Chapter 2

Cutting - Topological singularity removal

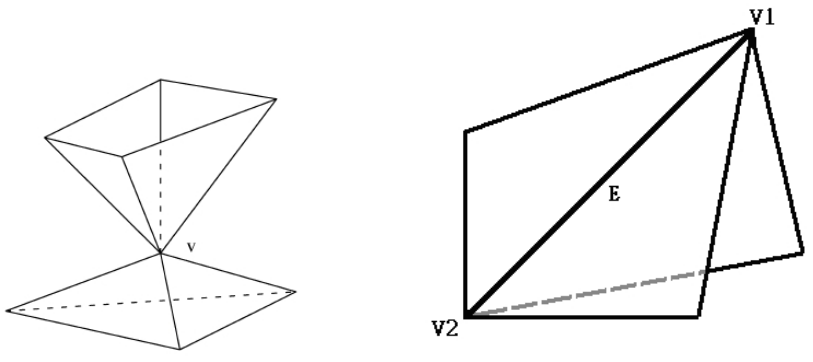
In real world, topological singularities existing in polygonal surfaces will degrade the performance of mesh processing algorithms. In recent years, a number of researchers have devoted themselves to tackle this problem [2]. Among the researchers who put their emphasis on this issue, André Guéziec et al. [4] proposed the 'Cutting and Stitching' method to remove topological singularities. They proposed two high-level operations - cutting and stitching, to execute the singularity removal task. In this study, we intend to apply their algorithm to convert an input polygonal surface to a manifold surface. Before discussing the cutting method, we will provide some basic definitions that will be used throughout this thesis.

After executing the cutting method proposed by André Gueziéc et al. [4], there are several follow-up steps to make the polygonal surface stitched. Because a polygonal surface after executing the above process may still contain many broken pieces, the user will consider it imperfect. However, the cutting method is only a preprocessing work of our filling holes algorithm, and the cutting process will help us achieve the goal of topological singularity removal, so we do not stitch the polygonal surface at this stage. In what follows, we shall define some basic terms.

2.1 Basic Notation

2.1.1 Polygonal Surfaces

Polygonal surface is a commonly used unit in the process of mesh representation. A polygonal surface is composed of a set of vertices v_i and a set of faces f_i. The vertices of a surface usually indicate the geometrical information of the surface and we can use these vertices to draw the corresponding polygonal surface in 3-D space. A face, on the other hand, is cyclically composed of at least three vertices v_i, v_j, v_k .... It is worthy of noticing that the constituent vertices of a face should be different. In other words, the constituent vertices of a face cannot form a self-loop. There are two possible orderings for the constituent vertices of a face, and this distinction will result in two different orientations, i.e.,



(a) Isolated Singular Vertex (b) Singular Vertex and Singular Edge

Figure 2.1: Topological singularity

clockwise and counterclockwise. As to the definition of an edge, it is connected by two vertices, which can be identified by an unordered pair (v_i, v_j) or (v_j, v_i).

From the viewpoint of topology, people concern more about the adjacency relationship. A face is said to be incident on its constituent vertices while an edge is said to be incident on its two endpoints, v_i and v_j . Two vertices can be called adjacent vertices if they share an edge. As to two edges sharing a vertex, we can call them adjacent edges. If two faces share a common edge, we call them adjacent faces.

For the set of faces that shares a vertex v, we call it star V, and it can be represented as v^*. The term valence represents the number of faces belongs to v^*. The link of *v*, l(v), is a graph whose nodes are the faces of v^*. An arc is created between two nodes if their two corresponding faces share an edge incident on the vertex v. A vertex is regular if its link is either a chain or a cycle; Otherwise, it is a singular vertex. We can also use the property to test an isolated singular vertex. A singular edge is an edge whose number of incident faces is greater than three. A singular vertex is also the endpoint of a singular edge. But it is not true for an isolated singular vertex.

For an edge that contains only one adjacent face, we can call it a boundary edge. The two endpoints of a boundary edge are called boundary vertices. A vertex which is neither singular nor boundary is called an interior vertex. In sum, any vertex can be categorized as one of the following three types: boundary vertex, regular vertex, and singular vertex. In addition, an edge can be classified into several types: regular edge, boundary edge, singular edge, and non-boundary edge. A regular edge is composed of two regular vertices. The boundary edge and singular edge are already defined before. Now we focus on the non-boundary edge. A non-boundary edge is not a boundary edge and is not singular edge, but its two endpoints are boundary vertices, singular vertices, or isolated singular vertices. The definition of a non-boundary edge is as follows:

$E_{nb}(v_1, v_2)$ = {e(v_1, v_2) | not (B(e) or S(e)) and ((B(v_1) or S(v_1) or IS(v_1)) and (B(v_2) or S(v_2) or IS(v_2)))}

B(*), S(*), and IS(*) are all Boolean operators. B(*) is true if the input is a boundary vertex or it is a boundary edge. S(*) is true if the input is a singular vertex or it is a singular edge. If the input vertex is an isolated singular vertex, IS(*) is true.

A connected component is a set of faces inside which two adjacent faces share an edge, and this edge is neither a singular edge nor a non-boundary edge. The modified definition will affect the result of the cutting algorithm. The full process proposed by ourselves includes cutting and filling. Therefore, we not only remove the topological singularities but also remove the effect that will cause potential topological singularities. A polygonal surface is classified as a manifold if every vertex belongs to it is a regular vertex; Otherwise it is non-manifold. Now, the goal is obvious:if we cut along the singular edges and the non-boundary edges in polygonal surfaces, these polygonal surfaces will become manifold. For continuing the subsequent processes, we have to make all polygonal surfaces become manifolds. In what follows, we shall proceed with describing the cutting algorithm.

2.1.2 The Cutting Algorithm

The input of our algorithm is a triangular surface. A triangular surface, also known as triangular mesh, is a special case of polygonal surface. Each face of a triangular surface is

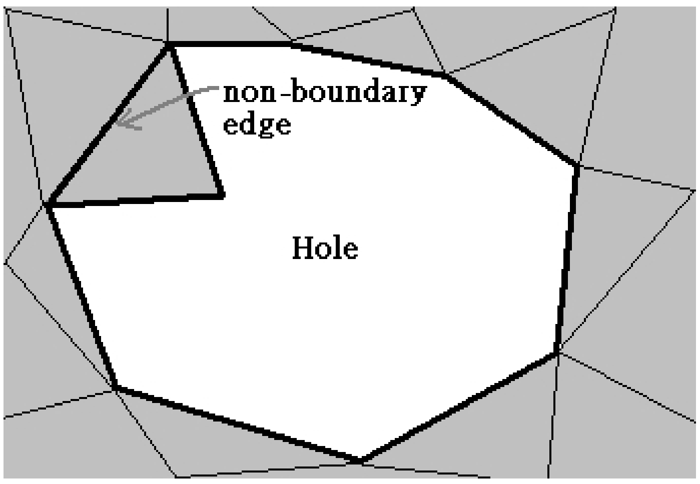


Figure 2.2: Non-boundary edge

formed by three vertices. These vertices can be used to do some basic operation such as calculating normal vector. For a face that contains more than three vertices, we have to do the following. If all the vertices of the face are on the same plane, then we just choose three to calculate the corresponding normal vector. Otherwise, we have to design some strategies to deal with the problem.

For avoiding the self-loop problem, our algorithm does not accept repeated vertices in a face as input. Therefore, we first check every face to see if it is a self-loop face before converting it from a triangular surface to a manifold triangular surface. Another important thing is that the input must be a triangular surface. Hence, when a polygonal surface comes in, we first convert the input to a triangular surface. There are quite a number of existing methods available for converting a surface from a polygon to a triangular one. Therefore, we shall simply choose one from these existing algorithms to fulfill our purpose. Next, we shall put our emphasis on the topological problem. In what follows is the detail of the algorithm.

Step 1 : Identify singular edges Count the number of adjacent faces of every edge. If the number is more than three, the edge is a singular one. We label all the singular edges. Furthermore, we also detect the edges near the boundary vertices and the singular vertices, and then label the non-boundary edges. These edges will be used to decide how many vertices should be duplicated when we apply the cutting method to a vertex.

Step 2 : Cut For each vertex, we apply the cutting method to break the triangular surface along the singular edges and non-boundary edges. We will get several connected components of manifold surface after executing the global cutting method.

2.2 Cut

The above mentioned cutting process breaks a surface along the singular edges, achieving the goal of topological singularity removal. After applying this process, the number of vertices will increase, but the number of faces will maintain the same. The difference between triangular surfaces before and after cutting will be discussed after showing the cutting method. In order to run this cutting method, we need a cutting operator that operates on a specific vertex.

2.2.1 Global Cutting

Global cutting, like its name, operates on all the vertices of a surface. This step is executed after identifying all singular edges. We do not have to locate where the singular vertices and the isolated singular vertices are because the cutting operator will process all vertices. In what follows, we shall elaborate the global cutting algorithm.

The global cutting algorithm:

Step 1: for each vertex we calculate its valence. We then derive the number of connected component using the reachable relation. If there is a singular edge between two faces, the two faces are said un-reachable. Under the constraint of reachable relation, we can partition v*, a set of faces that is incident to the vertex, into n connected components.

Step 2: duplicate n-1 copies of the vertex. As a result, n vertices and n connected components are obtained. The next step is to match them one by one. The matching problem is easy because there is no constraint, so we simply execute a matching algorithm. After we have matched these faces and vertices of the vertex that is under concern, the cutting on the vertex is completed.

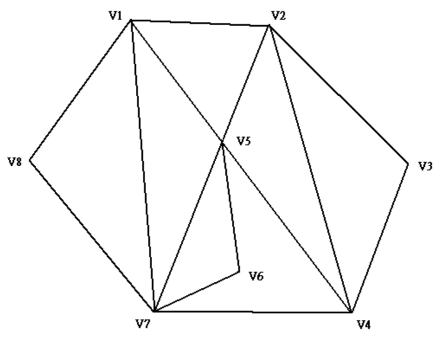
Step 3: repeat the same action for the remaining vertices until all vertices are processed.

After processing all the vertices, we know that there are many pieces of surfaces produced. How we choose the more useful parts for different applications is an important step in our experiment. It is possible to abandon the little broken pieces of the resulting mesh model. It is also possible to stitch some vertices which have similar geometrical properties such as geometrical coordinates. There are many follow-up processes that can be executed after our cutting procedure. However, here we shall focus on the cutting method, so the topic about how to make a good selection will be discussed in chapter 4. In chapter 4, we shall discuss more details about the strategies of our experiments.

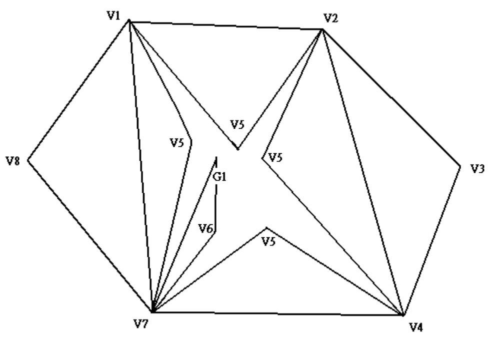
For clearly demonstrating how the global cutting algorithm works, we use the following example to explain. Fig. 2.3(a) is the input of triangular surface. When we process the vertex v5, the first thing is to count the number of the connected components whose faces are incident to v5. In this case, we can find that there are two connected components connected to v5. We duplicate one vertex of v5, and assign two vertices, v5 and G1, to the two connected components as Fig. 2.3(b) show. Fig. 2.3(b) shows after duplication there are in total 4 vertices of v5, but in fact there is only one vertex of v5. The reason is for convenience of telling. When we finish processing vertex v5, we continue to carry on the remaining vertices. Fig. 2.3(c) is the result after applying the cutting operator.

2.2.2 Discussion

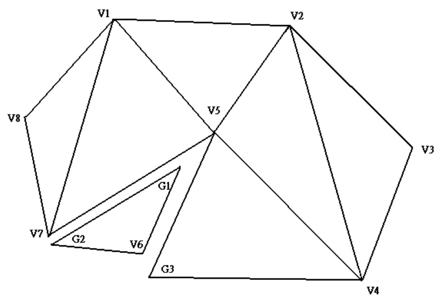
Our global cutting method is not exactly the same as that of André Guéziec's. We mix their global cutting and local cutting method. For each vertex, we adopt their local cutting



(a) Original mesh



(b) Cutting Operation on vertex V_5



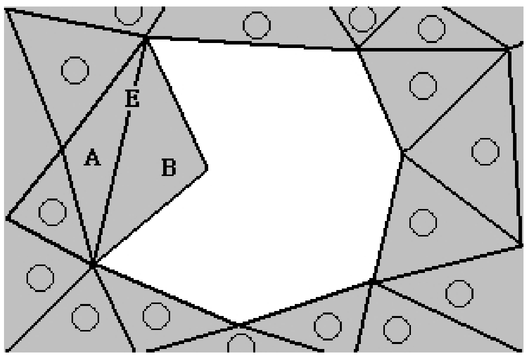
(c) Resultant mesh

Figure 2.3: Global Cutting

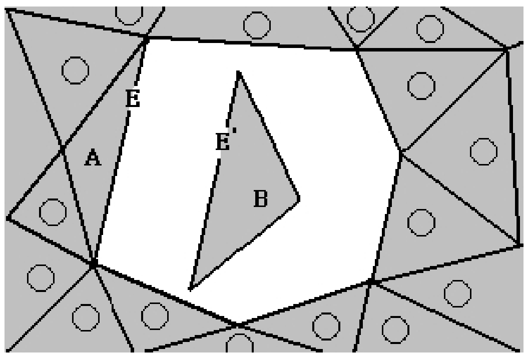
strategy to duplicate the vertex under processing, and match them to the neighboring faces respectively. In addition, we apply the cutting operator to all the vertices of a triangular surface.

There are two reasons for us to choose the combination of their global cutting and local cutting method. First, the local cutting operator is easy to implement. Tracing the surrounding faces and partitioning them using the reachable relation is not difficult to realize. Since our algorithm is fast, we can afford to process all the vertices of a surface. The advantage of doing do is that we do not need to label isolated singular vertices in advance. However, if a pre-processing step could lead us to know where the singular vertices and the isolated singular vertices are, the computation time can be even shorten.

Besides removing topological singularities, the cutting method also make the boundary



(a) Before cutting through the non-boundary edge E



(b) After cutting through the non-boundary edge E

Figure 2.4: Example on Non-boundary edge

face regular if there are holes in the mesh model. It means that we want the face adjacent to the boundary vertices to be adjacent to zero or only one boundary edges. Our definition also involves the singular vertices and isolated singular vertices because we know that these singular vertices would become boundary vertices after cutting process. Hence in order to take precautions for such situation, we make our definition of connected components more clearly.

When we apply the new definition of connected component, we will, of course, produce more split parts in mesh models. To overcome this problem, we use selecting operation that will be discussed in Chapter 4 instead of using stitching operation.

Take Figure 2.4 for example. Now our mesh model has a hole inside as Fig. 2.4(a) show. And we can find that the edge E is a non-boundary edge because it is not boundary but its two endpoints are both boundary vertices. The circles in the triangular faces mean that the face is fit for our definition of regular boundary face but faces A and B are not. Because faces A and B are adjacent to the non-boundary edge E, we label this edge, and cut it during the cutting process. In Fig. 2.4(b), the edge is already cut into two different edges, E and E'. Then the boundary faces adjacent to the hole now have only two kinds of faces. The first one is a face that has no nearby boundary edges. The other is a face that has only one adjacent boundary edge. And the cutting method removes the topological singularities finally.