A RECTIFICATION ALGORITHM FOR MANIFOLD BOUNDARY REPRESENTATION MODELS

Guoling Shen, Takis Sakkalis and Nicholas M. Patrikalakis
Massachusetts Institute of Technology
Cambridge, MA 02139–4307, USA
guoling@rentec.com, takis@aua.gr, nmp@mit.edu

Abstract In this paper we present a boundary reconstruction methodology which builds a valid model in the neighborhood of an object described by a traditional boundary representation model with floating point specification. This method converts an erroneous model into an interval model, guaranteed to be gap-free. An example illustrates our methodology for robust conversion.

Keywords: Manifold boundary model, defects, rectification methods

Introduction

Boundary representation (B-rep) contains topological and geometric information of solid boundaries. Topological information, in general, is represented by a graph describing incidence and adjacency relations between topological boundary entities. Geometric specification involves equations for points, curves and surfaces. A B-rep model is called *valid* if it describes a solid boundary. However, as is well-known, the validity of B-rep models is not self-guaranteed [4, 10, 12].

Invalid B-rep models contain defects – representational entities that do not conform to modeling constraints due to topological errors and/or inconsistencies between topological and geometric specification. Defects often appear as gaps, inappropriate intersections, dangling entities, internal walls and inconsistent orientations [17]. Causes of defects exist throughout the entire life cycle of a model, such as computational inaccuracy stemmed from floating point arithmetic [4, 5] and approximation algorithms in model creation and inconsistent conversions in data exchange across heterogeneous modeling systems [11].

Research on model rectification (correction) has been done mainly on triangulated models, specifically, STL models for rapid prototyping. Most algorithms [1, 9] identify erroneous triangle edges, string such edges to form hole boundaries, and then fill holes with triangles. These algorithms use local topology (incidence and adjacency) to rectify defects and are successful in the majority of candidate models, but may create undesirable global topological and geometric changes.

In our earlier work [16], we argued that the model rectification problem should be approached as a boundary reconstruction problem in order to achieve a global optimal solution, and showed that the boundary reconstruction problem is NP-hard. In order to construct gap-free B-rep models, we further developed the concept of interval solid models [15], which was first introduced by Hu et al [7, 8]. Based on these, in this paper, we develop a boundary reconstruction methodology. The objective is to reconstruct a B-rep model from a given B-rep model represented in a certain format (e.g. STEP) using interval arithmetic.

The methodology proposed in the following sections consists of three main steps: 1) construction of a graph induced by surface intersections for each surface, 2) face reconstruction, and 3) shell reconstruction. The methodology follows very closely the proof of NP-hardness of the boundary reconstruction problem in Shen et al [16]. It mainly focuses on capturing miniature numerical gaps and fills them with appropriate boxes.

In the following, we first describe the three steps of the procedure in detail. Then, we show an example to illustrate the use of interval methods in model verification and rectification.

1. Intersection-induced graph

Edges of a B-rep model are embedded on intersection curves of the underlying surfaces. On each of the surfaces, intersection curves form a geometric embedding of a graph with their intersections as nodes and the curve segments as arcs. As discussed in Shen et al [16], face boundaries must consist of arcs in this graph in order to achieve geometric consistency of all topological relations (incidence and adjacency) in which the face is involved.

In the proposed boundary reconstruction method, an intersection curve is computed using surface intersection algorithms developed by Hu et al [6], and is represented as an ordered list of non-degenerate rectangular boxes [14]. Therefore, the geometric embedding of an arc in the graph is an ordered list of boxes, called a *box curve*, and that of a node is a (unordered) cluster of boxes, called a *box point.* Whenever a new intersection curve is computed, the two graphs on the intersection surfaces need to be updated. The new curve is intersected with the geometries of all arcs and nodes in both graphs, and is subdivided into a list of box points and box curves, which are then inserted into both graphs to create new nodes and arcs.

2. Face reconstruction

Let R be the underlying surface of face f^o in model M^o, and G_f be the intersection-induced graph. As in model creation, boundary reconstruction proceeds in a bottom-up manner.

An edge e^o involved in a certain adjacency relation between f^o and another face f_1^o must be embedded in the intersection curve C of R and R_1. The reconstruction of e^o starts with its vertices v_1^o, v_2^o. The corresponding new vertices v_1^n, v_2^n are two nodes in G_f. Each of these nodes must have at least one incident arc constructed from C, i.e. the new vertices are on the intersection curve. In addition, each selected node should be the closest to the original position of the corresponding old vertex among all such nodes to minimize the geometric change in boundary reconstruction. See Figure 1 for illustration, where the dotted lines are arcs of G_f.

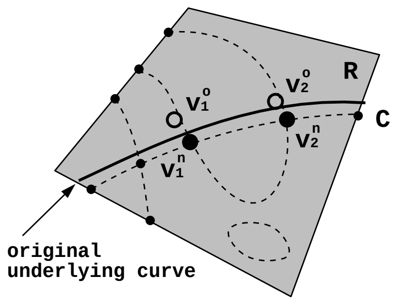

Figure 1. Reconstruction of vertices

The reconstruction of e^o amounts to finding a path between the two selected nodes in G_f. Again, the selected arcs must be constructed from the intersection curve C. If there exist multiple paths, the one with the minimum deviation from the original geometry of e^o should be selected. This often happens when C contains self-intersections or is closed, e.g. a circle. With the edge orientation given in e^o, it shall be clear which part of C belongs to an edge, unless the underlying curve of e^o in M^o is far different from C. Thus, new edge e^n is oriented in the same way as e^o.

For a loop l^o, new edges form a subgraph G_e in G_f, which may not be a simple closed curve. Though an arbitrary graph could be very complicated, the proposed method intends to resolve miniature features such as small dangling arcs and gaps. It first identifies two types of nodes in G_e: free node that has exactly one incident arc, and branch node that has more than two incident arcs. Dangling arcs are then identified by marching from a free node until another free node or a branch node is reached, and then trimmed away if the geometric change due to such actions is within a given tolerance. Large pieces of dangling arcs indicates the existence of gaps. Let v_1, v_2 be two ends of a gap. The shortest path between v_1, v_2 is then searched in G_l, where $G_l = G_f - (G_e - \{v_1, v_2\})$[1]. See Figure 2 for illustration. Finding the shortest path between two nodes in a graph is a polynomial problem. Algorithms can be found in many textbooks on graph theory, such as [2]. Edges in the newly constructed loop l^n are oriented in the same way as their corresponding old edges are in l^o. Consistency can be verified by marching through these edges.

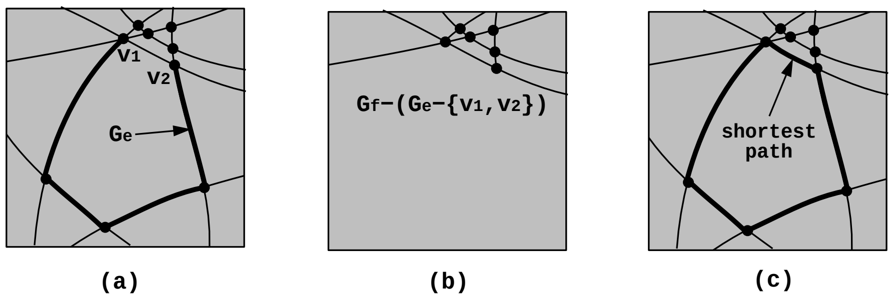

Figure 2. Gap closing in loop reconstruction

A face may have more than one loops. Loops reconstructed individually need to be modified so that together they define a valid face boundary. The modification process includes eliminating intersections between loops, verifying relative locations of loops, and orienting loops consistently. To retain the design intent of the face topology, the resulting loops should also be homeomorphic to the original face topological structure in the strong sense[2], provided that the latter is a graph consisting of simple cycles sharing at most one common node pairwise [13].

Any two loops share at most one common vertex. Assume two loops l_1^n and l_2^n share more than two vertices. We will resolve cases of two

[1]Here, the difference between graph $G(V, E)$ and its subgraph $G_1(V_1, E_1)$ is another subgraph consisting of nodes $V - V_1$ and arcs incident to these nodes.
[2]Two face topological structures are homeomorphic in the strong sense if they are homeomorphic and their outer loops are homeomorphic as well [16].

common vertices. Cases involving more than two common vertices may follow the same approach. Let v_1, v_2 be two common vertices of l_1^n and l_2^n. Let P_1 be the shorter path between v_1, v_2 in l_1^n, and P_2 be that in l_2^n. Also assume that the corresponding original loops l_1^o and l_2^o are topologically correct. The following is a brief description of the algorithm. See also Figure 3.

1 If both l_1^n and l_2^n are inner loops,

   (a) If P_1 is outside l_2^n, and P_2 is outside l_1^n,

      i If l_1^o and l_2^o do not have common vertices, first, delete v_1 and v_2 and their incident arcs in l_1^n, i.e. $l_1^n \leftarrow l_1^n - \{v_1, v_2\}$, and then, fill the gap in graph $G_f - l_1^n - l_2^n$. If the gap can not be filled, try the same in l_2^n.

      ii If l_1^o and l_2^o share one common vertex, first, delete v_1 and its incident arcs in l_1^n, i.e. $l_1^n \leftarrow l_1^n - \{v_1\}$, and then, fill the gap in graph $G_f - l_1^n - l_2^n$. If the gap can not be filled, try the same in l_2^n, or by deleting v_2.

      iii If l_1^o and l_2^o share one common edge, delete P_1, P_2 and merge l_1^n, l_2^n.

   (b) If P_1 is inside l_2^n, and P_2 is inside l_1^n, switch P_1 and P_2 and do the same as Step 1(a).

   (c) If P_1 is inside l_2^n while P_2 is outside l_1^n, delete P_1, P_2 and merge l_1^n, l_2^n.

2 If one, say l_1^n, is the outer loop,

   (a) If l_2^n is inside l_1^n, do the same as Step 1(a).

   (b) If one part of l_2^n is outside l_1^n, delete P_1, P_2 and merge l_1^n and l_2^n.

In Step 1(a) of the above algorithm, if l_1^n and l_2^n can not be modified in accordance with the topological structure of l_1^o and l_2^o, topological changes are then necessary. For example, if Step 1(a)i fails, we may try Step 1(a)ii by ignoring the topological structure. For two loops sharing a common edge, if the feature exists in the original topological structure, the two loops should be merged by trimming the edge; otherwise, the edge should be deleted from one of the loops and the induced hole needs to be closed as above.

Whether a part or an entire loop l_1^n is inside another loop l_2^n, can be verified by checking a box of l_1^n which does not intersect l_2^n. The ray casting algorithm should be applied to the pre-images in the parameter domain of the underlying surface. Let $\mathbf{R}_b$ be a horizontal or vertical

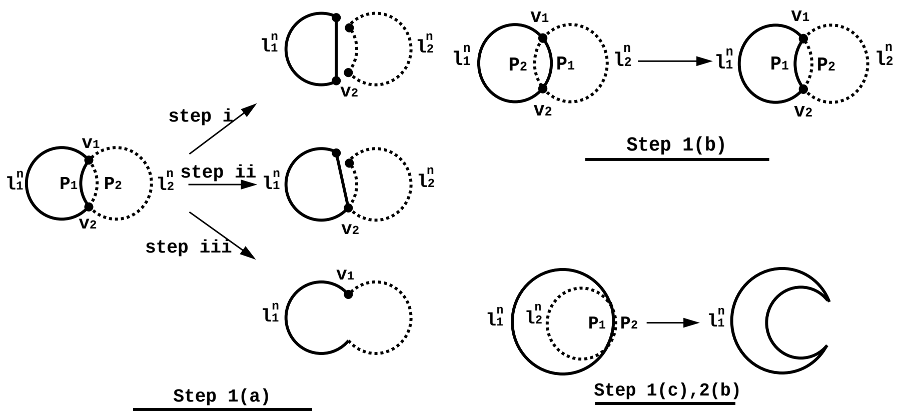

Figure 3. Loop modification

ray from a box b. Compute the intersections between $\mathbf{R}_b$ and boxes in l_2^n. The intersections are intervals, representing coordinate values in the direction of the ray. Merge all intersecting intervals. The number of remaining intervals is then the number of intersections. The same algorithm is also used to verify the relative locations of the loops. All inner loops must be inside the outer loop, and no inner loop is inside another inner loop.

The orientation of a loop can be determined by computing the rotation index [3] of its pre-image in the parameter domain. Select an arbitrary point $\mathbf{c}$ inside a loop. Let $\{\mathbf{c}_i\}$ be centers of boxes in the loop. Then, the rotation index is

$$r = \frac{1}{2\pi} \sum_i \angle(\mathbf{cc_{i-1}}, \mathbf{cc_i}), \tag{1}$$

where $\angle$ denotes the positive angle between two vectors. This number should be close to ± 1 if box sizes are small. A loop is positively oriented if $r \approx 1$, and is negatively oriented if $r \approx -1$. The outer loop must be positively oriented, and all inner loops must be negatively oriented.

3. Shell reconstruction

A shell consisting of faces reconstructed individually may have dangling patches, internal walls and holes, and may be inconsistently oriented. Similar to algorithms for STL model rectification, shell reconstruction identifies those edges shared by only one or more than two faces, and string them together to form simple closed curves. Such a simple closed curve may bound a hole if each of its edges has exactly one incident face, and bounds an internal wall if each of its edges has more than two incident faces. If it consists of two connected pieces, one having edges with one incident face and the other having edges with three incident faces, then, the closed curve bounds a piece of dangling patch. More complex situations similar to what were studied by Bohn and Wozny [1] could happen. Heuristic rules could help in such situations, but user assistance is frequently needed in resolving certain ambiguities. Here, we only deal with the above mentioned three situations, and leave other cases to user resolution.

Dangling patches with very small sizes shall be trimmed away first. The boundary of an internal wall actually bounds three connected pieces, each of which is an open shell. The one inside the shell formed by the other two is the internal wall and shall be deleted. Whether an open shell is inside a closed shell can be verified by a ray-casting algorithm in a similar manner as in loop reconstruction. In the following, we present a method for filling holes using surface patches constructed from the underlying surfaces by surface intersections. Let $\{e_i\}$ be the edges in a hole boundary and already sorted in order. The method fills the hole progressively by attaching new patches to the boundary and computing the new hole boundary. This is a trial-and-error process. It tests all possible combinations of patches until one is found to fill the hole. In the selection of patches, those with small sizes are preferable so as to minimize the geometric change. For an edge e in $\{e_i\}$, a patch shall be selected first if it is embedded on the same surface as the face sharing e. This patch can then be merged to the face, so that no new face is introduced and the change to the topological structure is minimized. See Figure 4 and the following description:

1 Start with e_0. Find a patch of small size and incident to e_0. Denote the new patch by f'_0.

2 Let e_i be the current edge, v_i be the common vertex of e_{i-1} and e_i, and e'_i be the edge on f'_{i-1} incident to v_i.

3 Find a patch f'_i of small size and incident to e_i and e'_i.

4 If no such patches exist, go back to edge e_{i-1} and select a different patch as f'_{i-1}. It may be necessary to search further back.

5 Repeat Step 2 to 4 until the process reaches e_0 again.

6 Update the hole boundary $\{e_i\}$.

7 If $\{e_i\}$ is not empty, repeat Step 1-6.

Shells constructed individually may share vertices, edges and/or faces. Such features should be detected and eliminated. Similar actions to

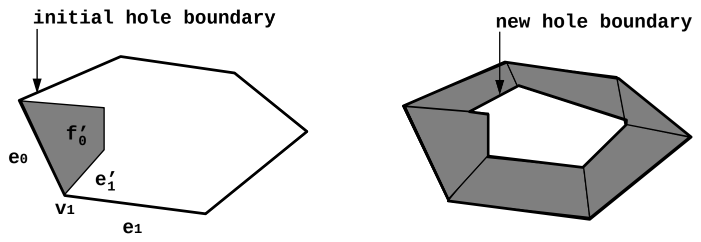

Figure 4. Hole filling in shell reconstruction

those of loop modification should be taken to achieve topological validity and retain design intent. For instance, if two shells share one common face and the face is in the original topological structure, then, it should be deleted and the two shells should be merged, because such a face is likely to be an internal wall left by improperly implemented regularized Boolean operations. However, if it is a face introduced to fill a hole, it should be trimmed away, together with all of its adjacent faces, and the induced hole must be closed.

4. Example

The model shown in Figure 5 is one part of a shaver handle, and created using a commercial CAD system. The size of the model is roughly $0.04m \times 0.06m \times 0.14m$. The underlying surfaces of the model consist of 16 integral and rational B-spline surfaces, 3 cylindrical surfaces and 5 planes. The global uncertainty measure is given by the designer as $10^{-6}m$. The model has $V = 40$ vertices, $E = 62$ edges, $F = 24$ faces, no inner loop, and one shell. The topological structure satisfies the sufficient conditions presented in Sakkalis et al [13]. Therefore, from the Euler-Poincaré formula $V - E + F = 2(1 - G)$, we can deduce that the model has genus $G = 0$ and thus it is homeomorphic to a ball.

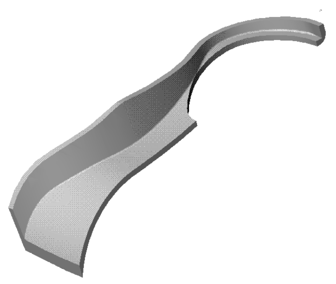

Figure 5. Part of a shaver handle

We convert this model into an interval solid model. Because edges in the original model are reasonably computed, initial conversion of each face is performed by growing the width of underlying curves of its edges by a given resolution. If such growth gives a valid face, the conversion of the face is finished, unless adjacency relations with neighboring faces are violated. For the latter cases, the reconstruction process using the proposed methodology is then performed.

The experiment starts with resolution $10^{-6}m$, given in the original STEP file. Eight faces have edges not on their underlying surfaces. Figure 6 shows one face with 2 edges partially on the underlying surface and one edge not overlapping the surface at all. Further computation reveals that the original face becomes valid at resolution $2 \times 10^{-4}m$. Figure 7 shows the valid face boundary. Overall, the model becomes valid at resolution $5 \times 10^{-4}m$.

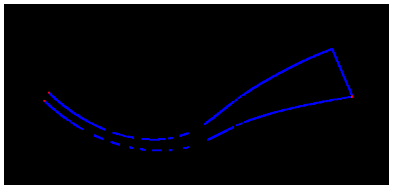
Figure 6. An invalid face with resolution $10^{-6}m$

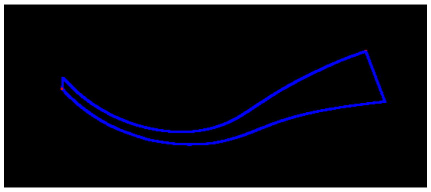
Figure 7. The same face as in Figure 6 becomes valid resolution $2 \times 10^{-4}m$.

We now test whether it is possible to reconstruct an interval model at the given resolution $10^{-6}m$, i.e. when all the surfaces in the model become interval surfaces with width $10^{-6}m$. For the face in Figure 6, because its underlying surface, now an interval surface, does not intersect one of the surfaces on which its adjacent faces are embedded, no valid face boundary can be constructed from surface intersections. If we further grow the width of all the surfaces to $10^{-5}m$, the face can then be reconstructed. The remaining seven invalid faces can be rectified at various resolutions, and an interval model can be constructed at resolution $5 \times 10^{-5}m$.

Acknowledgments

Funding for this work was obtained in part from NSF grant DMI-9215411, ONR grant N00014-96-1-0857 and from the Kawasaki chair endowment at MIT.

References

[1] J. H. Bohn and M. J. Wozny. A topology-based approach for shell-closure. In P. R. Wilson, M. J. Wozny, and M. J. Pratt, editors, *Geometric Modeling for Product Realization*, pages 297–318. Elsevier Science Publishers BV, 1993.

[2] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[3] P. M. do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[4] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[5] C. M. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer*, 22(3):31–41, March 1989.

[6] C.-Y. Hu, T. Maekawa, N. M. Patrikalakis, and X. Ye. Robust interval algorithm for surface intersections. *Computer Aided Design*, 29(9):617–627, September 1997.

[7] C.-Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part I, Representations. *Computer Aided Design*, 28(10):807–817, October 1996.

[8] C.-Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part II, Boundary evaluation. *Computer Aided Design*, 28(10):819–830, October 1996.

[9] I. Mäkelä and A. Dolenc. Some efficient procedures for correcting triangulated models. In *Proceedings of Solid Freeform Fabrication Symposium*, pages 126–134. University of Texas at Austin, 1993.

[10] M. Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, Maryland, 1988.

[11] T. J. Peters, N. F. Stewart, D. R. Ferguson, and P. S. Fussell. Algorithmic tolerances and semantics in data exchange. In *Computational Geometry '97*, Nice, France, 1997.

[12] A. A. G. Requicha. Representations of solid objects - theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464, December 1980.

[13] T. Sakkalis, G. Shen, and N. M. Patrikalakis. Representational validity of boundary representation models. *Computer Aided Design*, 32(12):719-726, October 2000.

[14] G. Shen, T. Sakkalis, and N. M. Patrikalakis. Interval Methods for B-Rep Model Verification and Rectification. In *Proceedings of the ASME 26th Design Automation Conference*, Baltimore, MD, September 2000. p. 140 and CDROM. NY: ASME, 2000.

[15] T. Sakkalis, G. Shen, and N. M. Patrikalakis. Topological and Geometric Properties of Interval Solid Models *Graphical Models*, 63(3):163-175, May 2001.

[16] G. Shen, T. Sakkalis, and N. M. Patrikalakis. Boundary Representation Model Rectification *Graphical Models*, 63(3):177-195, May 2001.

[17] S. Wolfe. Fixing bad CAD data. *Computer Aided Design Report*, 17(1):4–7, January 1997.