, the user can select another Reeb graph (Fig.19 (b)). After the topology is set, an initial mesh is constructed (Fig.19 (c)). Finally the mesh is smoothed and output (Fig.19 (d)).

We used the Loop subdivision scheme[22] to smooth the mesh. The final shape does not exactly "interpolate" the original contour data because the Loop subdivision scheme is an "approximating" subdivision scheme in comparison to "interpolating" subdivision schemes such as the Butterfly or the Kobblet schemes . We do not consider this problem crucial in this paper, but if the final surface must strictly passes through the input contours, interpolating subdivision schemes or, alternatively, other smoothing methods such as the global thin plate energy minimizing method[23] or the non-shrinking Gaussian smoothing method[24] should be used.

All the enumerated patterns when contour images with four slices are input are

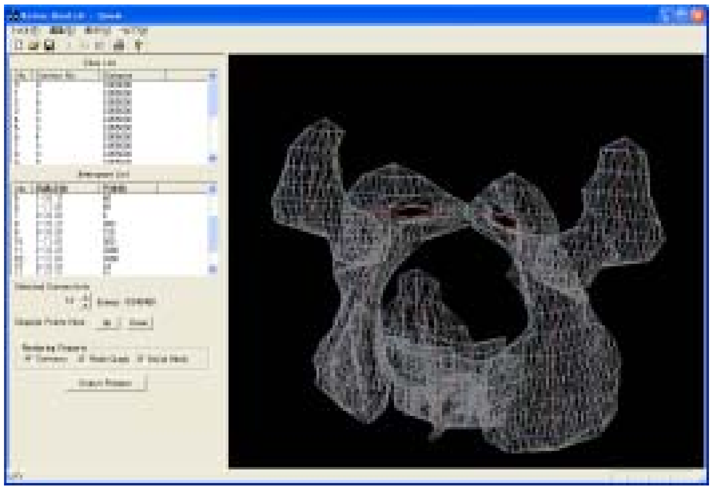


Fig. 18. A screen shot of our interactive topology selector

shown in Fig.20. In this example, the topmost image and the bottommost image contain no contour while the second image contains four contours and the third image contains two contours. In this case, only seven patterns are possible (Fig.20 (a)-(g)). The final mesh is rendered as translucent polygons. (e) and (f) may need more explanation. The difference between (c) and (e) is that in (e), the top-right small contour is connected to its exterior contour while in (c), it is deleted. The difference between (d) and (f) is the same.

Fig.21 shows the results of our method. The four columns show the input contours, the Reeb graph, the reconstructed initial mesh and the final smooth mesh, respectively. In the pelvis and the bronchus data, the contours are extracted manually by Bézier curves and then converted to point lists. Table 1 and 2 show the enumeration results. The leftmost column shows the indices of the interspace. The second column is the number of contours in the slices at the upper and the lower regions of the interspace. The third column is the number of correct singular points. The fourth column is the number of the enumerated Reeb graphs. The last column is the manually chosen correct indeces of the Reeb graph. If this number is 0, the top candidate of the sorted result was correct and no manual selection was required. In most cases, this is just the smallest possible number under the given input contours which can automatically be calculated. However in other cases, this is manually specified.

5. Discussion and Future Work

We have proposed an algorithm to enumerate every possible correspondence of contours when interpolating cross-sectional images. This allows us to explicitly

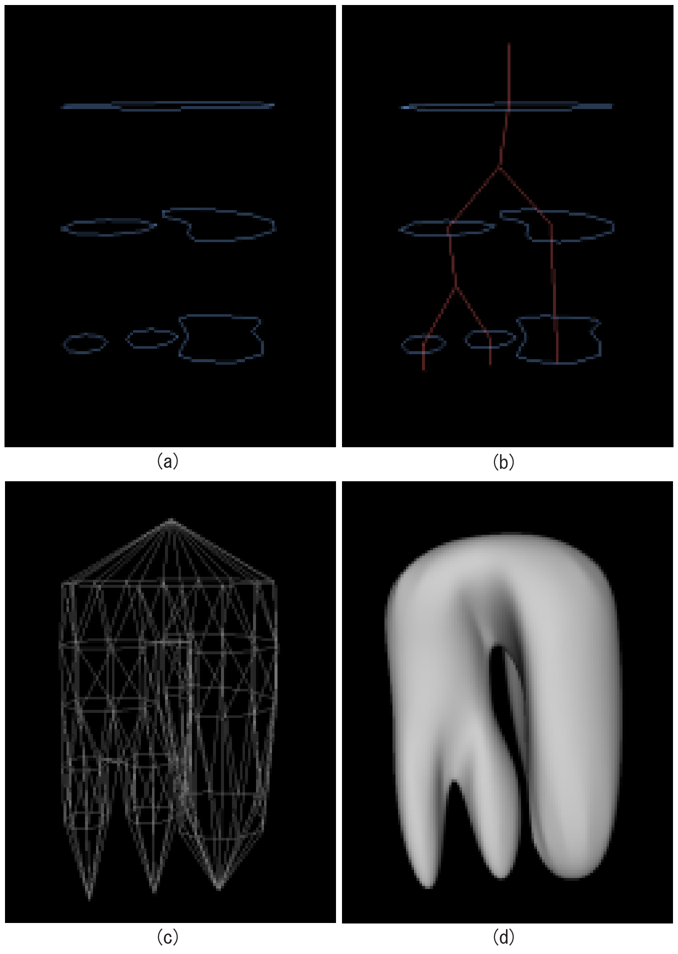


Fig. 19. An example of input contours (a), the Reeb graph which has the smallest objective function value (b), the constructed initial mesh (c), and the smoothed surface (d)

handle topological ambiguity and avoids falling into local minima by finding the answer which best matches the user's knowledge about the object.

However, the current implementation has some room for further improvements in order to achieve a fully automatic reconstruction. To begin with, we need to find an appropriate method to model the user's knowledge about an object. Our final goal is to develop a system which outputs a correct result without any interaction by the user. Although the current implementation automatically optimizes the number of singular points and average distances between contours, it is not, by itself, sufficient to achieve our goal. One possible improvement of the algorithm is the consideration of more global knowledge such as the entire integral of the surface curvature or the number of bifurcations or holes. A more straightforward approach

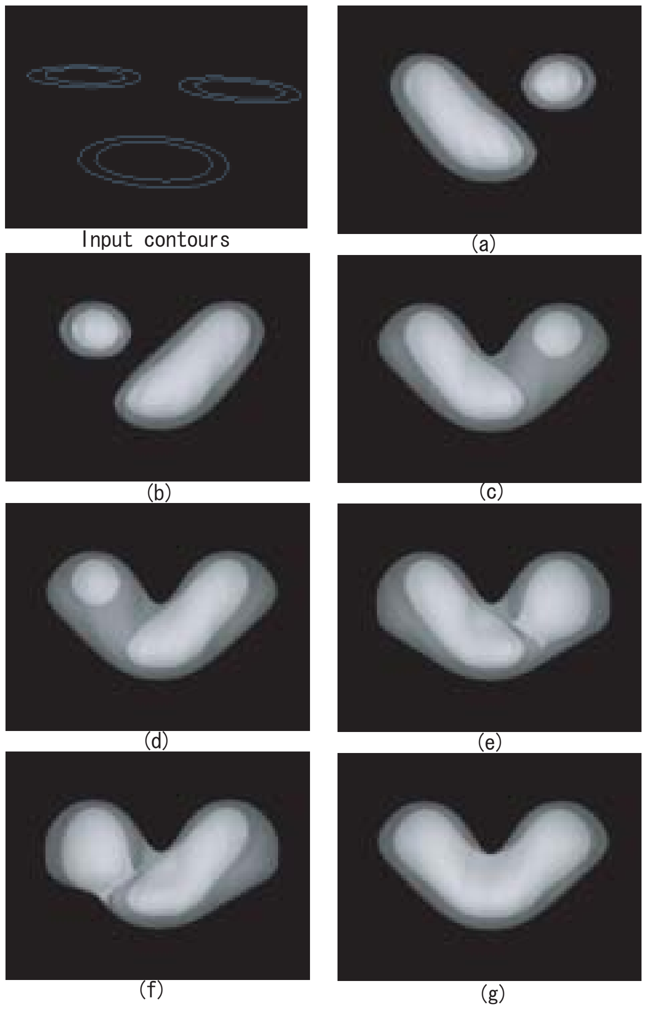


Fig. 20. Seven enumerated results from six contours on four slices rendered by translucent polygons

would be to input a complete 3D model into the system as the knowledge about an object, which is the so-called model fitting. In this framework, our enumeration strategy is also useful to avoid finding local minima.

In our algorithm, every contour is used during calculations. To handle larger size data , the grouping of contours will be essential because of its computational complexity. If the average number of contours in a slice is N, the upper bound on the number of enumerated cases in an interspace is $O(N^{2N})$.

Finally, how to handle contours on planes which are not parallel is not discussed in this paper. However, our method can easily be applied, provided that each cross section has only two adjacent cross-sectional slices on each side.

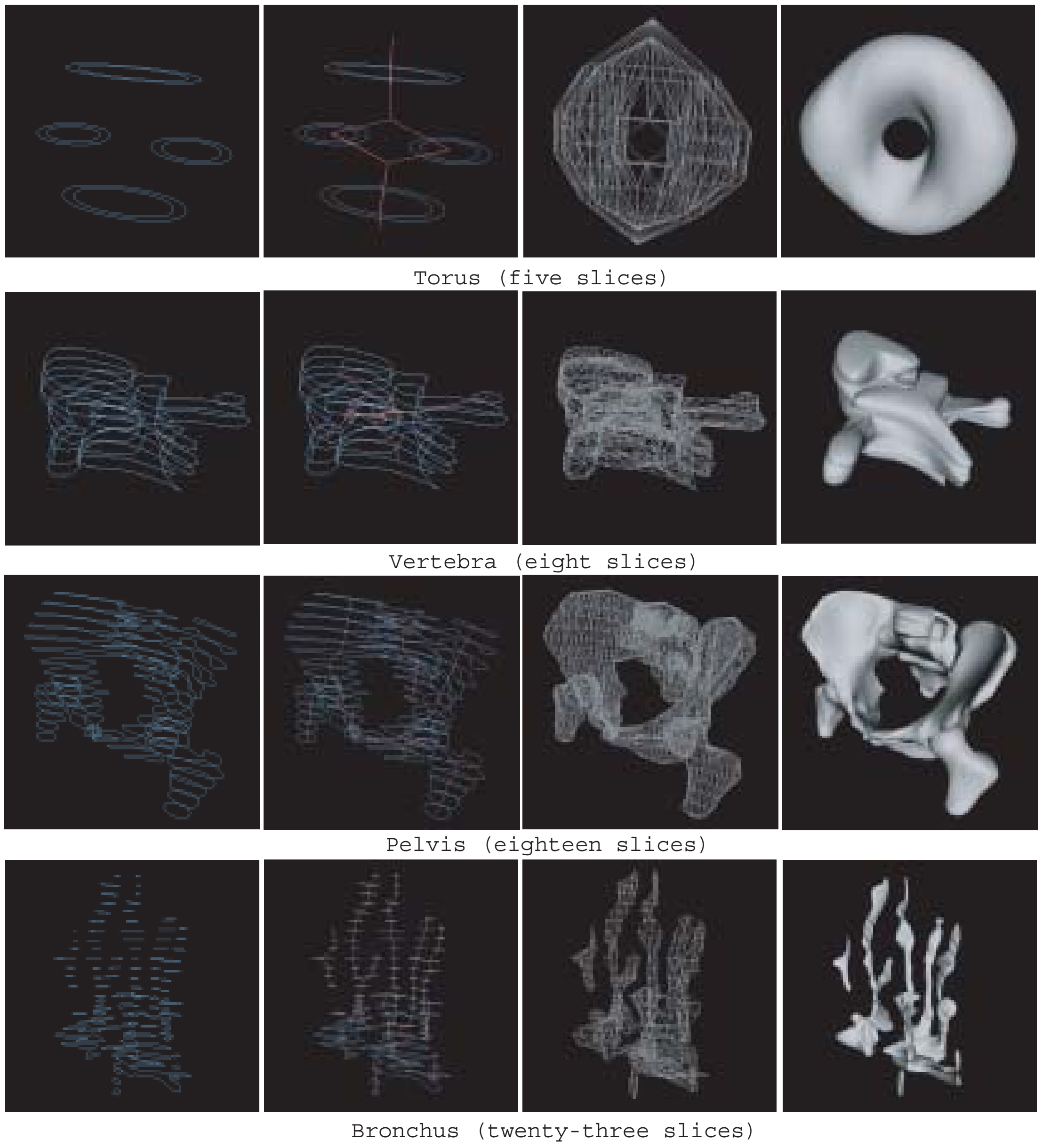


Fig. 21. Results of the application of our algorithm to various data

Acknowledgments

The experimental results use the datasets from *the Visible Human Project* (*http://www.nlm.nih.gov/research/visible/visible_human.html*) . We would also like to

Table 1. The result of enumeration for pelvis data

Id.	#Contours	#Singularities	#Candidates	Correct Id.
0	0-3	3	1	0
1	3-6	3	2100	35
2	6-3	3	2100	63
3	3-3	0	6	0
4	3-3	0	6	0
5	3-4	1	60	2
6	4-3	1	60	1
7	3-3	0	6	0
8	3-5	2	390	5
9	5-5	0	120	0
10	5-4	1	360	0
11	4-6	2	3360	36
12	6-4	2	3360	14
13	4-4	0	24	0
14	4-2	2	50	0
15	2-2	0	2	0
16	2-0	2	1	0

thank Michael J. Cibulskis for his quick and precise correction of the grammatical errors in the draft paper.

References

1. R. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics (SIGGRAPH '88 Proc.)*, 22:65–74, 1988.
2. M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.
3. H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.
4. S. P. Raya and J. K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9(1):32–42, 1990.
5. G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Surface interpolation for sparse cross sections using region correspondence. In *Medical Image Understanding and Analysis 1999*, 1999.
6. E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1):2–11, 1975.
7. M. Shantz. Surface definition for branching, contour-defined objects. *Computer Graphics (SIGGRAPH '81 Proc.)*, 15:242–270, 1981.
8. Y. Shinagawa and T. L. Kunii. The homotopy model: a generalized model for smooth surface generation from cross sectional data. *The Visual Computer*, 7(2-3):72–86, 1991.

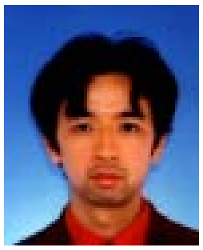
Table 2. The result of numeration for bronchus data

Id.	#Contours	#Singularities	#Candidates	Correct Id.
0	0-3	3	1	0
1	3-4	1	60	11
2	4-5	1	360	3
3	5-5	2	5400	8
4	5-7	4	378000	7
5	7-8	1	181440	2
6	8-7	1	181440	0
7	7-7	0	5040	0
8	7-6	3	670320	0
9	6-6	2	52920	1
10	6-6	0	720	0
11	6-6	0	720	0
12	6-5	1	2520	0
13	5-5	0	120	0
14	5-5	0	120	0
15	5-3	2	390	0
16	3-2	1	12	0
17	2-2	0	2	0
18	2-2	0	2	0
19	2-2	0	2	0
20	2-2	0	2	0
21	2-0	2	1	0

9. H. N. Christiansen and T. W. Sederberg. Conversion of complex contour line definition into polygonal element mosaics. *Computer Graphics (SIGGRAPH '78 Proc.)*, 12:187–192, 1978.
10. A. B. Ekoule, F. C. Peyrin, and C. L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, 10(2):182–199, 1991.
11. J. D. Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, 1988.
12. G. T. Herman, J. Zheng, and C. A. Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, 12(3):69–79, 1992.
13. B. Payne and A. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, 1992.
14. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (SIGGRAPH '92 Proc.)*, 26:71–78, 1992.
15. C. L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics (SIGGRAPH '95 Proc.)*, 29:109–118, 1995.

16. G. Cong and B. Parvin. A new regularized approach for contour morphing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:1458-1463, 2000.
17. Y. Shinagawa and T. L. Kunii. Constructing a Reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11(6):44–51, 1991.
18. M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics (SIGGRAPH '01 Proc.)*, 35:203–212, 2001.
19. H. J. A. M. Heijmans. Connected morphological operators for binary images. *Computer Vision and Image Understanding: CVIU*, 73(1):99–120, 1999.
20. Y. Shinagawa, Y. L. Kergosien, and T. L. Kunii. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 1991.
21. T. Haig, Y. Attikiouzel, and M. D. Alder. Border marriage: matching of contours of serial sections. *IEE Proceedings I*, 138(5):371–376, 1991.
22. C. Loop. Smooth subdivision surfaces based on triangles. *Master's thesis,University of Utah, Department of Mathematics*, 1987.
23. W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. *Computer Graphics (SIGGRAPH '94 Proc.)*, 28:247–256, 1994.
24. G. Taubin. A signal processing approach to fair surface design. *Computer Graphics (SIGGRAPH '95 Proc.)*, 29:351–358, 1995.

Photo and Bibliography



Shigeru Owada received his B.Sc and M.Sc. degrees in Department of Information Science, The University of Tokyo, Japan in 1998 and 2000, respectively. He worked as a research assistant in Sony Computer Science Laboratories, Tokyo, from 2000 to 2001. He was a visiting researcher at the University of Illinois in 2002. His interests include image processing, volume rendering, and volume modeling.

Yoshihisa Shinagawa received his B.Sc. (in 1987), M.Sc. (in 1990) and D.Sc. (in 1992) degrees in information science from the University of Tokyo. He is an associate professor in the Department of Electrical and Computer Engineering at the University of Illinois and a full-time faculty member in the Beckman Institute Artificial Intelligence Group. His fields of professional interest include computer graphics, and computer vision and its applications.

Frank Nielsen received his B.Sc. and M.Sc. degrees from École Normale Supérieure (ENS) of Lyon and Ph.D from INRIA Sophia-Antipolis in 1992, 1994, and 1996, respectively. In 1997, he served army as a scientific member in the computer science laboratory of École Polytechnique (LIX). In 1998, he joined Sony Computer Science Laboratories, Tokyo, Japan as a researcher. His current research interests include computational geometry, algorithmic vision, combinatorial optimization for geometric scenes and graphics.