3-D Object Extraction Using Volume Computation

Varakorn Ungvichian
Department of Computer Engineering,
Chulalongkorn University
Bangkok, Thailand
ungvichian@thaimail.com

Pizzanu Kanongchaiyos
Department of Computer Engineering,
Chulalongkorn University
Bangkok, Thailand
pizzanu@cp.eng.chula.ac.th

ABSTRACT

This paper describes an alternative approach to extracting 3-D objects and volumes, from lists of given faces, edges, vertices, and the vertices' coordinates. Most graphics file formats store 3-D information for various purposes as a list of polygons, which does not provide a direct indication of structure or relationships between each object. This leads to the limitation of object identification within the list of data. The proposed algorithm was developed as part of a method for finding the Abstract Cellular Complex of an object. The volumes (whether closed or open) of an object are determined from the input set of faces. Each object is then extracted according to its manifold. This algorithm can identify every volume and extract them from the set of given data when the object(s) represented by the data have a genus of 0.

Keywords

Geometry, Modeling, 3-D Object Analysis, Surface Reconstruction

1. INTRODUCTION

There are many types of file formats being used to store 3-D graphics for multimedia purposes. Many of these store the information as a list of polygons. Storing the information in this way does not explicitly explain how many objects are in a given file, and does not provide a direct indication of structure or relationships between each object, which leads to the limitation of object identification within the list of data.

This algorithm was developed as part of research on converting a 3-D wireframe model into the Abstract Cellular Complex data structure described by Kovalevsky [Kov01]. Part of the conversion requires finding closed volumes from a set of given faces.

The algorithm analyzes its inputs, namely, a list of faces (and the vertices and edges that form each face), a list of edges, and a list of the vertices' coordinates, to produce grouped set(s) of faces, edges, and vertices which indicate each volume in the given input.

Currently, the algorithm is known to work on objects with genus 0 (i.e., with no holes). However, it can handle multiple objects in a single scene.

2. PREVIOUS WORK

In our research, we focus on finding individual closed volumes in an object represented by a given set of faces, which also involves determining the object's outside surface.

The problem described here is essentially a classic topological problem [Man88]. While simple traversal of the topology would be sufficient when the topology has already been determined, we have not completely organized the input topologically. For example, we have not radially sorted the faces incident around each edge. Because our input is not topologically complete, we have opted for an alternative approach utilizing geometric calculations.

One example of previous research on constructing solids from faces is Higashi et al.'s method for unified geometric modeling [Hig93], an extension of Mantyla's solid modeler [Man82]. It differs from our research in that Higashi's work utilizes a modified version of Mantyla's half-edge structure [Man88], while our research uses a simpler and more straightforward hierarchy: vertices, edges, and faces. Such a hierarchy is more intuitive to the way the average person views a solid, as consisting of these three kinds of elements.

Also, based on Baumgart's winged-edge structure [Bau75], we will have edge adjacent to exactly two faces in any given closed volume.

3. DEFINITIONS AND CONDITIONS

In this paper, an "object" refers to a set of faces, edges, and vertices *O* such that each face in *O* is adjacent to another face in the set and not adjacent to any face outside the set, while each edge and vertex in *O* is adjacent to a face in the set.

"Closed volume" refers to a set of faces, edges, and vertices *V* such that each edge is adjacent to exactly two faces; each vertex is adjacent to the same number of edges as faces; the numbers of faces, edges, and vertices satisfy Euler's polyhedral equation (in its most simplified form, $V-E+F=2$, where *V* is the number of vertices, *E* the number of edges, and *F* the number of faces); and the volume enclosed is not split up by any set of faces. "Open volume" refers to such a set *V* each edge in *V* is adjacent to *at most* two faces, other than closed volumes. Objects need not be comprised of just one volume. For example, two cubes with one shared face are considered as one object, while each cube is considered a separate closed volume. Meanwhile, a cube with one face missing would be considered an open volume. This paper concentrates more on obtaining the closed volumes.

The inputs for this algorithm are lists of faces, edges, and vertices, the relationships between faces and edges (i.e. which faces are adjacent to a given edge, and vice versa), and the relationships between edges and vertices (i.e. which edges are adjacent to a given vertex, and vice versa). The preconditions are that the object(s) represented by the input have a genus of 0.

The output from the algorithm are grouped set(s) of faces, edges, and vertices which describe each volume (closed or not).

4. ALGORITHM

The first step of the algorithm is to sort the vertices by their *x*, *y*, and *z* coordinates respectively (e.g., <0, 0, 0> comes before <0, 0, 1>, <0, 1, 0>, and <1, 0, 0>), and then transpose and/or reverse the order of the vertices in each face, so that the first vertex of the face is the one that is earliest in the list, and the second vertex is the earlier of the two vertices adjacent to that first vertex, with the new order still representing the face. For example, a face with vertices labeled 1-6-8-2 can be transposed to 6-8-2-1 and then reversed into 1-2-8-6. The transposed faces are then ordered by their first few vertices. For example, a face with vertices 1-3-4-5 comes before 1-4-5-6, but after 1-2-3-4.

The next step is tracing each object out, by starting at a random edge and adding it to a list, adding faces adjacent to that edge, then the edges in each adjacent face, and then faces adjacent to those edges, repeatedly until no more new faces or edges are added.

Each object may consist of several closed volumes. Therefore, closed volumes are traced in a different, and more complex, manner. First, the program finds the leftmost "available" face, with the following procedure:

The program looks for the vertex with the "smallest" coordinates (i.e., least *x*, then least *y*, then least *z*) in the current object (*A* in Figure 1). After finding the vertex, the program checks its adjacent vertices (*a*, *b*, *c*). The program calculates the cosine of the angle between a vector parallel to the *z*-axis and the vectors between the vertex with the smallest coordinates and each of its adjacent vertices, and picks the edge that is part of the object and has the highest absolute value of cosine (i.e., the smallest angle with the *z*-axis). After selecting the edge (***Aa***), the program calculates the vectors that are perpendicular with the edge and the normal vectors of its adjacent faces (***f***, ***g***). The program then picks the face where the resulting vector has the least angle with the *y*-axis (***f***).

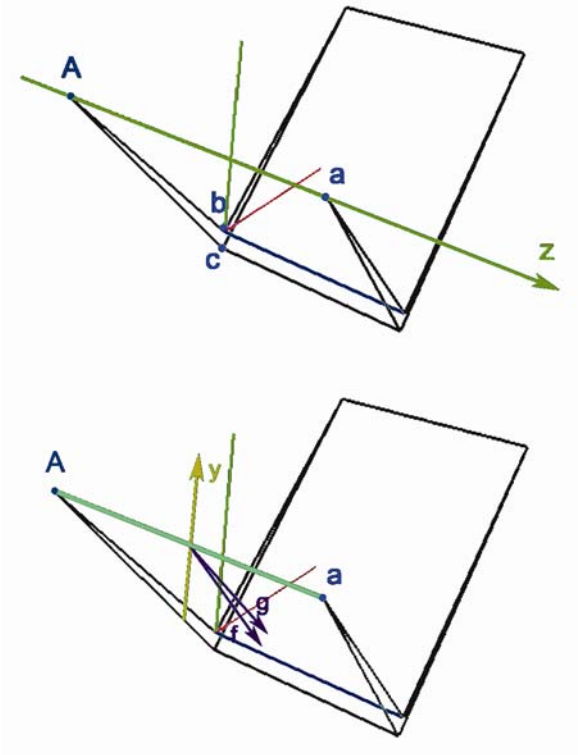

Figure 1. Finding the leftmost available face.

 ISBN 978-80-86943-02-2

After the leftmost "available" face is found, this face is put into a list of faces, with its edges and vertices also added to corresponding lists. For each edge adjacent to exactly two "available" faces, those two faces are added to the list of faces, with their edges and vertices added to their respective lists also. This repeats until no more faces are added. Figure 2 shows the results at the various stages of this portion of the algorithm on a 2-D mesh of triangles. Starting at the center of the mesh, one triangle and its edges are selected (in green). The triangles that are adjacent to the selected edges (in yellow) are next to be added, along with their edges (while edges with two selected faces are removed from the list. This repeats until no new faces and edges are added.

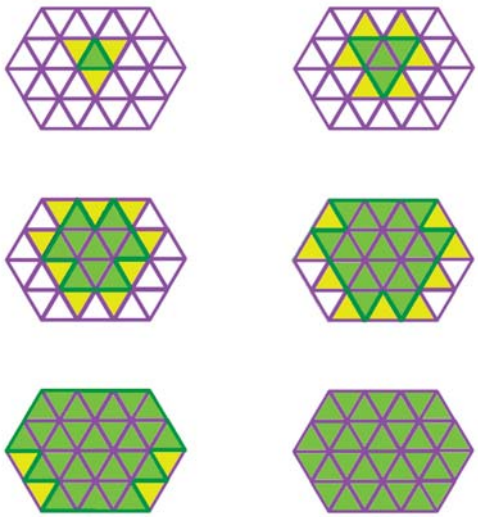

Figure 2. Tracing faces.

If the faces do not correspond to Euler's polyhedral equation, and there are faces remaining to be added, the program looks for edges which only have one selected face attached, and checks how many available faces are adjacent to each such edge (including the face already selected). If there is an edge adjacent to exactly two faces (i.e., there is one unselected face), it will simply add the unselected face, along with its edges and vertices. However, if all such edges are adjacent to at least three faces, the program needs to determine which face is part of the closed volume. To do this, the program calculates vectors which lie perpendicular to both the current edge being considered, and the normal of each of those faces. The program then traces edges with only one selected face attached, starting from the current edge and going in either direction, to find a series of such edges which form a continuous chain (see Figure 3, for an example), before finding the average coordinates of the vertices in said chain (the chain need not necessarily comprise a continuous loop as in Figure 3). This is to provide the program with a "general idea" of where the "inside" of the closed volume is.

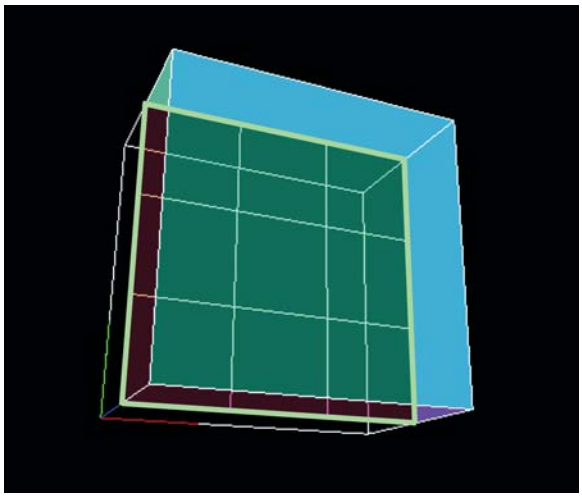

Figure 3. Edge chain (in light green).

The program calculates the vector between the centre of the current edge (***a***) and the average coordinates of the vertices (***b***), and, if necessary, modifies the vector to be perpendicular to the edge while lying on the same plane as the original vector ($(\boldsymbol{a} \times \boldsymbol{b}) \times \boldsymbol{a}$). This is used to determine the proper turn direction, and thus the face with the smallest dihedral angle in the proper direction, which is then added.

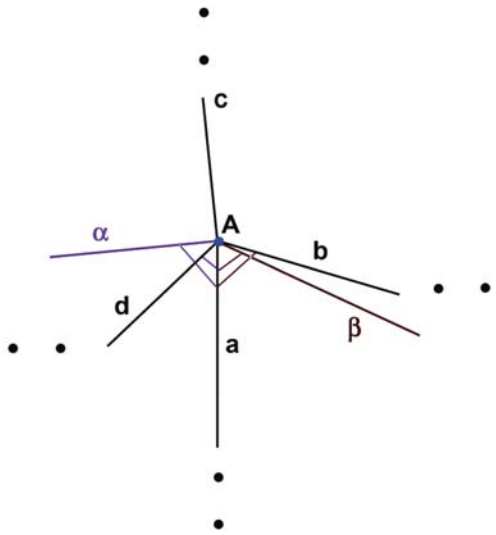

Figure 4. Face selection (explained below)

As an example, consider Figure 4. Here, 4 faces (represented by the edges marked *a*, *b*, *c*, and *d*) are adjacent to an edge (represented with vertex *A*). Face

a has already been selected. The program will either select face *b* or *d*, depending on the vector between the centre of edge *A* and the average of the vertices in a chain that starts with the same edge. Possible results are represented here with α and β. With α as the result, the program finds that the face with the smallest dihedral angle (in the direction of α) from α is *d*, and thus picks that face. However, with β as the result, the program will select *b* instead, since it is the face with the smallest dihedral angle in the same direction as β. Using the sample shape, the process is illustrated in Figure 5, using the same labels as Figure 4. Here, the turn direction is determined with the vector α, and the face with the least angle in that direction is *b* (ahead of *c*).

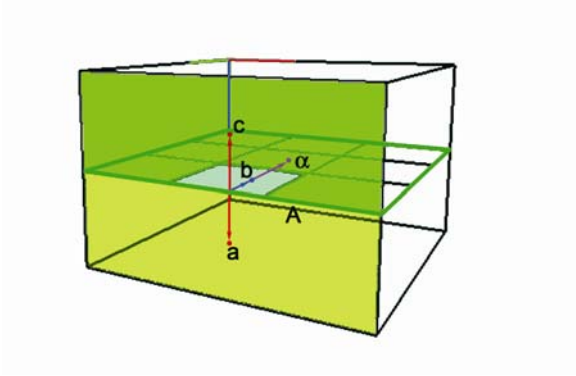

Figure 5. Face selection on sample shape.

This process is then repeated until the numbers of faces, vertices and edges correspond to Euler's polyhedral equation, thus comprising a closed volume, or until there are no more faces remaining in the object, in which case the selected faces represent an open (i.e., not closed) volume.

To determine which faces to remove from consideration, the outside surface of the whole object has to be traced. Tracing the outside surface uses the same algorithm as that used to find the closed volumes, except that when considering dihedral angles, the face with the *largest* dihedral angle is added instead (using Figure 4 as an example, α results in the program selecting *b*, and β results in the program selecting *d*). The list of faces in the closed volume is compared with the list of faces in on the surface. The faces in the closed volume that are not on the outside surface of the object are retained, along with the faces that have not been used so far.

A potential flaw in both the closed volume and surface finding algorithms is that the algorithm calls for calculating a vector that is perpendicular to the currently selected edge and is planar with the edge's adjacent face, as well as calculating another vector between the centre of the selected edge and the centre of the chain of edges starting from that edge. There is a possibility of those two vectors being in the same direction (resulting in a zero vector as their cross product), which would create difficulties in properly tracing surfaces and closed volumes, since the program cannot determine the proper turn direction in this case. Currently, the program solves this issue by testing different edges instead, and if all the edges produce this same result (which is most likely when there is just one selected face, or when the selected faces form a single plane), the program modifies the vector from the centre of the edge to centre of the edge chain, by adding to the *x* (and *y*, if necessary) values of the actual vector. The use of this special case solution is due to how both finding algorithms start at faces with the smallest coordinates. Figure 6 illustrates the special case solution, with a single-face edge chain: *a* marks the centre of one edge, while *b* marks the centre of the face / edge chain. Vector ***a*** is planar with the face and perpendicular to the edge, and vector ***b*** is the vector between the centre of the edge and the centre of the edge chain. Since vector ***a*** is parallel to vector ***b***, vector ***b*** is modified into vector ***c***, as if *c* were the centre of the edge chain.

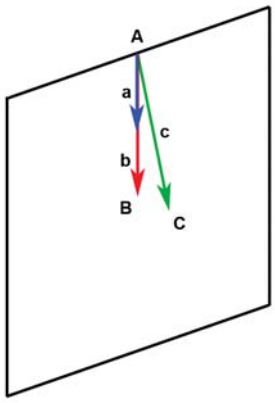

Figure 6. Special case solution.

The faces that are not in the split plane list (i.e., are on the object's outside surface) are removed from the list of faces in the object. The process then repeats until the face list is empty, and by this point, the program has identified all volumes (both closed and open) in the object.

5. RESULTS

The following figures show select results obtained using the algorithm.

Sample shape

Figure 7 shows the result obtained from the sample shape described in the previous section, correctly

identifying the nine faces in the middle as the split plane, and identifying the two boxes in the object (seen from different angles here for clarity).

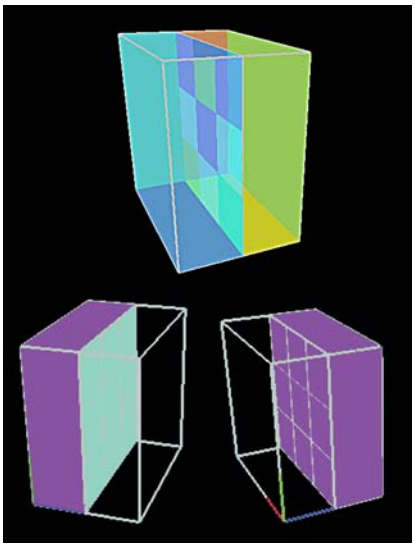

Figure 7. Sample shape results.

Triangle prism

The triangle prism is split into three parts by walls that connect to the centre of the prism. The algorithm identifies these three parts correctly, as seen in Figure 8.

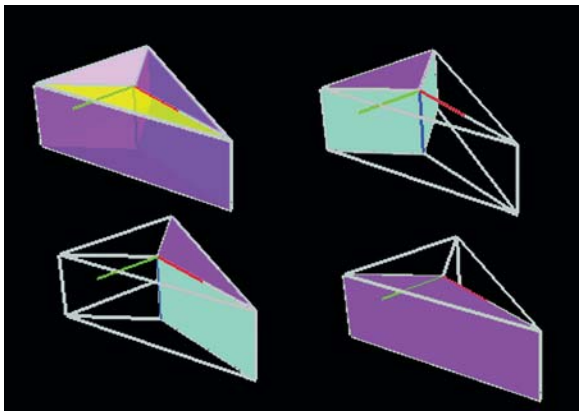

Figure 8. Triangle prism results.

Nine boxes

The algorithm correctly identifies the nine boxes that make up the larger box. The highlighted boxes in Figure 9 show the order in which each box is identified.

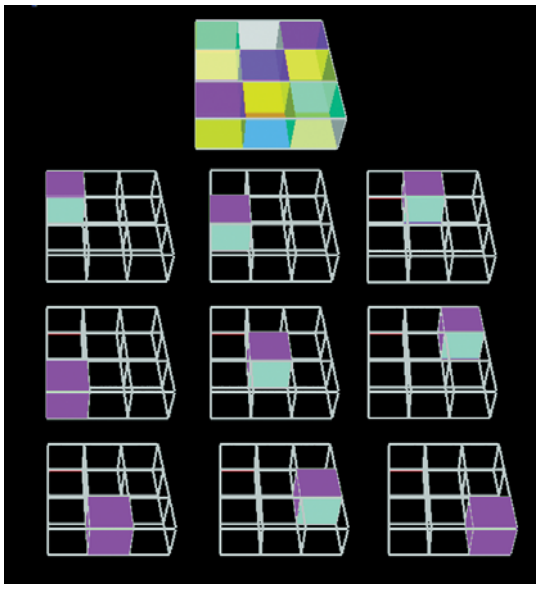

Figure 9. Nine boxes results.

Wedge box

This box with wedges attached to its sides shows an example of a non-contiguous non split plane (note the second shape in the first row of Figure 10). The algorithm handles this correctly to produce the five closed volumes that comprise the wedge box.

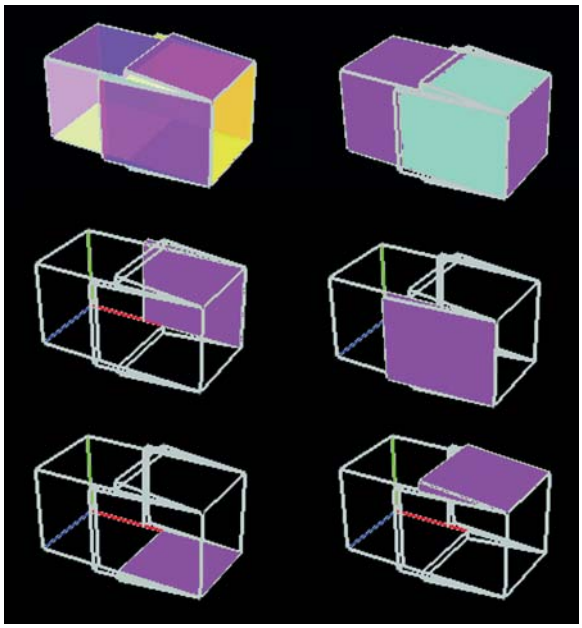

Figure 10. Wedge box results.

Fish

To illustrate the algorithm's use on a complex structure, we use the wireframe of a fish and its faces as an example. The algorithm identifies the main body of the fish as a single open volume. However, it

 ISBN 978-80-86943-02-2

also identifies a few "leftover" faces as distinct "volumes" from the main body, and none of the volumes are closed. The areas where the "leftover" volumes are identified are circled in Figure 11. This may be due to the shortcomings of the model itself, however.

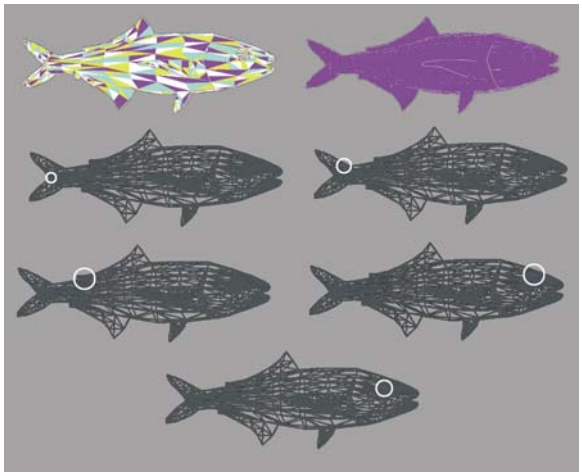

Figure 11. Fish results.

Results

The results of testing the algorithm using the sample shape in the main text, and then the three shapes described in the previous three subsections, are detailed in the following table.

Shape	**Sample shape**	**Triangle prism**	**Nine boxes**
Faces	19	12	42
Edges	40	16	64
Vols.	2	3	9
Vol. analysis time (s)	1.219	0.924	5.132
Total proc. time (s)	11.804	6.664	26.698
Shape	**Wedge box**	**Fish**	
Faces	26	2204	
Edges	44	3316	
Vols.	5	6	
Vol. analysis time (s)	2.253	49.826	
Total proc. time (s)	16.398	510.229	

Table 1. Results in numbers

6. SUMMARY

As currently implemented, the algorithm correctly identifies the volumes of each object. This algorithm has potential applications where counting 3-D objects in given wireframe data is necessary.

However, possible improvements to the algorithm include providing a less error-prone solution to the problem solved by the special case solution described in the main text, improving the algorithm to work with objects with genus 1+ (i.e., with holes), and possibly a more accurate (and foolproof) method for selecting the next face for each volume.

7. REFERENCES

[Bau75] Baumgart, B.G., Winged-edge polyhedron representation for computer vision, 1975.

[Hig93] Higashi, M., Yatomi, H., Mizutani, Y., and Murabata, S. Unified Geometric Modeling by Non-Manifold Shell Operation. SMA '93: Proceedings on the second ACM symposium on Solid modeling and applications, pp. 75-84, 1993.

[Man82] Mantyla, M. and Sulonen, R. GWB: A solid modeler with Euler operators. IEEE Computer Graphics and Applications, Vol. 2, No. 7, pp. 17-31, 1982.

[Man88] Mantyla, M., Introduction to solid modeling, 1988.

[Kov01] Kovalevsky, V., Algorithms and data structures for computer topology. Digital and image geometry: advanced lectures, pp. 38-58, 2001.