Matchmaker: Manifold BReps for non-manifold r-sets

Jarek Rossignac and David Cardoze
Georgia Institute of Technology

Abstract

Many solid modeling construction techniques produce non-manifold r-sets (solids). With each non-manifold model N we can associate a family of manifold solid models that are infinitely close to N in the geometric sense. For polyhedral solids, each non-manifold edge of N with 2k incident faces will be replicated k times in any manifold model M of that family. Furthermore, some non-manifold vertices of N must also be replicated in M, possibly several times. M can be obtained by defining, in N, a single adjacent face TA(E,F) for each pair (E,F) that combines an edge E and an incident face F. The adjacency relation satisfies TA(E,TA(E,F))=F. The choice of the map A defines which vertices of N must be replicated in M and how many times. The resulting manifold representation of a non-manifold solid may be encoded using simpler and more compact data-structures, especially for triangulated model, and leads to simpler and more efficient algorithms, when it is used instead of a non-manifold representation for a variety of tasks, such as simplification, compression, interference detection or rendering. Most choices of the map A lead to invalid (self-intersecting) boundaries and to unnecessary vertex replications for M. We propose an efficient algorithm, called Matchmaker, which computes a map A, such that there exists an infinitely small perturbation of the vertices and edges of M that produces a valid (non self-intersecting) boundary of a manifold solid. Furthermore, our approach avoids most unnecessary vertex replications.

Introduction

We are concerned with boundary representations of polyhedral solids **[**Requicha80**]**. This domain is of considerable importance in many applications of Solid Modeling and of 3D graphics, because it lends itself to the most efficient representation of approximations of manufactured parts, human organs, and 3D shapes used in realistic animations and in interactive virtual environments.

Many data-structures have been proposed for representing the boundary of polyhedra. They contains three types of information

1. **Geometry:** The location of the vertices.
2. **Interpolation:** The connectivity graph, which defines the edges implicitly in terms of their vertices and defines the faces in terms of their bounding edges
3. **Ordering:** Ordering conventions or additional references. These capture face-face adjacency relations, the order of edges around faces and around vertices, and possibly the order of faces around edges [Rossignac94, Weiler86, Weiler87, Gursoz91]. Ordering often defines a consistent orientation for the faces, which makes it easy to compute the outward pointing normal to each face.

Note that ordering information is redundant, but important to accelerate the systematic traversal of the polyhedron's boundary by algorithms that detect collisions [Gottschalk96, Held95], that compute minimal distances, that perform Boolean operations [Requicha85, Mantyla88, Hoffman98] or morphing operations [Rossignac94b], that compress [Deering95, Taubin98, Rossignac98, Rossignac98b, Touma98, Gumhold98] or simplify [Heckbert97, Ronfard96, Hoppe96] the model or that convert it to triangle strips to render it efficiently [Evans96].

Also note that the topology, which combines interpolation and ordering information, is independent of the embedding (choice of coordinates for the vertices). However, many choices of geometry will produce invalid embeddings. We may define conventions for interpreting these invalid embeddings as the boundary of polyhedra. For example, we may use the parity rule, which states that a point P lies inside the polyhedron, if and only if a half-ray from P to infinity that does not intersect any vertex or edge of the polyhedron crosses an odd number of faces [Tilove80, Horn89]. However, the “natural” faces, edges, and vertices (defined precisely in the next section) of such a polyhedron do not necessarily have a one-to-one correspondence with the entities stored in the topology part of the representation or may not hold the relations defined in the topology [Eastman84]. For the purpose of this paper, we distinguish the following types of inconsistencies:

Osculating contacts: Two or more edges intersect at a point or a vertex is imbedded in the relative interior of a face or edge. However, a valid embedding (i.e.: and embedding that is consistent with the topology) could be produced by displacing some vertices by an infinitely small amount.

Coincident edges and vertices: Edges or vertices that are represented as different entities in the topology coincide in the embedding. However, a valid embedding (i.e.: and embedding that is consistent with the topology) could be produced by displacing some vertices by an infinitely small amount [Desaulniers92], or by bending some edges by an infinitely small amount.

Edge-crossings: Two or more edge represented in the topology as separate entities coincide and cannot be bent so that the ordering of faces around these edges would correspond to the information stored in the topology. In other words, the pointset defined by the topology is mapped into a pointset that crosses itself at these edges. These situations may confuse algorithms that rely on the fact that the orientation of the faces implied by the ordering information defines an outward pointing normal, i.e.: a normal that points towards the side of the face that is in contact with the exterior of the polyhedron.

Face-intersections: Two faces overlap or intersect in their interior:

4. Overlapping faces may no longer be part of the boundary of the resulting polyhedron and may lead to algorithmic errors, when for example computing minimal distances or inclusion relations between polyhedra.
5. Intersections of faces produce new edges and vertices in the embedding that are not reflected in the topology. The presence of these "unexpected" edges and vertices may cause algorithms that operate on the edges and vertices represented in the topology to produce incorrect results.

The data-structures for representing polyhedra [Weiler87] and the algorithms for processing them may be greatly simplified [Baumgart72], if the topological domain is restricted to manifold polyhedra. A polyhedron is manifold, if each edge is bounding exactly two faces and if each vertex has a single incident cycle of edges and faces (a more precise definition is provided in the next section). Yet, these simpler data-structures can rarely be used in practice, because the most popular construction operations for polyhedra may generate non-manifold models [Requicha85, Mantyla86, Hoffman89].

We describe here a process, which, given a non-manifold polyhedron, **N**, computes the topology, T, of a manifold polyhedron, **M**, and its geometry G, so that:

- each vertex of G corresponds to vertex of **N** (a many-to-one mapping)
- each face of T corresponds to a face of **N**
- the imbedding of **M** defined by G may have coincident edges and vertices, but has no osculating contacts, no edge-crossings, and no face-intersections inconsistencies (as defined above).

Representing **N** in terms of T and G, rather than in terms of its natural topology and geometry, permits to use safely most data-structures and algorithms designed for manifold polyhedra [Desaulniers92].

Our method, called Matchmaker, computes a topology, T, which considerably reduces the number of coincident vertices in G. In many cases, G is identical to the vertex list of **N**. For example, the topology produced by Matchmaker for a solid resulting from the union of two cubes that share a common edge (Fig. 1, left) would correspond to a deformed manifold model (Fig. 1 right) with no vertex replication. Note that the number of edge-entities in T is defined by **N**. Avoiding unnecessary vertex replication in T and G helps reduce the storage and processing costs of the representation.

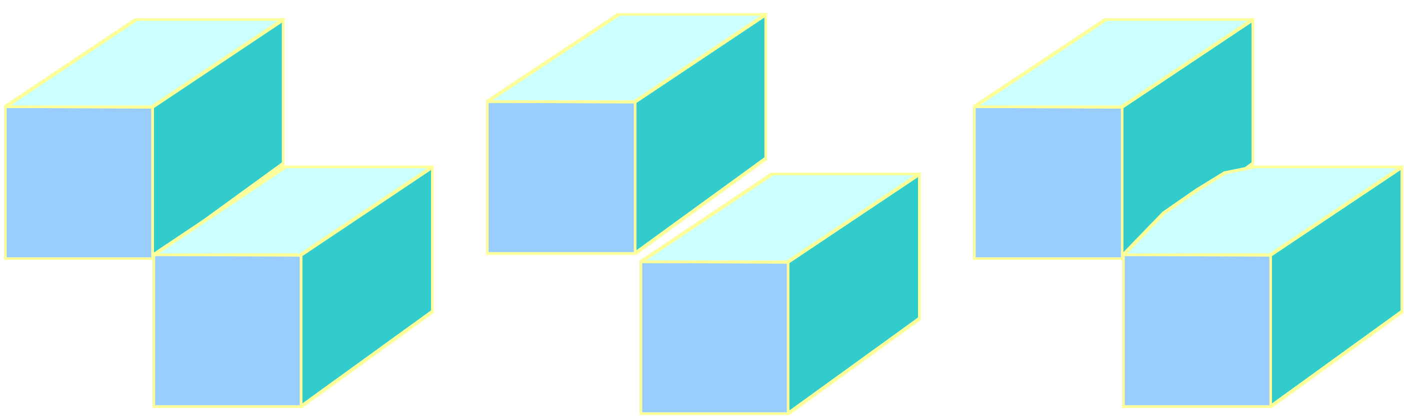

Figure 1: *The non-manifold polyhedron (left) has one non-manifold edge. It may be represented in several ways by a manifold topology with an inconsistent embedding that has coincident edges and vertices. One such representation (center) has two pairs of coincident vertices. Its topology is shown by slightly separating the coincident entities, so as to form a valid imbedding. The representation produced by Matchmaker does not have any coincident vertices and thus has the same vertex entities as the original model. Its structure is shown (right) by bending the two coincident edge-entities that are obtained by replicating the non-manifold edge of the original object.*

The paper is organized as follows. After defining our terminology and introducing some basic properties, we discuss prior art, describe the Matchmaker process of assigning face-face adjacency, and then suggest how a pseudo-manifold boundary representation may be derived from the resulting adjacency table.

For simplicity, we assume that the faces of the polyhedron are triangulated. Although, with some restrictions, our approach may be applied to more general polygons and even to curved faces, the domain of triangle meshes is important in its own right. Indeed, manifold triangle meshes are popular in many 3D settings, because they may be represented compactly [Rossignac98, Taubin98, Touma98, Deering95, Gumhold98], downloaded progressively [Hoppe97, Taubin98c], and rendered efficiently using popular graphics libraries [Rockwood98, Chow97] that interface to hardware rasterizers optimized for triangles.

BACKGROUND

We develop in this section the definitions, concepts, and data structures used throughout this paper.

Polyhedra, faces, edges, and vertices

A three-dimensional pointset, **N**, is a polyhedron, if it satisfies the following properties:

- **N** is finite (i.e.: bounded)
- **N** is equal to the closure of its interior [Requicha80]
- The boundary of **N** is contained in a finite union of planes

Zero-dimensional geometric singularities of the boundary of **N** (i.e., isolated points, where the normal is not defined) are its vertices. Their set is denoted V_N. The connected components of the one-dimensional geometric singularities (i.e., curves of points where the normal is not defined) minus V_N are the edges of **N**. Their set is denoted E_N. The faces of **N**, whose set is denoted F_N, are the connected components of the difference between the boundary of **N** and the union of its vertices and edges. Note that, by our definition, edges and faces are relatively open, i.e.: they do not contain their boundary [Alexandroff61, Massey67]. Furthermore, they are pair-wise disjoint.

An edge **E** is said to be a manifold edge of **N**, if and only if, **E** is bounding exactly two faces of **N**. A vertex **V** of **N** is said to be a manifold vertex of **N** if and only if **V**.star, the star of **V** (i.e., the union of all the edges and faces incident upon **V**), does not contain any non-manifold edges and if the pointset **V**.star–**V** is connected. A polyhedron is manifold, if and only if all its edges and vertices are manifold.

Manifold and non-manifold data-structures

Various data-structures has been proposed for representing the faces, edges, and vertices of such polyhedra (see [Who85, Rossignac94] for surveys). These data-structures typically represent edges indirectly through references to their bounding vertices. Similarly, they represent faces indirectly through references to their bounding edges. Listing these references without any order suffices to unambiguously define the edges, faces, and the interior of **N** [Rossigna89]. Nevertheless, most authors and developers organize these references so as to capture the order of edges around the loops that bound each face and the adjacency between pairs of faces with one or several common edges [Rossignac94].

Several data structures have been proposed for representing manifold polyhedra [Baumgart72, Weiler85, Kaley89]. They encode adjacency and ordering relations between half-edges (sometimes called darts). A half-edge h is the association of a face **F** with an edge that bounds **F** on its right (the term right is defined with respect to a coordinate system where the tangent to the edge is the forward direction and where the outward pointing normal to **F** is the up-vector). Most of these data-structures store for each half-edge h:

- the references to the half-edges, h.n and h.p, that immediately follow and precedes h in the loop that bounds **F**
- the reference to the opposite half-edge, h.o, which spans the same line segment, but has opposite orientation
- the references to the starting vertex, h.s, and optionally to the end vertex, h.e, of h. (Note that h.e=h.o.s.)

Note that, although the boundary of a face **F** of a manifold polyhedron needs not be manifold, its loops are well-defined sequences of half-edges. Basically, each loop of **F** is a maximally connected component of the relative boundary of **F**. Each loop may contain vertices with more than two incident edges, even though these vertices are manifold vertices with respect to the polyhedron. The order of the half-edges around the loop is defined as follows. Half-edge h.n follows h in the loop if the starting vertex, h.n.s, of h.n is the ending-vertex h.e of h and if there is a curve-segment that connects h and h.e and is contained in the intersection of the interior of **F** with the difference between an infinitely small open disk centered at h.e and the point h.e. (Basically, h' is the right-most turn amongst the half-edges of **F** when coming from h into h.e.) A non-manifold face and its loops are shown Fig. 2.

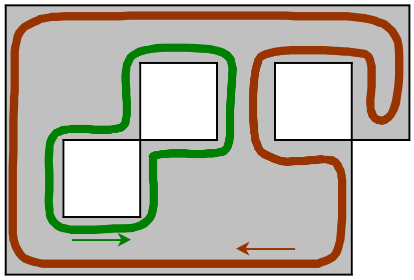

Figure 2: *This non-manifold face, which, according to our definition, is as maximally connected two-dimensional component of the interior of the intersection of the polyhedron's boundary with a plane, has two loops, marked with thicker lines. The loops are traced by taking the right-most turn at each vertex (thick curves show the closed loops). This traversal orients the loops (arrows) so that they have the interior of the face on their right.*

If the representation, v, of each vertex, **V,** is associated with one of the half-edges v.h that has it as a ending-vertex (h.e==v), one can access at least one half-edge h in each loop that contains v by setting h=v.h and by iterating over the sequence h=h.n.o until h becomes v.h again. Each loop may be followed by iterating the assignment h=h.n. Because faces may be bounded by more than one loop, each face usually points to a list of half-edges, one per loop, starting from a half-edge of the external loop. This situation is illustrated Fig. 3.

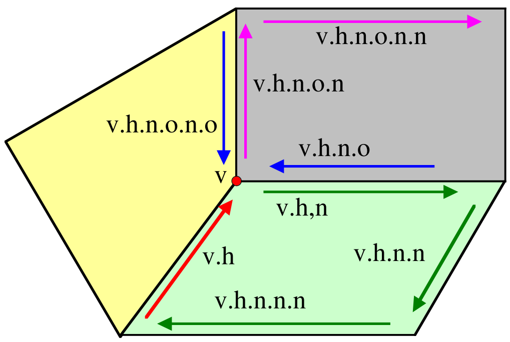

Figure 3: *The representation, v, of a vertex,* ***V****, points to v.h, a half-edge that ends at v. From v.h, we can follow the ".n.o" links (blue arrows) to access one half-edge in each loop that contains v. We can traverse the corresponding loops by following the ".n" links (green arrows).*

Although similar datastructures have been proposed for non-manifold polyhedra, they are less regular and more verbose and they require more complex algorithms to process the boundary representation. For instance, in a non-manifold polyhedron, a non-manifold edge will correspond to an even, but larger than two, number of half-edges. Consequently, a half-edge, h, does not longer have a unique opposite half-edge, h.o. Furthermore, two half-edges with opposite orientation may be associated with the same face (See [Rossignac94] for examples and further discussions). Similarly, a non-manifold vertex v does not have a unique incident cone of edges and faces. Therefore one must store with each non-manifold vertex, v, a list of out-going half-edges, v.H, such that each half-edge h of v.H bounds a different cone [Mantyla86, Gursoz91].

PRIOR ART

Pseudo-manifold representations of polyhedra have been used since the early days of solid modeling [Requicha82, Requicha83], prior to the invention of data-structures capable of representing non-manifold models [Requicha92, Rossignac97] These contained osculating contacts and coincident edges and vertices, and were produced by Boolean or extrusion operations. They could not, in general, be used as arguments to such operations. The use of pseudo-manifolds to represent non-manifold r-sets was formalized by Desaulniers and Stewart [Desaulniers92], who show how to extend the usual manifold data structures and the associated Euler operators to the class of r-sets.

The work reported here is related to techniques for restoring a topological adjacency graph from an incomplete representation or inconsistent representation of a triangle mesh [Murali97, Gueziec98] or from a list of triangles whose vertex coordinates may contain numerical errors [Segal90]. Techniques for avoiding the production of inconsistent geometric data are discussed in [Sugihara89, Banerjee96].

Recently, Gueziec et al. [Gueziec98] have presented several approaches for converting collections of polyhedra into piecewise manifold polyhedral surface models. They do not restrict their domain to the boundaries of polyhedral solids and do not attempt to avoid edge-crossings. Instead, they focus on avoiding the creations of pairs of edges in the manifold model that share the same two vertices. One of their approaches cuts the surface at each non-manifold edge. These cuts contain replications of the non-manifold edges and vertices and are now part of the boundary of the surface. The result is a fragmented surface that is a topological manifold, but whose imbedding may have osculating contacts, coincident edges and vertices in its boundary, or face intersections, but no edge-crossing. Then, to reduce the number of bounding edges and the number of vertex replications, they stitch the surface by pairing instances of the same edge using a greedy approach, while avoiding the creation of non-manifold situations and the creation of pairs of manifold edges that share the same vertices at both ends. They propose a variation, which only stitches faces that are part of the same connected component of the model that results from the initial cuts. For the two cubes that share an edge, this strategy would produce the central solution shown in Fig. 1, which replicates two vertices, instead of the solution on the right produced by Matchmaker.

INPUT AND OUTPUT

The Matchmaker technique described here takes as input a vertex table V and a triangle-vertex incidence table TV. Each triangle, **T**, is associated with an index, *t*. The entries TV[t,1], TV[t,2], and TV[t.,3] contain integer indices, which identify three vertices in V. These vertices bound **T**. We assume that there exists a choice of vertex locations for V, such that the union of all the triangles in TV forms the boundary of a polyhedral r-set [Requicha80].

Matchmaker first computes a triangle-adjacency table, TA, which for each edge **E**, matches all the triangles incident upon **E** into pairs. For each triangle index *t*, TA contains the triangle-indices of three adjacent triangles:

- TA[*t*,1], which is the index of a triangle that is incident upon vertices TV[*t*,2] and TV[*t*,3]
- TA[*t*,2], which is the index of a triangle that is incident upon vertices TV[*t*,3] and TV[*t*,1]
- TA[*t*,3], which is the index of a triangle that is incident upon vertices TV[*t*,1] and TV[*t*,2]

This notation is illustrated Fig. 4.

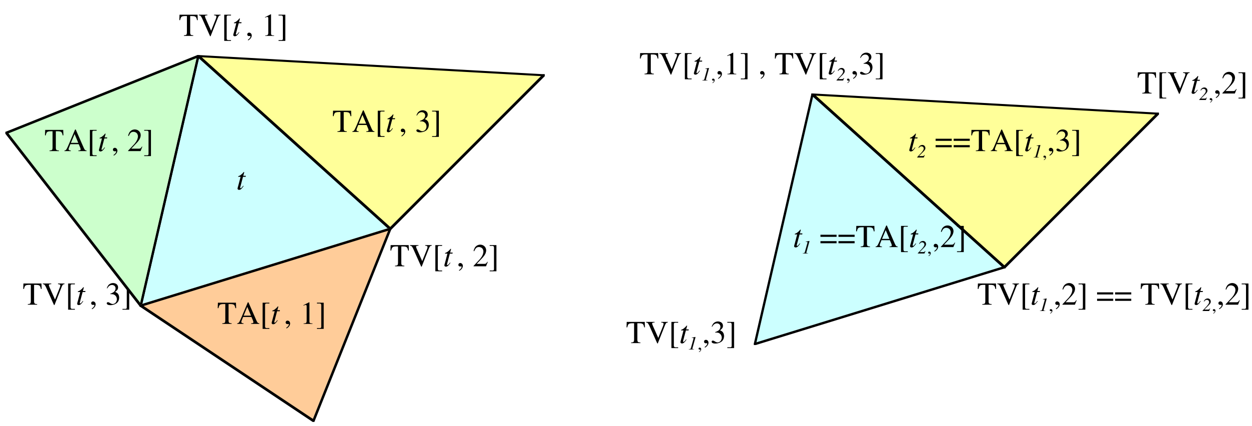

Figure 4: *Given a triangle index, t, the triangle-adjacency entries are sorted (left) so the TA[t,i] points to an adjacent triangle incident upon the edge opposite to the vertex TV[t,i]. The adjacency matches are reciprocal relations (right).*

Each adjacency relation is its own inverse: If t'==TA[t,i], then t==TA[t',j] for some j. Each pair of such adjacencies corresponds to an edge of the final manifold representation of the original polyhedron. If these edges were bent by an infinitely small amount in the appropriate direction, the resulting shape would either be manifold or would have only isolated non-manifold vertices. We say that the resulting model is edge-manifold. This term distinguishes topological representations with non-manifold vertices but no non-manifold edges from non-manifold topologies and from manifold topologies. The edge-manifold representation is produced by splitting each non-manifold edge with 2k incident faces into k manifold edges.

To obtain a manifold topology from an edge-manifold one, each non-manifold vertex, **V**, must be replicated (Fig. 5) into as many instances as there are cycles in its crown. The crown of a vertex **V** is the relative boundary of the star of **V**. The star is defined as the union of the edges and triangles incident upon **V**. To cluster the triangles incident upon a single non-manifold vertex into the different cones (one per cycle in the crown), we walk the circuits of its crown as explained below.

The replicated copies may be appended at the end of the V table. The paper focuses on the computation of an adjacency map TA that does not produce any edge-crossings and avoids unnecessary vertex replications. Each triangle that was incident upon **V** is now incident upon one of its instances. Since only one of the instances may be associated with the same index as was originally used for **V**, the description in TV of the triangles that are now incident to the other copies of **V** must be updated.

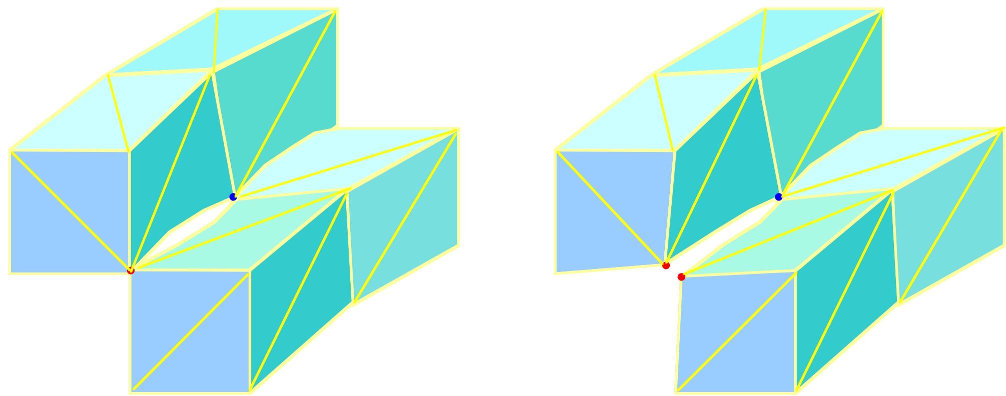

Figure 5: *The original model had two non-manifold edges and three non-manifold vertices. The adjacency assignment performed by Matchmaker is illustrated (left) by bending edges, so as to show the pairing of adjacent triangles. The blue vertex is a manifold vertex, because it has a single cycle of incident faces and edges. The red vertex (in front) is a non-manifold vertex. It will be replicated by Matchmaker to produce a manifold topology, illustrated on the right.*

The result of this process is the extended vertex table, V, the updated triangle-vertex incidence table, TV, and the triangle-triangle adjacency table, TA. Together, they define a manifold topology and an embedding which may have coincident edges and vertices, but is free from edge-crossings, osculating contacts, and face-intersections.

More complete boundary graphs, such as the half-edge structure described above, may be easily derived from these tables. Each triangle defines three half-edges. For each half-edge, h, of triangle t, the table TV defines h.p. h.n, h.s, and h.e.

OPTIMAL ADJACENCY MAP FOR A SINGLE NON-MANIFOLD VERTEX

In this subsection, we explain how to turn a single non-manifold vertex into one or several manifold vertices by assigning adjacency maps to all of its incident edges. Furthermore, the procedure proposed below produces the minimum number of vertex replications.

Vertex-neighborhood, regions, and loops

Consider a non-manifold vertex, **V**, of a polyhedron, **N**. Let **C**(**V**,**N**) be the *vertex-neighborhood* [Requicha82, Mantyla86, Gursoz91] of **V**, defined as the intersection of the interior of **N** with an infinitely small spherical surface centered at **V**. **C**(**V**,**N**) is a two-dimensional subset of a spherical surface. It is equal to the closure of its interior relative to the sphere. The boundary of **C**(**V**,**N**) is contained in a finite union of circles and is not manifold if **V** bounds a non-manifold edge.

The vertices of **C**(**V**,**N**), which we call *points* to distinguish them from the vertices of **N**, correspond to edges of **N**. Non-manifold points of **C**(**V**,**N**), i.e., points where more than two arcs meet, correspond to non-manifold edges of **N**. Each arc corresponds to a face **F** of **N** and is oriented so that the outward pointing normal of **F** points to the left of the arc. We define *left* by considering the arc orientation to be the forward direction and the outward normal to the sphere to be the up-vector.

The maximally connected components of the relative interior of **C**(**V**,**N**) are called *regions*. The boundary of each region is composed of one or more loops. A *loop* is a maximally connected component of the boundary of a single region. Each loop may be represented as a circular list of oriented arcs. The arcs are oriented (as explained above) so as to have the region on their right. Each arc shares an end-point with the next one in the loop. We compute the proper order of arcs along the loop by taking the right-most arc at each end-point when tracing the loop. Note that each arc belongs to a single loop and that, by definition, two loops that bound the same region are disjoint.

When **V** is a manifold vertex, **C**(**V**,**N**) has a single loop and each point of that loop is a manifold point. When **V** is an isolated non-manifold vertex that has zero incident non-manifold edges, **C**(**V**,**N**) has several disjoint bounding loops, each one of which is a one-manifold curve bounding a different region. Finally, when **V** is bounding one or more non-manifold edges, the loops may be non-manifold (i.e., may visit the same vertex more than once) and two loops that bound separate regions of **C**(**V**,**N**) may intersect at one or more points. We will use the term *self-contact* for non-manifold points that are bounding a single region. The other non-manifold points will be called *contacts*.

Manifold circuits

Each maximally connected component of the union of all the loops of **C**(**V**,**N**) is a *circuit*. We describe in this subsection a novel approach for arranging the arcs of a circuit so that they form a cycle. The end-point of each arc is identical to the starting-point of the next arc in the cycle. (The starting and end points of each arc are identified using the arc's orientation defined above.) The cycle splits each crossing or self-crossing with 2k incident arcs into k manifold points, each one being the common boundary of two consecutive arcs in the cycle. This split defines the adjacency map for the corresponding triangles: if two arcs are consecutive in the cycle, the corresponding triangles are adjacent. There may be more than one way to define a cycle for each circuit.

To construct the cycle for each circuit, we first construct the loops (as explained above) and mark each edge with the ID of its loop. We initially assign the adjacency of the non-manifold edges incident upon **V** so as to reflect the adjacency of the corresponding arcs in the loops. Then, we proceed to merge loops, one at a time, by changing that adjacency. To better understand the nature of our cycle construction, consider a graph whose nodes are loops and whose links correspond to crossings and connect pairs of loops incident upon the crossing. We want to select a subset of these links that defines a spanning tree of that graph, while avoiding edge-crossings. Each link of that spanning tree corresponds to a crossing where two loops are merged into one.

To achieve this objective, we mark the visited arcs and maintain a table, which for each loop indicates whether it has been already merged or not. We pick an edge and start walking along its loop. When we reach a manifold point or a self-crossing, we proceed to the next edge in the loop (thus turning right). When we reach a crossing, **c**, arriving by arc **A**, we are only allowed to depart by an arc, **D**, such that:

1. **D** has **c** as its starting vertex,
2. **D** has not been previously visited,
3. **D** is not separated from **A** by two previously visited adjacent arcs, so that one of them would be after **A** and before **D** and the other one after **D** and before **A** in the circular ordering of arcs around **C**.

We follow these rules:

- If there are no arcs that satisfy these three conditions, we have completed the circuit and can chain the last arc with the first one.
- If there are arcs **D** that satisfy these three conditions and belong to loops that have not yet been merged, we proceed to the left-most of these arcs (the first one reached by turning clockwise around **c**). We mark it as visited and mark its loop as merged.
- If there are arcs **D** that satisfy these three conditions but all belong to previously merged loops, we proceed to the right-most of these arcs (i.e.: the first one reached by turning counter-clockwise around **c**). We mark it as visited and mark its loop as merged.

This process is illustrated Fig. 6. Throughout this paper, we will draw **C**(**V**,**N**) as a topologically equivalent polygonal region. Each vertex in the drawing is a point of the neighborhood of **V** and represents an edge incident upon **V**. Each line in the drawing represents an arc and is associated with a different triangle.

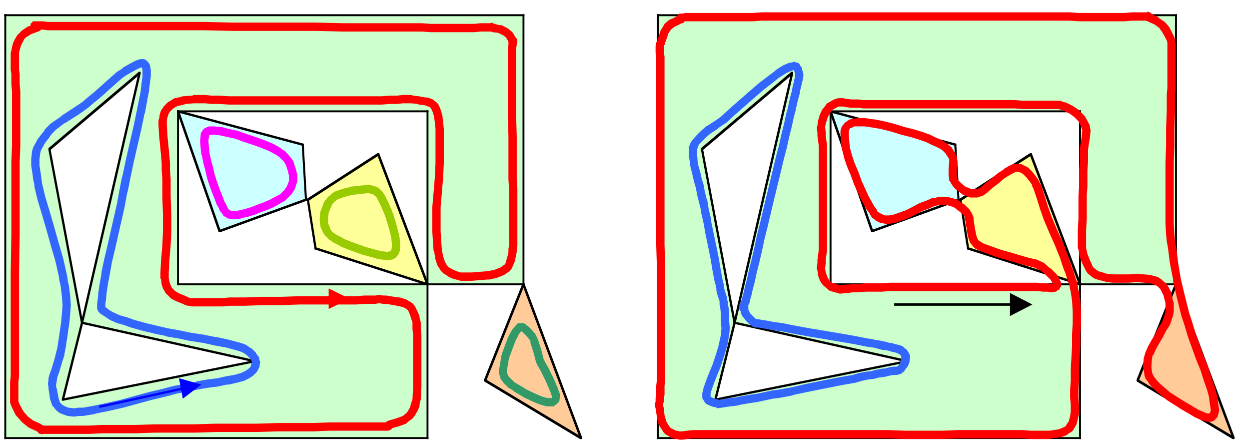

Figure 6: *The vertex neighborhood of a non-manifold vertex,* ***V****, of a polyhedron,* ***N****, is a two-dimensional region,* ***C(V,N)****, that is the intersection of the interior of* ***N*** *with a small sphere around* ***V****.* ***C(V,N)*** *may be decomposed into its maximally connected components, called regions. These are shaded using different colors (left). The loops of these regions are the connected components of their boundary. They are shown left using thicker lines of different colors. In our example, the union of these loops has two connected components (right). These are the circuits of* ***C(V,N)****. The cyclic order of the arcs along the circuit (see fat curves on the right) is produced by picking any arc and by marching along the circuit in a manner consistent with the arcs orientation and without ever crossing the path followed so far. When, at a crossing, we can access arcs that belong to loops not previously reached, we take the left-most turn. Otherwise, we take the right-most turn. The cyclic order of arcs produced by this process may depend on the starting arc (shown here using a black arrow).*

Note that to identify the left-most and the right-most arcs, we used an ordering of arcs around each crossing, which is equivalent to an ordering of the corresponding triangles around the corresponding non-manifold edge. The ordering may be easily computed from the location of the vertices of the triangles, either directly in 3D or by using their projection onto a plane that is orthogonal to the non-manifold edge.

Adjacency assignments for the edges of a non-manifold vertex

Given its cyclic order, each circuit may be represented as a curve with a manifold topology and a non-manifold embedding. Each circuit corresponds to a cone of **V** in the edge-manifold topology. A cone is a cycle of edges and triangles that are incident upon **V**. Note that **V** is the only intersection of its cones.

When the procedure outlined above and illustrated in Fig. 6 for computing manifold circuits is applied to the neighborhood of a vertex **V**, it groups the arcs that are incident upon any given crossings or self-crossings point into pairs that are consecutive in the circuit. Note that this grouping corresponds to an adjacency match for the corresponding triangles. Indeed, the two corresponding triangles share an edge and will be consecutive in a cone of **V**. Therefore, we have defined a procedure for computing optimal adjacency assignments for all the edges incident upon a single vertex. These assignments do not produce any edge-crossings and they minimize the number of cones, and thus the number of times **V** must be replicated.

Locally optimal adjacency assignments may not lead to globally optimal solutions

Notice also that we can repeat the above process for another vertex, as long as that vertex does not share any incident edge with the first one, and so on. This naïve approach will produce adjacency assignments for some of the non-manifold edges. These assignments split the corresponding crossings and self-crossings in the neighborhoods of the remaining non-manifold vertices. These splits may be incompatible with the best adjacency assignment for these remaining vertices. Indeed, a non-manifold edge, **E**, is represented as a crossing or self-crossing in the neighborhoods of its two bounding vertices. Each triangle incident upon **E** is presented by one arc in each of these neighborhoods. If two of these arcs are consecutive in a circuit in one of the neighborhoods, they must also be consecutive in a circuit in the other neighborhood. Building manifold cycles that agree with the previous splits may result in vertex replications that could be avoided by a better choice of all adjacency maps.

MATCHMAKER GLOBAL STRATEGY

The Matchmaker process described in this paper focuses on assigning face-face adjacencies while minimizing the total number of circuits. A locally optimal adjacency assignments for one non-manifold vertex, **V**, may result in the creation of several additional circuits for other vertices that bound edges incident upon **V**. To address this problem, Matchmaker assigning adjacency maps to the non-manifold edges in a specific order. The Matchmaker process is structured in three phases detailed in the next sections:

1. Assignment of adjacency to manifold edges
2. An initial adjacency assignment to all non-manifold edges
3. An optimization phase that attempts to further reduce the total number of circuits

ADJACENCY ASSIGNMENT FOR MANIFOLD EDGES

Adjacency may be assigned trivially to manifold edges, since there are only two triangles for each such edge. An efficient process based on a sorting algorithm is proposed in this section.

The table TV does not contain any ordering or edge-face connectivity information. We must therefore establish a data-structure that makes it easy to identify all triangles that share a common edge. For this, we create a table ET, which identifies all triangles incident upon each edge. Each entry in ET corresponds to a half-edge. It contains:

- The indices ET[*h*].s and ET[*h*].e of the two vertices that bound half-edge *h.* These indices are sorted so that ET[*h*].s<ET[*h*].e
- The triangle ET[*h*].t associated with *h*
- The index ET[*h*].i of the vertex in TV that is opposite to *h*. Consequently, TV[ET[*h*].t,ET[*h*].i] is different from both ET[*h*].s and ET[*h*].e.

We compute a sorted index [Aho74] to this table using the concatenation of ET[*h*].s and ET[*h*].e as the key. In the sorted order, all the triangles incident upon a given edge are consecutive.

We simply traverse the table and when there are only two consecutive entries with the same key (ET[*h*].s==ET[*h*+1].s and ET[*h*].e==ET[*h*+1].e), we fill the corresponding entries in the TA table:

- TA[ET[*h*].t, ET[*h*].i] = ET[*h*+1].t
- TA[ET[*h*+1].t, ET[*h*+1].i] = ET[*h*].t

Fig. 7 illustrates this process on the example of a tetraheron.

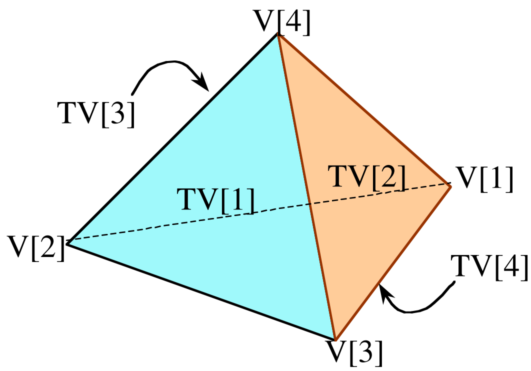

Figure 7: *This tetrahedorn has 12 half-edges and produces the following triangle table:*
TV[1]= (2, 4, 3), TV[2]= (1, 3, 4), TV[3]= (1, 4, 2), TV[4]= (1, 2, 3)
and the following sorted ET table of (ET[h].s, ET[h].e, ET[h].t,ET[h].i) entries:
(1 , 2 , 3 , 2), (1, 2 , 4 , 3), (1 , 3 , 2 , 3), (1 , 3 , 4 , 2), (1 , 4 , 2 , 2), (1 , 4 , 3 , 3),
(2 , 3 , 1 , 2), (2 , 3 , 4 , 1), (2 , 4 , 1 , 3), (2 , 4 , 3 , 1), (3 , 4 , 1 , 1), (3 , 4 , 2 , 1)
The resulting adjacency AT for manifold edges is:
TA[1]= (2, 4, 3), TA[2]= (1, 3, 4), TA[3]= (1, 4, 2), TA[4]= (1, 2, 3)

INITIAL ADJACENCY ASSIGNMENT TO NON-MANIFOLD EDGES

Once the adjacency of manifold edges is set, we compute an initial adjacency assignment for non-manifold edges. For this, we traverse the list of edges as explained in the previous section. In fact, both traversals may be combined, although we have separated them here for sake of simplicity. Non-manifold edges are identified as sequences of more than two consecutive edges in ET with the same key: ET[h].s==ET[h+i].s and ET[h].e==ET[h+i].e for i=1 to 2k–1. As we visit these edges, we insert them into a graph, whose nodes correspond to non-manifold vertices and whose links correspond to non-manifold edges. Each link of this graph identifies the corresponding entries in ET, which in turn point to the relevant entries in TA and in V. Thus, given a link in the graph, one can easily access the corresponding triangles, their vertices, and the adjacency map. Initially, the adjacency information for non-manifold edges is not filled, as we processed these edges, we do not remove them from the graph, but simply mark them as split. We will reuse the graph in the final optimization phase. to alter these adjacency maps in an attempt to minimize the total number of circuits.

As we build the graph, we also chain its vertices into a doubly linked list. Each time we visit a non-manifold edge, we inspect its two vertices. If they are not in the list, we append them. If they are, we remove them. At the end of the process, the list contains references to non-manifold vertices that have exactly one incident non-manifold edge. This is the list of *dead-ends* (i.e. the degree one nodes of the graph).

We “burn” these dead-ends, one at a time, as follows. We pick any dead-end, **V**, (the first one in our list) and compute its adjacency map as explained above. However, in this process, we do not treat the previously processed non-manifold edges as crossings or self-crossings. Instead, we follow the previously constructed adjacency information available in TA.

If we were to replace each previously split non-manifold edge by several manifold copies with the proper adjacency maps, we obtain a topology in which there is only one non-manifold edge, **E**, incident upon each dead-end **V**. Thus, when splitting **E**, we only need to trace one circuit. This circuit is the connected component of the union of the loops of the neighborhood of **V** that is connected to the unique crossing. We can start the traversal by picking an arc that leaves the crossing and follow the procedure outlined above for assigning the adjacency maps for the edges incident upon a non-manifold vertex.

The order in which triangles are visited as we follow the circuit of the neighborhood of **V**, defines the adjacency map for **E**. We update the corresponding TA entries to reflect this order. This traversal process *splits* the only non-manifold edge incident upon the current dead-end into two or more manifold edges, which all have the same starting vertex **V**, and the same ending vertex, **V’**. Consequently, the current dead-end becomes a manifold vertex and we remove it from the list of dead-ends. We visit **V’**, check whether it is a dead-end, and update the list if needed. Note that if only **V’** was a dead-end, it now is either a manifold vertex or an isolated non-manifold vertex.

We keep burning dead-ends until the list is empty. Remember that burning a dead-end may expose a single new dead-end. Therefore, the list can only shrink. If at the end of this dead-end burning process all edges in the graph have been split, this initial adjacency assignment phase terminates. Otherwise the sub-graph formed by the remaining unmarked non-manifold edges has one or more cycles, but no leaves.

In the latter case, we pick a node (non-manifold vertex) at random in the remaining sub-graph and compute an adjacency assignment for all unmarked non-manifold edges incident upon it. We use the same procedure as the one used for burning dead-ends. If this transformation exposes new dead-ends, we insert them in the dead-end list and restart the dead-end burning process explained above, until the list is empty again. This combination of splitting a non-manifold edge and then burning the resulting dead-ends recursively is repeated until all non-manifold edges are processed.

At the end of this phase, all non-manifold edges have been split and we have an edge-manifold topology and an imbedding without edge-crossings. We have developed a simple and efficient implementation of these two phases and have found that they produce a nearly optimal solution for the typical situations.

Nevertheless, we propose in the next section an optimization phase, which attempts to further reduce the total number of circuits by finding alternate ways of splitting non-manifold edges.

OPTIMIZATION PHASE

The final optimization phase uses the graph introduced above. It visits all its links and classifies them into three categories:

1. **Good:** Edges for which there exists a different split (i.e., adjacency map) that would decrease the total number of circuits if the adjacency assignments of the other edges were maintained constant.
2. **Bad:** Edges for which any other split would increase the total number of circuits, if the current splits for all other non-manifold edges were maintained.
3. **Neutral:** All other edges. These may be split in different ways without affecting the total number of circuits.

The *Clean-Up* procedure maintains a list of good edges. While the list is not empty, it computes the best split for the first good edge in the list, updates its classifications, updates the classification of all adjacent edges, and updates the list of good edges.

When the list of good edges is empty, we randomly pick a neutral edge, rematch it to produce a different split that does not increase the total number of circuits, and update the classification of all adjacent non-manifold edges. If any of these edges become good, we construct a list of good edges and perform a clean-up. This step, which combines a random perturbation followed by a clean-up is repeated until we decide that the process is not likely to further reduce the total number of circuits. We propose to stop this iteration when the number of consecutive perturbation steps that do not produce any good edges exceeds a prescribed fraction of the total number of non-manifold edges.

The main technical difficulty in this process is to find a better split for a given edge, **E**, if it exists and when a better split does not exist, to find a different split that does not reduce the total number of circuits, if such a split exists.

A non-manifold edge, **E**, corresponds to one crossing or self-crossing in the neighborhood of each one of its two bounding vertices, **V** and **V'**. Let **c** denote the crossing for **E** in **C(V,N)** and let **c'** denote the crossing for **E** in **C(V',N)**. Note that to each arc, **a**, incident upon **c** in **C(V,N)** corresponds a unique arc, **a'**, incident upon **c'** in **C(V',N)**. Each pair of such arcs is associated with the same triangle, **T**, incident upon **E**. Note that if **c'** is the starting point of **a'**, then **c** is the end point of **a**.

We can represent the triangles incident upon **E** as points on a circle, ordered according to their clockwise ordering around **E**. An adjacency map for **E** establishes connections between these points that lie inside the circle and do not cross. If we were to number the points around the circle, only connections between an odd and an even point would be possible, because when walking around **E,** each time we cross a triangle, we alternate between being in **N** and being in its complement.

Consequently, if there are 2k triangles incident upon **E**, there are k choices for the triangle adjacent to triangle 1. Each one of these choices splits the circle into two smaller circles, which no longer contain the point representing triangle 1, nor the point connected to it. One such process is illustrated Fig. 8.

For each candidate edge, **E**, we generate all possible adjacency maps, and for each map, we compute the number of circuits in the neighborhoods of both vertices of **E**. If the lowest number of circuits is less than the current one, the edge is good. Otherwise, if there are several maps with the current number of circuits, the edge is bad. Finally, if all but the current adjacency map increase the number of circuits, the edge is bad.

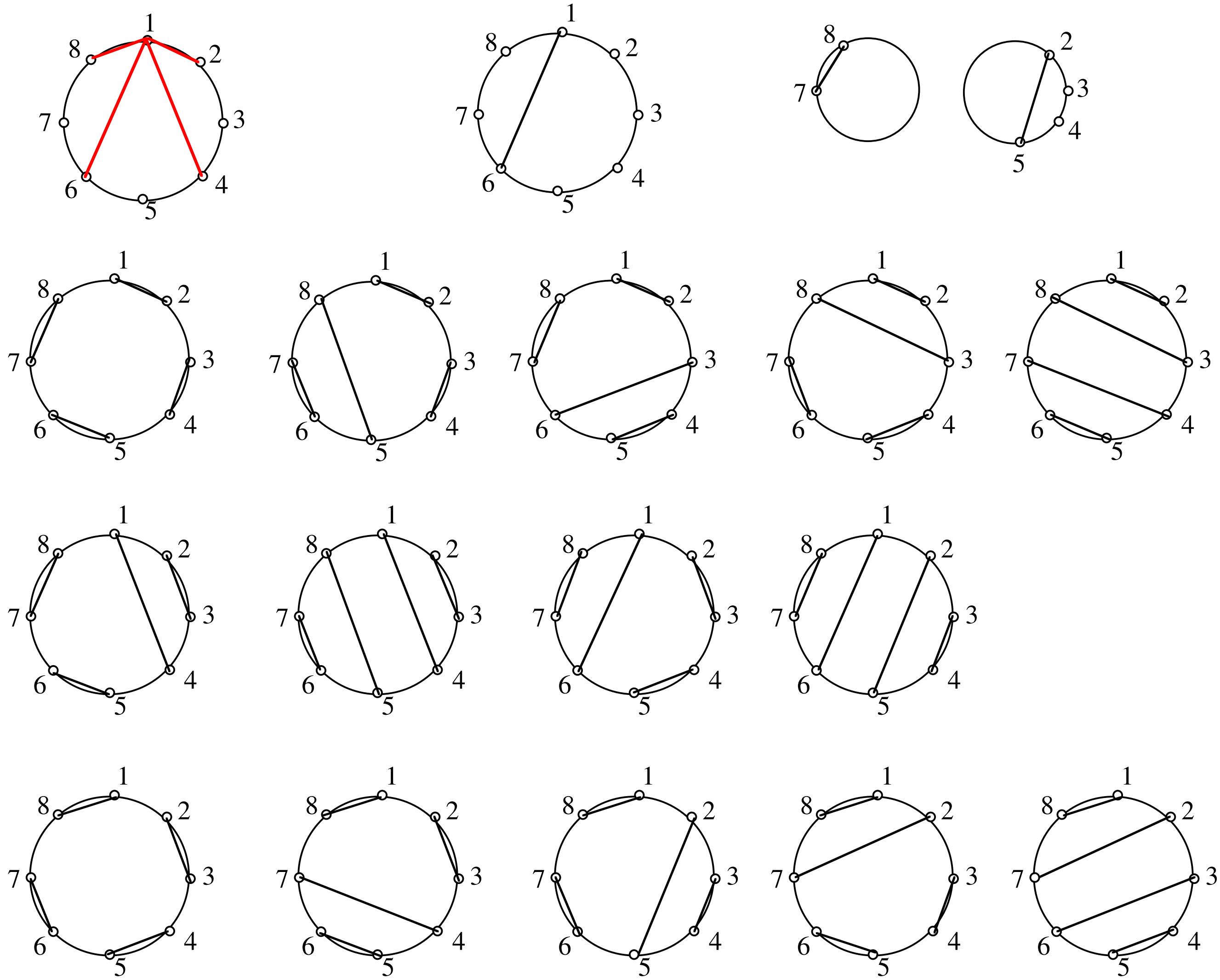

***Figure 8:** The adjacency map of a non-manifold edge, **E**, with 8 incident triangle may be encoded as a pairing of 8 points on a circle. The four choices of triangles adjacent to triangle 1 are shown with red lines (top left). One of these choices (top center) splits the other vertices in two sets (top right). There is only one match for the set {7,8}. There are two matches for the first vertex of the set {2,3,4,5}. One of them, (2,5) is illustrated on the right. The 14 possible adjacency maps are shown below.*

CONCLUSION

Given a set of quasi-disjoint triangles, represented by a vertex table, V, and by a triangle table TV, whose union is the boundary of a non-manifold r-set, Matchmaker computes for each edge an adjacency map which results in an edge-manifold topology with an imbedding that has coincident edges and vertices, but no edge-crossing.

The non-manifold vertices in this topology may be replicated if a manifold topology is required. Matchmaker attempts to minimize the total number of such vertex replications through a three phases process. Phase one assigns the adjacency of the manifold edges. Phase two computes an initial assignment for the non-manifold edges. Phase three perturbs this assignment randomly attempting to further reduce the number of non-manifold vertices.

The result of these three phases is an adjacency table, TA, which associates an adjacent triangle to each edge of each triangle. The only geometric test performed during this process is the ordering of incident triangles around non-manifold edges.

Given TV and TA, one can easily construct an extended vertex table by appending the additional copies of non-manifold vertices to V and by updating the entries in TA. A variety of manifold boundary data structures may be efficiently constructed from V, TV, and TA. These data structures are simpler and more compact. Furthermore, they support a family of efficient algorithms for compressing, simplifying, rendering, morphing, and intersecting polyhedral models.

REFERENCES

[Aho74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, MA, 1974.

[Alexandroff61] P. Alexandroff, Elementary Concepts of Topology, Dover Publications, New York, NY, 1961.

[Banerjee96] R. Banerjee and J. Rossignac, Topologically exact evaluation of polyhedra defined in CSG with loose primitives, to appear in Computers Graphics Forum, Vol. 15, No. 4, pp. 205-217, 1996.

[Baumgart72] B. Baumgart, Winged Edge Polyhedron Representation, AIM-79, Stanford University Report STAN-CS-320, 1972.

[Chow97] M. Chaw, Optimized Geometry Compression for Real-time Rendering, Proc. IEEE Visualization'97, pp. 347-354, Phoenix, AZ, October 19-24, 1997.

[Deering95] M. Deering, Geometry Compression, Computer Graphics, Proceedings Siggraph'95, 13-20, August 1995.

[Desaulniers92] H. Desaulniers and N. Stewart, An extension of Manifold Boundary Representations to the r-sets. ACM Transactions on Graphics, vol. 11, No. 1, pp.40-60, 1992.

[Eastman84] C. Eastman and K. Preiss, A review of solid shape modeling based on integrity verification, Computer-Aided Design, vol. 16, No. 2, pp. 66-80, March 1984.

[Evans96] F. Evans, S. Skiena, and A. Varshney, Optimizing Triangle Strips for Fast Rendering, Proceedings, IEEE Vizualization'96, pp. 319--326, 1996.

[Gottschalk96] S. Gottcshalk, M. Lin, D. Manocha, OBBTree: A hierarchical Structure for Rapid Interference Detection, Computer Graphics, Siggraph Proceedings, pp. 171-180, August 1996.

[Gueziec98] A. Gueziec, G.Taubin, F. Lazarus, W. Horn, Converting Sets of Polygons to Manifold Surfaces by Cutting and Stitching, Proc. IEEE Visualization, pp. 383:390, Research Triangle Park, NC, October 20-23, 1998.

[Gumhold98] S. Gumhold and W. Strasser, Real Time Compression of Triangle Mesh Connectivity. Proc. ACM Siggraph 98, pp. 133-140, July 1998.

[Gursoz91] E. Gursoz, Y. Choi, B. Prinz, Boolean set operations on non-manifold boundary representation objects. Computer-Aided Design, vol. 23, no. 1, pp. 33-39, January/February 1991.

[Heckbert97] P. Heckbert and M. Garland, Survey of Polygonal Surface Simplification Algorithms, in *Multiresolution Surface Modeling Course,* ACM Siggraph Course notes, 1997.

[Held95] M. Held, J. Klosowski, J. Mitchell, Evaluation of collision detection methods for virdual reality fly-through. Canadian Conference on Computational Geometry, 1995.

[Hoffmann89] C. Hoffmann, *Geometric and Solid Modeling: An introduction,* Morgan Kaufmann, San Mateo, CA, 1989.

[Hoppe96] H. Hoppe, Progressive Meshes, Proceedings ACM SIGGRAPH'96, pp. 99-108, August 1996.

[Hoppe97] H. Hoppe, View Dependent Refinement of Progressive Meshes, Proceedings ACM SIGGRAPH'97, August 1997.

[Horn89] W. Horn and D. Taylor, A theorem to determine the spatial containment of a point in a planar polyhedron, Computer Vision, Graphics, and Image Processing, 45:106-116, 1989.

[Kaley89] Y. Kaley, The Hybrid edge: A topological data structure for vertically integrated geometric modeling, Computer-Aided Design, vol. 21, no. 3, April 1989.

[Mantyla86] M. Mantyla, Boolean Operations of 2-manifold Through Vertex Neighborhood Classification, ACM Trans. on Graphics, 5(1):1-29, January 1986.

[Mantyla88] M. Mantyla, *An introduction to Solid Modeling.* Computer Science Press, 1988.

[Massey67] W. Massey, *Algebraic Topology: An Introduction,* Harcourt, Brace& World Inc., 1967.

[Murali97] T.M. Murali and T.A. Funkhouser, Consistent solid and boundary representations from arbitrary polygonal data, Proc. 1997 Symposium on Interactive 3D Graphics, ACM Press, pp. 155-162, Providence, R.I., April 1997.

[Requicha80] Representation of Rigid Solids: Theory, Methods, and Systems, A.A.G. Requicha, ACM Computing Surveys, 12(4), 437:464, Dec 1980.

[Requicha82] A. A. G. Requicha and H. B. Voelcker, Solid modelling: a historical summary and contemporary assessment, IEEE Computer Graphics and Applications, Vol. 2, No. 2, pp. 9-24, March 1982.

[Requicha83] A. A. G. Requicha and H. B. Voelcker, Solid modelling: current status and research directions, IEEE Computer Graphics and Applications, Vol. 3, No. 7, pp. 25-37, October 1983.

[Requicha85] A. A. G. and H. B. Voelcker, Boolean operations in solid modeling: boundary evaluation and merging algorithms, Proc. IEEE, Vol. 73, No. 1, pp. 30-44, January 1985.

[Requicha92] A. A. G. Requicha and J. R. Rossignac, Solid modeling and beyond, IEEE Computer Graphics & Applications, (Special issue on CAGD) Vol. 12, No. 5, pp. 31-44, September 1992.

[Rockwood98] A. Rockwood, K. Heaton, and T. Davis, Real-time Rendering of Trimmed Surfaces, Computer Graphics, 23(3):107-116, 1989.

[Ronfard96] R. Ronfard. and J. Rossignac, Full-range approximation of triangulated polyhedra, Proc. Eurographics'96 , Computer Graphics Forum, pp. C-67, Vol. 15, No. 3, August 1996.

[Rossignac89] J. Rossignac and M. O'Connor, SGC: A Dimension-independent Model for Pointsets with Internal Structures and Incomplete Boundaries, in *Geometric Modeling for Product Engineering*, Eds. M. Wosny, J. Turner, K. Preiss, North-Holland, pp. 145-180, 1989.

[Rossignac94] J. Rossignac, Through the cracks of the solid modeling milestone, *From Object Modelling to Advanced Visual Communication,* Eds. Coquillart, Strasser, Stucki, Springer-Verlag, pp. 1-75, 1994.

[Rossignac94b] *J. Rossignac and Anil Kaul, AGRELs and BIPs: Metamorphosis as a Bezier curve in the space of polyhedra, Computer Graphics Forum, Vol 13, No 3, pp. C179-C184, Sept 1994.*

[**Rossignac97**] J. Rossignac, Structured Topological Complexes: A feature-based API for non-manifold topologies, ACM Press, C. Hoffman and W. Bronsvort, Edts., pp. 1-9, 1997.

[Rossignac98] J. Rossignac, Edgebreaker: Compressing the incidence graph of triangle meshes, GVU Technical Report GIT-GVU-98-35, Georgia Institute of Technology, http://www.cc.gatech.edu/gvu/reports/1998.

[Rossignac98b] J. Rossignac, 3D Geometry Compression: Just-in-time upgrades for triangle meshes, in *3D Geometry Compression,* Course Notes 21, Siggraph 98, Orlando, Florida, July 18-24, 1998.

[**Segal90**] M. Segal, Using tolerances to guarantee valid polyhedral modeling results. *Computer Graphics,* vol. 24, No. 4, pp. 105-114, August 1990.

[Sugihara89] K. Sugihara and M. Iri, A solid modeling system free from topological inconsistency. Journal of Information Processing, vol. 12, No. 4, pp. 380-393, 1989.

[Taubin98] G. Taubin and J. Rossignac, Geometric Compression through Topological Surgery, ACM Transactions on Graphics, Volume 17, Number 2, pp. 84-115, April 1998.

[Taubin98c] G. Taubin, A. Gueziec, W. Horn, F. Lazarus, Progressive Forest Split Compression, Proc. ACM Siggraph 98, pp. 123-132, July 1998.

[Tilove80] R. Tilove, Set Membership Classification: A Unified Approach to Geometric Intersection Problems, IEEE Trans. on Computers, vol. C-29, NO. 10, pp. 874-883, October 1980.

[Touma98] C. Touma and C. Gotsman, Triangle Mesh Compression, Proceedings Graphics Interface, pp. 26-34, 1998.

[**Weiler85**] K. Weiler, Edge-based data structures for solid modeling in curved surface environments, IEEE Computer Graphics and Applications, vol. 5, No. 1, pp. 21-40, 1985.

[**Weiler86**] K. Weiler, Non-Manifold Geometric Boundary Modeling, ACM Siggraph, Tutorial on Advanced Solid Modeling, Anaheim, California, July 1987.

[Weiler87] K. Weiler, Topological structures for geometric modeling, PhD Thesis, Rensselaer Poytechnic Institute, Troy, NY, 1986.

[Who85] T. Who, A combinatorial Analysis of Boundary Data Structure Schemata, IEEE Computer Graphcis and Applications, Vol. 5, No. 3, pp. 19-27, 1985.