Topological Mesh Operators

Luiz Velho [a] Hélio Lopes [b,*] Esdras Medeiros [a] Thomas Lewiner [b,c]
Geovan Tavares [b]

[a] *Laboratório VISGRAF – IMPA – Rio de Janeiro – Brazil.*
[b] *Laboratório Matmídia – PUC-Rio – Rio de Janeiro, Brazil.*
[c] *Geométrica Project – INRIA – Sophia Antipolis, France.*

Abstract

In this paper we introduce an unified framework for basic operations on combinatorial 2-manifolds with or without boundary. We show that there are two kinds of primitive operators on the underlying meshes: operators which change the topological characteristic of the mesh and operators which just modify its combinatorial structure. We present such operators and demonstrate that they provide a complete set of elementary operations for mesh modification. We also give a description of the algorithms and data structures for an efficient implementation of these operators.

Key words: Geometric Modeling, Handle Operators, Stellar Operators.

1 Introduction

Polygonal meshes constitute one of the fundamental representations for objects in computer graphics and geometric modeling. They describe the spatial support where attributes of the objects are defined, such as geometry and texture.

* *Email addresses:* lvelho@visgraf.impa.br (Luiz Velho), lopes@mat.puc-rio.br (Hélio Lopes), esdras@impa.br (Esdras Medeiros), tomlew@mat.puc-rio.br (Thomas Lewiner), tavares@mat.puc-rio.br (Geovan Tavares).

In spite of the fact that other representations, such as point sets, are becoming increasingly popular in recent years, polygonal representations are still prevalent and will always be necessary in one way or another. The main reason is that meshes describe in a convenient piecewise manner the *global* space, intrinsic to the object. Point sets, on the other hand, provide only a *local* description. Indeed, the generation of polygonal meshes from point data is an active area of research.

Two dimensional surfaces are, arguably, the most common type of object in computer graphics. Moreover, we are often interested in non-degenerate surfaces, i.e. 2D manifolds. These objects are best represented by a simplicial mesh, or a *combinatorial manifold structure*.

Contributions: In this paper we investigate operators to build and modify combinatorial manifolds with or without boundary.

The main contributions of this work are:

- The introduction of an unified framework for primitive operations on combinatorial 2-manifolds with or without boundary. This framework is based on the integration of two fundamental theories in computational topology: the Handlebody theory and the Stellar theory.
- The definition of a complete and sufficient set of operators to change the combinatorial structure, as well as, the topological characteristic of a polygonal mesh. This result is substantiated by the main theorems of the Handlebody and Stellar theories.
- The description of an implementation of these basic mesh operators, including the specification of the data structures used for mesh representation.

We also discuss how the proposed mesh operators can be applied in the solution of several problems in geometric modeling and computer graphics. We present examples of prototypical applications and point out how the framework could be incorporated with advantages in previously known algorithms.

Paper outline: Section 2 describes some previous and related works. Section 3 introduces some concepts of combinatorial topology and presents the Handlebody and Stellar theories. Section 4 proposes the complete set of operators for surface modeling. Section 5 presents the data structures and algorithms. Finally, section 7 concludes this work by giving some final remarks and suggestion for future works.

2 Related Works

The representation of a surface by a polygonal mesh has two components that describe its topology and geometry, respectively. The topological component defines neighborhood relations within the surface, while the geometric component defines the shape embedding in ambient space.

In this paper, we are mainly concerned with the combinatorial structure of a polygonal mesh. For this reason, we will not address geometric issues extensively here. Nonetheless, we note that there is an interdependency between the implementation of geometric operations and the combinatorial representation of a mesh. Related work in the area fall into three categories: topological data structures; topological operators; and geometric operators.

The neighborhood relations within a mesh are encoded by a topological graph that indicates incidence relationships among vertices, edges and faces. Thus, the major issue in terms of topological data structures, is the trade-off between: size of the representation; time complexity for queries; and flexibility of making structural changes.

Practically all topological data structures are based on edges. The classical structure is the winged edge [2], which links vertices and faces, and also includes information about orientation. Variations of this basic structure, such as the half-edge [18], decouple the two uses of an edge by a face (i.e., incidence and orientation), and encode the orientation in an implicit way.

The quad-edge [10] is also an edge-based data structure, but is able to represent both the primal and dual graphs of the mesh. Another important characteristic of the quad-edge data structure is that it was defined together with an edge algebra (see comments below). As an alternative to the quad-edge, Shewchuk [27] proposed a triangle-based data structure that supports equivalent operations. A similar triangle-based data structure is the corner table [26], which is optimized for compression.

All the above data structures were designed to represent only manifold surfaces. The radial edge [35] is a data structure that can represent non-manifold surfaces, as well. In many applications, the radial edge is used to describe the non-manifold skeleton of a space decomposition in $\mathbb{R}^n$. We note that, in such cases, it suffices to use a manifold structure in $\mathbb{R}^{n+1}$.

In this paper we adopt an edge-based mesh representation. It is similar to the half-edge, but it is enhanced to support manifolds with boundary. Such data structure shows to be very suitable for an efficient implementation of the proposed mesh operators.

Operations on a surface are implemented through operators on its mesh representation. These operators can be classified according to various criteria, such as level of abstraction and functionality.

Euler operators [18] are low-level operators for editing a mesh representation of the boundary of a solid. They are based on the Euler-Pointcaré theory. This theory says that an orientable combinatorial surface S with boundary is uniquely identified by its Euler characteristic $\chi(S) = |V| - |E| + |F|$, where $|V|, |E|$ and $|F|$ indicate respectively the number of vertices, edges and faces of S. The Euler characteristic classifies the surface according to the Euler formula that is $\chi(S) = 2s - 2g - b$ (where s is the number of connected components, g is the number of holes or genus, and b is the number of boundary curves of the surface). This formula states that the genus on an orientable combinatorial surface with s connected components and b boundary curves is $g = s - b/2 - (|V| - |E| + |F|)/2$.

Mäntylä proved that Euler operators form a complete set of modeling primitives for manifold solids. That is, every topologically valid polyhedron can be constructed from an initial polyhedron by a finite sequence of Euler operators. There are two groups of such operations: the *make* group and the *kill* group. One disadvantage of the Euler operators is that, in the process of editing a mesh with these atomic operations, some intermediate results may not represent valid solids. Moreover, the Euler operator that generates a genus, assume that the 2-manifold being operated is the boundary of a solid in $\mathbb{R}^3$. Therefore, Euler operators are usually encapsulated into higher level operators.

Quad-edge operators are low-level operators based on the edge algebra defined in [10]. Their main advantage is conciseness. Guibas and Stolfi showed that only two atomic operations are sufficient for the construction and modification of arbitrary topological graphs embedded in two-dimensional manifolds.

The operators proposed in this paper work at a higher-level than the ones above, and, as such, could be defined either in terms of the Euler or quad-edge operators — although this is not necessary. Here we have chosen to define them directly, as atomic operations, since we believe that they provide the right level of abstraction. In addition, they are not restricted to $\mathbb{R}^3$, i.e., they don't depend on the space where the surface is embedded.

Because of their importance in applications, many high level operators have been proposed to change the resolution of a mesh. These operators can be used for mesh simplification or mesh refinement. The meshes that they operate on can have regular or irregular connectivity.

Multiresolution operators for regular meshes are usually associated with subdivision algorithms. In this area, the classical operators are the quadrisection for faces [5], [16] and vertices [7] (e.g. *primal* and *dual* refinement). The drawback of these operators is that they cannot be used for adaptive refinement without compromising the regularity of the mesh. Recently, two new schemes, $\sqrt{3}$ subdivision [13] and $\sqrt{2}$ subdivision [33], introduced operators that are suitable for adaptive refinement. These schemes employ trisection and bisection operators, respectively.

The most popular multiresolution operators for irregular meshes are the *edge collapse* and its inverse, the *edge split*. Hoppe [12] proved that these two operators can be used to transform between any two equivalent simplicial complexes. Although edge collapse was designed originally in connection with progressive meshes [11], it has also been extensively used in many mesh simplification methods [8].

In this paper we introduce a set of high-level operators that can be used to make changes both on the combinatorial and topological structure of the mesh. In that sense, our operators have more expressive power than the multiresolution operators discussed above and can be used to implement them.

As we mentioned before, geometric operations are not the focus of this paper. Nonetheless, we would like to briefly discuss their relationship with topological operators.

Some geometric operations, such as warping deformations, are defined only in terms of pointwise information of a shape embedded in the ambient space. Therefore, this type of operators are independent of the mesh structure.

Other geometric operators, such as the *umbrella* operator [28] used in Laplacian smoothing, depend on the local geometry of the surface. There are also operators that associate geometric quantities with elements of the mesh, for example differential properties [6]. These two types of operators need information about the neighborhood of a topological entity, and, thus, they rely on queries about the mesh structure.

The data structure proposed in this paper supports efficient mesh queries and can be augmented with geometric atributes associated with different topological elements. Thus, it is suitable for the implementation of geometric operators.

3 Fundamental Concepts

In this section, we lay out the fundamental concepts of our framework for mesh operations. We distinguish between two kinds of operators on meshes: query operators and modification operators. We can further classify the modification operators into two types: the ones that change the topology of the mesh, and the ones that just alter its combinatorial structure.

We will see next, that basic concepts from topology are sufficient to define query operators. Operators that change the mesh topology are based on the Handlebody theory, while operators that alter the mesh structure are based on the Stellar theory.

3.1 Basic Topological Concepts

A simplex σ^p of dimension p (p-simplex, for short) is the convex hull of $p+1$ points $\{v_0,...,v_p\}, v_i \in \mathbb{R}^m$, in general position, i.e., the vectors $v_1 - v_0, v_2 - v_0, ..., v_p - v_0$ are linearly independent. The points $v_0,...,v_p$ are called the vertices of σ. A face of σ is the convex span of some of the vertices of σ and therefore is also a simplex. The simplices of dimensions 2 and 1 will be called, respectively, triangles, edges. If σ is a face of a simplex τ then σ is said to be incident to τ. The boundary of a p-simplex σ, denoted by $\partial\sigma$, is the collection of all of its proper faces, i.e., different from σ itself. Two k-simplices σ and $\rho \in K$ are *adjacent* when $\sigma \cap \rho \neq \emptyset$, and *independent* otherwise. The valence or degree of a vertex $v \in K$ is the number of edges which have v as a vertex, and is denoted by $\deg(v)$.

A simplicial complex K is a finite set of simplices together with all its subsimplices such that if σ and τ belong to K, then either σ and τ meet at a subsimplex λ, or σ and τ are independent.

A complex K is *connected* if it cannot be represented as a union of two non-empty disjoint subcomplexes L and M without common simplexes. A *component* of a complex K is a connected subcomplex that it is not contained in a larger connected subcomplex of K.

The *underlying polyhedron* $|K| \subset \mathbb{R}^m$ corresponds to the union of the simplexes in K. A triangle mesh is the underlying polyhedron of a 2-dimensional simplicial complex.

The *join* $\sigma \star \tau$ of independent simplices σ and τ is the simplex whose vertices are those of σ and τ. The join of complexes K and L, written $K \star L$, is $\{\sigma \star \tau : \sigma \in K, \tau \in L\}$ if the following holds:

(1) If $\sigma \in K$ and $\tau \in L$, σ and τ are independent.
(2) If $\sigma_1, \sigma_2 \in K$ and $\tau_1, \tau_2 \in L$, then $\sigma_1 \star \tau_1 \cap \sigma_2 \star \tau_2$ is either empty or a face of $\sigma_1 \star \tau_1$ and $\sigma_2 \star \tau_2$.

Consider a simplicial complex K and $\sigma \in K$. The local neighborhood of σ is described by the following elements:

- The *open star* of σ is

$$star(\sigma, K) = \{\tau \in K : \sigma \text{ is a face of } \tau\}.$$

- The *star* of σ is

$$\overline{\text{star}}(\sigma, K) = \{\tau \in K : \tau \text{ is a face of an element of } star(\sigma, K)\}.$$

- The *link* of σ is

$$link(\sigma, K) = \{\tau \in K : \tau \text{ and } \sigma \text{ are independent and } \sigma \star \tau \in K\}.$$

Definition 1 (combinatorial surface) *A simplicial complex S, $|S| \subset \mathbb{R}^m$ is a combinatorial surface if: Every edge in S is bounding either one or two triangles and the link of a vertex in S is homeomorphic either to an interval or to a circle.*

The edges in a combinatorial surface S incident to only one face are called *boundary edges*. A vertex incident to a boundary edge is called a *boundary vertex*. The subcomplex of S of those boundary simplices forms the *boundary* of S and is denoted by ∂S. The boundary of a combinatorial surface is a collection of closed curves. The edges and vertices that are not on the boundary are called, respectively, *interior edges* [1] and *interior vertices*.

The set of faces, edges and vertices of a surface S will be denoted, respectively, by $F(S), E(S)$ and $V(S)$.

3.2 Handlebody Theory

The topological setting applied to boundary representation of solids [2] has traditionally been the Euler-Poincaré theory, dated from the turn of the 19^{th} century [24].

The Handlebody theory [21] refines the Euler-Poincaré theory by bringing several new topological invariants for n-dimensional manifolds. The fundamental problem of Handlebody theory is to study the topological changes generated by handle attachments to a manifold with boundary.

In the surface case, three types of handles are to be defined and they will be distinguished by an index λ that varies from 0 to 2. Here, D^i denotes the i-dimensional disk and ∂P the boundary of a set P.

Definition 2 *A handle of index λ, denoted by H_λ, is a pair of topological spaces (A_λ, B_λ) such that $B_\lambda \subset A_\lambda$, $A_\lambda = D^\lambda \times D^{2-\lambda}$ and $B_\lambda = \partial D^\lambda \times D^{2-\lambda}$.*

According to this definition, one can observe that: 1) the set A_0 is a 2-disk and B_0 is the empty space; 2) the set A_1 is a square and B_1 is defined to be two of its opposite sides and 3) the set A_2 is a 2-disk and B_2 is its boundary (see Figure 1).

To attach a handle $H_\lambda = (A_\lambda, B_\lambda)$ to the boundary of a surface S means to identify by a homeomorphism the set B_λ with a subset I contained in the boundary of S that is homeomorphic to B_λ.

[1] Observe that the *link* of an interior edge is the pair of opposite vertices.

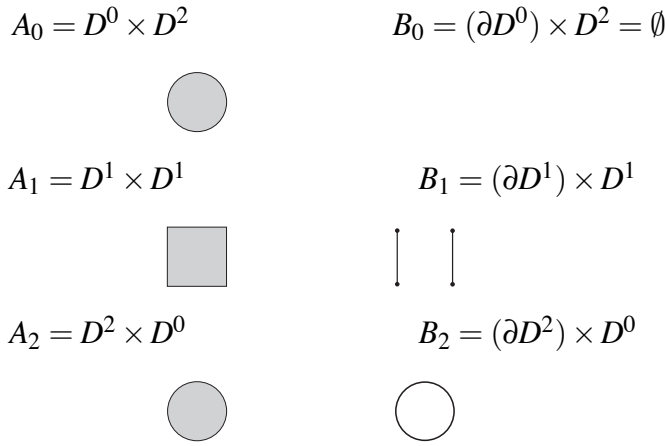


Fig. 1. The 2D Handles: $\mathbf{H}_0 = (A_0, B_0)$; $\mathbf{H}_1 = (A_1, B_1)$; $\mathbf{H}_2 = (A_2, B_2)$.

The next theorem is the main mathematical tool in which the Handlebody Theory is based.

Theorem 3 *(Handlebody Decomposition) For every surface S there is a finite sequence of surfaces* $\{S_i\}$, $i = 0..N$, *such that* $S_0 = \emptyset$, $S_N = S$ *and the surface* S_i *is obtained by attaching a handle* $H_\lambda = (A_\lambda, B_\lambda)$ *to the boundary of* S_{i-1}. *This sequence is called the Handlebody Decomposition of S.*

Figure 2 illustrates the handlebody decomposition of a torus, $S_4 = (((S_0 + H_0) + H_1) + H_1) + H_2$.

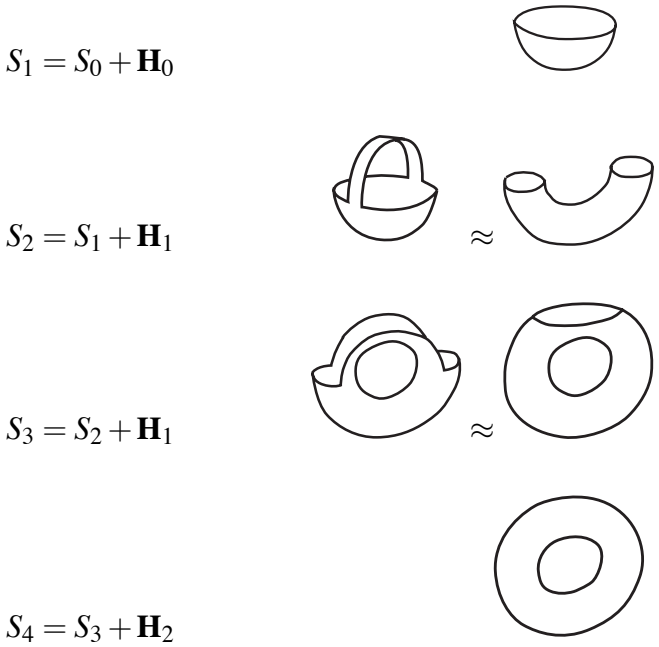


Fig. 2. Handlebody decomposition of a torus, $S_4 = (((S_0 + H_0) + H_1) + H_1) + H_2$.

When a handle $H_\lambda = (A_\lambda, B_\lambda)$ is attached to the boundary of S_{i-1} to obtain S_i, a topological change is generated and such change depends *only* on the index λ.

Theorem 4 *If S_i is obtained by attaching the handle H_λ to S_{i-1}, then $\chi(S_i) = \chi(S_{i-1}) + (-1)^\lambda$.*

As a consequence, the Euler characteristic of a surface S provided with a handlebody decomposition $\{S_i\}$, $i = 0..N$ is

$$\chi(S) = |H_0| - |H_1| + |H_2|$$

where $|H_k|$, $k \in \{0,1,2\}$ corresponds to the number of handles of type k in $\{S_i\}$. For example, in the handlebody decomposition of the torus in Figure 2, there are one handle H_0, 2 handles H_1, and 1 handle H_2. The formula above is, then, verified, since the Euler characteristic of a torus is zero. This is a new topological invariant introduced by the Handlebody theory.

Handles can be attached to an orientable surface with boundary in such a way to preserve its orientability, i.e., the identification has to be coherent. If one starts with an orientable surface, then after attaching a handle coherently the surface is again orientable.

We observe that if we keep track the number of connected components and the number of boundary curves, we can easily calculate the number of genus on the surface and classify it whenever it is necessary. We are to present how to count those two numbers by studying the topological changes caused by a handle attachment that preserves the orientability.

The topological change generated by a handle attachment of index 0 is a creation of a new surface component (see S_1 in Figure 2). This handle attachment increases the Euler characteristic by one.

When the handle H_1 is coherently attached to a surface S_i, three situations can occur:

(1) The set A_1 is attached to disjoint intervals on the same boundary curve component. In this case, the topological change is a inclusion of a new boundary curve component in the surface (see S_2 in Figure 2).
(2) The set A_1 is attached to intervals on different boundary curve components of a surface component. The topological change is here characterized by the creation of a new genus on the surface. In addition, the number of boundary curve components decreases (see S_3 in Figure 2).
(3) The set A_1 is attached to intervals on different surface components. Here, a boundary curve component and a surface component is removed.

In these three situations, when a handle H_1 is attached coherently to S_{i-1} to obtain S_i, we have $\chi(S_i) = \chi(S_{i-1}) - 1$. Notice that, all of them alter the number of boundary curves. Moreover, the last one also changes the number of connected components on the surface.

Handles of index 2 close a boundary curve component (see S_4 in Figure 2).

Concluding, there are three types of handles and five different situations in which they can be attached to a boundary surface.

3.3 Stellar Theory

In the previous section, saw how to change the topology of a manifold. Now, we will see how to manipulate the structure of a combinatorial surface *without* modifying its topology, which is the main point of Stellar theory [1, 22, 23, 15].

As we have seen in Section 3.1, the link and the star of a simplex σ provide a combinatorial description of the neighborhood of σ. We can use them to define certain changes in a triangle mesh, without modifying essentially (i.e., "topologically") that neighborhood. That is, we do not want to change the topology of the realization of the surface in $\mathbb{R}^3$. The *stellar operations* provide a such change. They comprise *bistellar moves* and *stellar subdivision*:

Definition 5 *Let K be an n-dimensional simplicial complex. Take an r-simplex $\sigma \in K$, and an $(n-r)$-simplex $\tau \notin K$, such that $link(\sigma, K) = \partial\tau$. Then, the operation $\kappa(\sigma,\tau)$, called* bistellar move*, consists of changing K by removing $\sigma \star \partial\tau$ and inserting $\partial\sigma \star \tau$.*

The bistellar moves are atomic operations that make local changes to the neighborhood of an simplex, while maintaining the integrity of its combinatorial structure. In the case of combinatorial surfaces, there are three types of bistellar moves, for $\dim\sigma = 2,1,0$, called 2-move, 1-move, and 0-move. They are shown in figure 3.

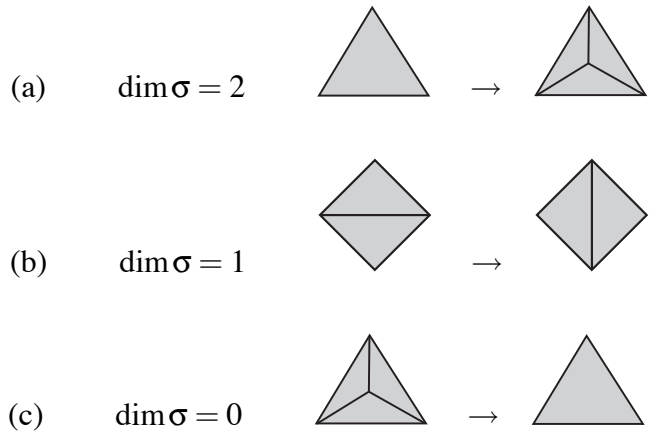


Fig. 3. Two dimensional bistellar moves.

The fundamental result of the Stellar theory is given by the following theorem:

Theorem 6 *([22], [23]) Two combinatorial surfaces are piecewise linearly homeomorphic if and only if they are bistellar equivalent.*

The above result guarantees that bistellar moves can change any triangulation of a closed piecewise linear manifold to any other. A version of this theorem for manifolds with boundary uses all stellar operations, including stellar subdivision [23].

Definition 7 *Let K be a 2-dimensional simplicial complex, take an r-simplex $\sigma \in K$ and a vertex ν in the interior of σ.*

The operation (σ,ν) removes $\overline{star}(\sigma,K)$ and replaces it with $\nu \star \partial\sigma \star link(\sigma,K)$ is called a stellar subdivision.

The inverse operation $(\sigma,\nu)^{-1}$ is called a stellar weld.

Note that, some of the stellar subdivision and welds are also stellar moves. For example, in the two dimensional case, for $\dim\sigma = 2$, see $\kappa(\sigma,\nu)$ and $\kappa(\nu,\sigma)$ that are shown in the top and bottom rows of figure 3.

The new operation in two dimensions, is the stellar subdivision on edges, called 1-split. It is shown in figure 4 the interior edge case and in figure 5 the boundary edge case.

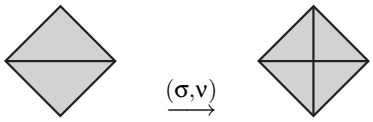


Fig. 4. Two dimensional stellar subdivision on interior edges.

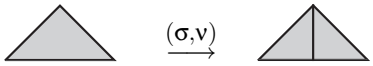


Fig. 5. Two dimensional stellar subdivision on boundary edges.

Stellar subdivision is a very powerful concept and it is the cornerstone of Stellar theory. Here, we will only mention some results of the stellar subdivision theory [1].

Proposition 8 *Any stellar move, $\kappa(\sigma,\tau)$, is the composition of a stellar subdivision and a weld, namely $(\tau,\nu)^{-1}(\sigma,\nu)$.*

This result can be easily seen through an example, shown in figure 6.

Proposition 9 *Any stellar operation can be decomposed into a finite sequence of elementary stellar operations on* edges.

This result is even stronger than the previous one. It basically allows us to restate the main theorem of Stellar theory only in terms of operations on edges.

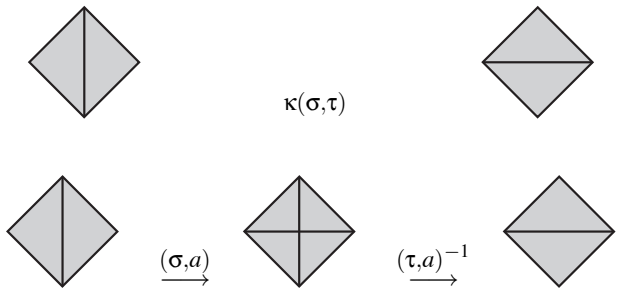


Fig. 6. A bistellar move on an edge can be decomposed into a subdivision and an weld.

4 Computational Framework

The purpose of this section is to introduce a new representation for surfaces based on the concepts of Handlebody and Stellar theories. It consists of a topological data structure that describes the incidence and adjacency relations on a combinatorial surface with or without boundary. It also includes operators for building/unbuilding meshes and to change the structure and resolution of a mesh.

We remark that, although the Handlebody theory can be applied to general combinatorial manifolds, the Stellar theory is restricted to simplicial complexes. Therefore, from now on, we will focus primarily on triangular meshes. This is not a limitation, since any manifold surface can be triangulated and, in practice, triangular meshes are a common choice in applications.

4.1 Handle Operators

The Handlebody theory presented in Section 3.2 studies the topological changes in a surface caused by a handle attachment. There are three types of handles to build a handlebody decomposition of a surface. From a combinatorial point of view, three types of operators are to be defined to represent the handle attachments:

- Handle operator of type 0 – This operator creates a new combinatorial surface component with only one triangle (see Figure 7).
- Handle operator of type 1 – The purpose of this operator is to identify two given boundary edges *with no vertices in common*. There are three situations for this group:
  Case 1: the boundary edges are on different surfaces. In this case the operator attaches the surfaces and removes one boundary curve (see Figure 8(a)).
  Case 2: the given boundary edges are incident to the same boundary curve. The operator splits the boundary curve into two different components (see Figure 8(b)).
  Case 3: the boundary edges are on different boundary curves on a surface component. It creates a new genus in the surface and reduce in one the number of boundary curve components of the surface (see Figure 8(c)).

- Handle Operator of type 2 – In this group the operator identify two given boundary edges *with two vertices in common*. The operator closes one boundary curve component and transform those boundary vertices into two interior vertices (see Figure 9).

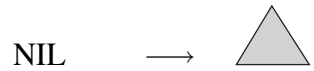
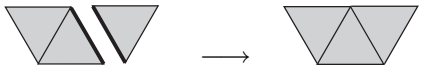
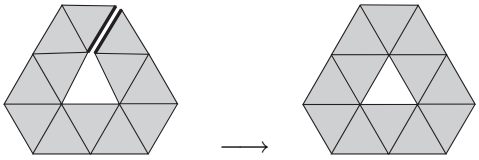


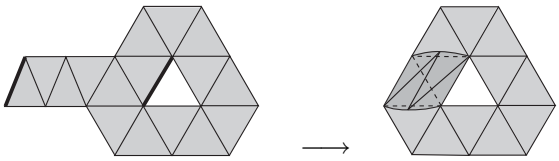
Fig. 7. Handle operator of type 0 (triangle creation).



(a) Boundary edges belong to different surfaces

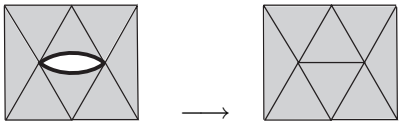


(b) Boundary edges belong to the same boundary curve of a surface



(c) Boundary edges belong to different boundary curves of a surface

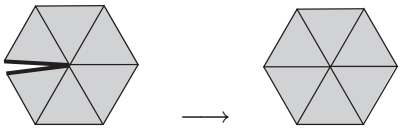
Fig. 8. Handle operator of type 1 (joining boundaries).



(a) Boundary edges have two vertices in common

Fig. 9. Handle operator of type 2 (closing boundaries).

According to the definitions above, we notice that if a Handle operator of type λ is applied to a combinatorial surface S_1 to obtain S_2, then $\chi(S_2) = \chi(S_1) + (-1)^{\lambda}$. This is a direct consequence of theorem 4.



(a) Boundary edges have one vertex in common

Fig. 10. Zip operator.

One can observe that the Handle operators of type 1 and type 2 identify two boundary edges to make an interior edge. The first is applied when the edges have no vertices in common, and the second when the edges have two vertices in common. Thus, there is one missing case to consider: when the boundary edges have one vertex in common. So, it is suitable to define the Zip operator, which identifies two boundary edges with one vertex in common. This operator removes one edge and one vertex, then it doesn't change the Euler characteristic of the surface. Its main purpose is to close the vertex link (see Figure 10). In fact, such operator can be derived from the Handle operators together with their inverse. However, it is very convenient to have a direct implementation of it.

4.1.1 Inverse Handle Operators

There is an inverse operator not only for each handle operator, but also for the Zip operator. The topological changes caused by their inverse operation are now described.

The inverse handle operator of index zero destroys a triangle. Inverse handle operators of index 1 and index 2 split an interior edge into two boundary edges. There are five cases to consider when splitting an interior edge. Such cases are distinguished according to the number of boundary vertices incident to the interior edge that will be operated, which could be 2,1 or 0. The inverse handle operator of type 1 is used when the incident vertices to the interior edge are both in the surface boundary. The inverse handle operator of type 2 is applied when the incident vertices of the interior edge are on the interior of the surface. In the last case, when the interior edge has one vertex in the boundary, one should use the inverse Zip operator.

The topological changes caused by an inverse handle operator of index 1 when applied to a given interior edge e, depend on the answer to the following question:

Are the incident boundary vertices to e on different boundary curve components?

If the answer is affirmative then the inverse handle operator will remove one boundary curve component (see the inverse operation in the Figure 8(b)). In contrary, the second question has to be answered.

Are those edges on the same boundary curve component?

When the vertices are incident to the same boundary curve, the inverse operation not only will add a new boundary curve component to the surface but also it will either remove a genus (see inverse of Figure 8(c)) or disconnect the surface (see inverse of Figure 8(a)).

Inverse operator of index 2 splits an interior edge with zero incident boundary vertices. The topological change in this situation is an addition of a new boundary curve to the surface.

The inverse Zip operator (the unzip op.) is applied when the interior edge e has one incident vertex on the boundary. It simply splits an interior edge and transforms an interior vertex into a boundary vertex.

With the set of operators presented above one can easily build and unbuild all kinds of orientable combinatorial surfaces, with or without boundary. Handle operators shall be used to perform paste operations on the surface, while the inverse operators shall be used to make cut operations.

4.2 Stellar Operators

The Stellar theory presented in Section 3.3 studies structural modifications to the neighborhood of a simplex that do not alter the topology. These modifications are the stellar moves, stellar subdivision and welds. They can be used to change the connectivity and the resolution of a mesh.

We classify the stellar operators in terms of their effect in the number of faces, $|F|$, in the mesh. Accordingly, there are three groups of operators: *isolevel*, *refinement*, and *simplification*.

- The isolevel operators do not change $|F|$. The operator in this group is the bistellar 1-move, also called 1-flip (or edge flip). It simply exchanges two existing triangles by two new triangles. This operator is shown in Figure 3(b).
- The refinement operators increase $|F|$, and the resolution of the mesh. The operators in this group are the 2-split (face split), and 1-split (edge split). The face split replaces one existing triangle with three new triangles, and thus, it increases $|F|$ by 2. This operator is shown in Figure 3(1). The edge split replaces two existing triangles sharing that edge with four new triangles, when the edge is an internal edge. When the edge is a boundary edge, it replaces one existing triangle with two new triangles. This operator increases $|F|$, by 1 or 2, depending of whether the edge belongs to the boundary or not. Figure 4 shows the 1-split of an internal edge.
- The simplification operators are the inverse of the refinement operators. The inverse of the face split is the face weld, and the inverse of the edge split is the edge weld. Observe that, weld operations $(\sigma,\nu)^{-1}$, are specified through a vertex ν, whose star defines the neighborhood to be changed.

At this point it is appropriate to note that stellar operators can be used as primitives to define other multiresolution operators. For example, edge collapse and its inverse, vertex split, can be decomposed into a sequence of elementary stellar operations. This is a natural consequence of Theorem 6. More specifically, the edge collapse is given by a composition of edge flips and a final edge weld, while the vertex split is given by an edge split composed wit a sequence of edge flips. This is shown in Figure 11.

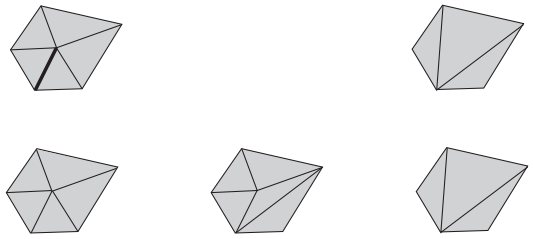


Fig. 11. Decomposition of an edge collapse (top) into an edge swap followed by an edge weld (bottom).

We remark that stellar operations are more flexible in general. In the case of edge collapse / vertex split, it is easy to see that there are many possible sequences of edge flips leading to the final edge weld. Therefore, these edges flips can be chosen such that the quality of the mesh is improved (for example, a measure of aspect ratio).

5 Implementation

The implementation for a mesh library based on the framework proposed in this paper, consists of the set of topological, handlebody and stellar operators.

The mesh representation itself employs a edge-based data structure that describes the mesh connectivity.

5.1 Mesh Operators

The basic topological operators allow queries and navigation of the mesh structure. They are: `c = link(s);` and `c = star(s)`. Note that they can take as arguments a simplex `s` of dimension 0 (vertex), 1 (edge) or 2(face). In our current implementation, we use only the vertex star, which returns an adjacency iterator object `c`, called *circulator* [20]. We also have the basic operators of the edge algebra [10]: `v = org(e)` (origin vertex `v` of a half edge `e`); `f = left(e)` (face `f` to the left of a half edge `e`); `h = sym(e)` (symmetric half edge `h`); and `n = lnext(e)` (next half edge `n` on left face). These operators are trivially computed from the edge-based data structure.

The handlebody operators allow cutting and pasting surfaces. They are: f = create(v0, v1, v2) (creates a new triangular face f); destroy(f) (destroys an existing face); glue(e0, e1) ("identify" two boundary edges to make one interior edge), and unglue(e) (split one interior edge to make two boundary edges).

The stellar operators allow changing the resolution and structure of the mesh. They are: flip(e) (swaps the edge e); split(e) (bisects the edge e and its incident faces); split(f) (trisects the face f); weld(v) (inverse of the split operators). Note that flip is only defined for internal edges. In our C++ implementation, split is defined using overload of operators. weld deduces the type operation from the star of the vertex v.

5.2 Mesh Representation

Different data structures can be used to implement a mesh representation that supports the operators defined above. We have chosen an edge based topological data structure because it gives a good compromise between simplicity and generality.

In this representation, a mesh is a collection of surface components pointers.

```
struct Mesh {
  Container<Surface*> surfaces;
}
```

The surface is structured as $S = (V, E, F, B)$ where V, E, F, B are the collections of vertices, edges, faces and boundary curves respectivelly. These sets are stored in containers of pointers to the correspondent topological data structures.

```
struct Surface {
  Container<Face*>   faces;
  Container<Edge*>   edges;
  Container<Vertex*>    vertices;
  Container<Edge*>   bndries;
}
```

The boundary container simply stores pointers to one of the edges of each boundary curve and it is the topological core of our data structure. Indeed it deals effectivelly with all topological changes, i.e., the increase or decrease in the number of boundary curves. Note that a compact surface will have an empty set as its boundary.

The face structure stores a pointer to the first half-edge of its outer loop. Here, we assume triangular faces and thus, the face loop contains exactly three edges.

```
struct Face {
  Half_Edge*    he_ref;
}
```

An edge is formed by two half-edges. In the case it is representing a boundary edge one of these half-edges points to a null face.

```
struct Edge {
  Half_Edge    he[2];
}
```

The half-edge is the central topological element of the data structure. It stores a pointer to its initial vertex, a pointer to the next half-edge in the face loop, and pointers to the edge and face it belongs. Note that the mate half-edge can be accessed through the pointer to its parent edge.

```
struct Half_Edge {
  Vertex*     org_ref;
  Half_Edge*    next_ref;
  Face*   f_ref;
  Edge*   e_ref;
}
```

The vertex structure stores one pointer to an incident half-edge. In the case of a boundary vertex, this half-edge is part of the boundary curve. This representation makes it trivial to identify if a vertex is on the boundary or is in the interior of the surface. Also, it is instrumental not only for the implementation of the vertex star iterator, but also for the boundary curve iterator. The vertex structure also holds a pointer to vertex geometry.

```
struct Vertex {
  Half_Edge*    star_i;
  Point*  p;
}
```

The point data structure stores a pointer to the vertex. It represents a "bridge" between geometry and topology. It also stores geometric data of the vertex and can also hold additional data, such as normals, texture and parametric coordinates.

```
struct Point {
  Vertex*     v;
  Data*       d;
}
```

These last two data structures separate the roles of points and vertices which are to represent geometry and topology of the mesh, respectivelly. This is a robust approach to compare geometric coincidences between vertices. Surfaces reconstruction is a typical example where this is necessary. Indeed the geometric operations acts on sampple points (some may belong to the boundary or not) whereas mesh vertices attach then by handle operations whenever new triangles are created. See for example [19].

5.3 Complexity Analysis

5.3.1 Data Structure

The data structure requires 1 pointer per face, 1 pointer per boundary, 8 pointers per edge and 1 pointer per vertex (plus the additional geometric data). To estimate the mesh size in terms of the number of vertices we can use the Euler formula, $|V| - |E| + |F| = \chi(S)$. If we assume that the genus is small compared to $|V|$, then $|V| - |E| + |F| \approx 0$ and the number of boundary edges is small. Since each internal edge is shared by two faces, $|E| = 3 \cdot |F|/2$. We also know that the average valence of a vertex is 6. Therefore, the size of the data structure is $6V + \frac{3}{2} \cdot 8V + 1V \approx 18V$ pointers.

Although this data structure cannot compete in terms of size with a compressed mesh format, it compares favorably with other edge based data structures, such as the winged edge and quad edge. Alternatively, the mesh representation can be implemented using a corner table data structure instead of a half-edge data structure. This option gives a control over the usual trade-off between computational and space complexity. The half-edge data structure occupies more space because it uses pointers, but support efficient editing operations. The corner table data structure is very compact because it uses arrays, but needs realocation if the mesh is modified. An implementation of such a table-based data structure for tetrahedral meshes is described in [14].

5.3.2 Mesh operators

Using the half-edge based data structure, we can affirm that all the API operators f = create(v0, v1, v2), destroy(f), flip(e), split(e), split(f), and weld(v) have constant time complexity.

The glue(e0, e1) operator internally decides whether to use a handle operator of type 1, a handle operator of type 2 or a zip operator. This decision is done in constant time using the presented data structure, by simply counting how many vertices in common e0 and e1 have. The low level implementation of the handle operator of type 2 and the zip operator have constant time complexity. While the handle operator of type 1 in the worst case have linear time complexity in the number of edges on the boundary curves of e0 (or e1, we choose the one that has a minimum number of edges.

The unglue(e) operator internally decides whether to use an inverse handle operator of type 1 or type 2 or the unzip operator. The complexity of this decision is also done in constant time in our data structure. Given an interior edge we have count how many interior vertices are incident to it. If it has two incident interior vertices, we have to apply the inverse handle operator of type 2. In the case it has only one incident interior vertex, we have to apply the unzip operator. Both of them have constant time complexity. Otherwise, we have to apply the inverse handle operator of type 1, whose complexity is linear in the number of edges on the boundary incident to the vertices of e.

6 Applications and Examples

Mesh operators embody the fundamental transformations for combinatorial manifolds. Applications that adopt meshes as a surface representation can greatly benefit from our operators, because they provide the correct level of abstraction for algorithm design and guarantee that the representation is always valid.

In this section we discuss how our framework fits into graphics applications. Below we give examples of previous algorithms that employed some of the concepts presented in this paper. We also point out how these prototypical applications can fully exploit our mesh operators.

Among the different strategies to compress the connectivity of meshes, many of the successful approaches are based on G. Taubin and J. Rossignac's *topological surgery* [29]. The Edgebreaker scheme [25] is one example. All algorithms of this kind during the encode phase, cut the surface along a set of edges, thus they could be naturally expressed in terms of inverse handlebody operators and the unzip. As an important consequence, the handle operators together with the zip operator are the natural ones to reconstruct the surface during the decoding process. These operators prove to be very useful to design and analyze a very concise algorithm for the Edgebreaker scheme to encode and decode surfaces with genus (see [17]).

The ball-pivoting algorithm [3] reconstructs a polygonal surface from point samples. It starts with a seed triangle and grows the surface by gluing new triangles to the surface boundary. The handlebody operators allows a very robust and concise implementation of this algorithm [19].

Other important recent application where the handle operators has a very suitable use is in the construction of geometry images [9]. In such application a surface with boundary is cut along a set of edges. One can easily implement this algorithm by the use of the detach operator.

Most refinement methods for subdivision surfaces can be implemented with stellar operators. Velho [31] showed that both primal [5] and dual [7] schemes can be factorized using edge splits. $\sqrt{2}$ subdivision [33] also uses edge splits. $\sqrt{3}$ subdivision [13] employs face splits and edge flips.

Simplification methods that are based on edge collapse [8] can be implemented using edge flips and edge splits [30, 34].

Stellar operators are very suitable for creating multiresolution structures. Progressive meshes [11] and binary multi-triangulations [32] are examples of hierarchical data structures that can be built with these operators.

Normal meshes have been proposed in [4] as a representation for combinatorial man-

ifolds with a non-regular triangulation. This canonical description is constructed using stellar operators.

Welch [36] describes a system for free-form modeling of smooth surfaces. Variational and topological methods are used to design the surface shape. This is an example of a system in which *both* handlebody and stellar operators could play an equally important role. This is in contrast with the previous examples which emphasize only one class of operators.

7 Conclusions

We presented in this work an unified framework for the representation of combinatorial 2-manifolds with or without boundary. This representation includes two kinds of primitive operators on the underlying meshes: operators which change the topological characteristic of the mesh and operators which just modify its combinatorial structure. We also introduced a new data structure that explicitly represents the boundary curves. This data structure shows to be very useful for the implementation of those operators.

References

[1] J. Alexander. The combinatorial theory of complexes. *Ann. Math.*, 31:294–322, 1930.

[2] B. G. Baumgart. A polyhedron representation for computer vision. *AFIPS National Computer Conference*, (44):589–596, 1975.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, October - December 1999.

[4] S. Bischoff and L. Kobbelt. Towards robust broadcasting of geometry data. *Computers & Graphics*, 26(5):665–675, 2002.

[5] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput. Aided Design*, 10:350–365, 1978.

[6] M. Desbrun, M. Meyer, P. Schroder, and A. Barr. Discrete differential-geometry operators in nd, 2000.

[7] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Comput. Aided Design*, 10:356–360, 1978.

[8] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Computer Graphics*, 31(Annual Conference Series):209–216, 1997.

[9] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361. ACM Press, 2002.

[10] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Trans. Graph.*, 4:74–123, 1985.

[11] H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH 96*, pages 99–108, New Orleans, Louisiana, August 1996.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. Technical report, University of Washington, 1993. TR UW CSE 1993-01-01.

[13] L. Kobbelt. $\sqrt{3}$ subdivision. In *Proceedings of SIGGRAPH*, Computer Graphics Proceedings – Annual Conference Series, pages 103–112, 2000.

[14] M. Lage, T. Lewiner, H. Lopes, and L. Velho. Chf: A scalable topological data structure for tetrahedral meshes. In *Proceedings of SIBGRAPI 2005 - XVIII Brazilian Symposium on Computer Graphics and Image Processing*, Natal, October 2005. SBC - Sociedade Brasileira de Computacao, IEEE Press.

[15] W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Proceedings of the Kirbyfest*, volume 2, pages 299–320, 1999.

[16] C. Loop. Smooth subdivision for surfaces based on triangles. Master's thesis, University of Utah, 1987.

[17] H. Lopes, J. Rossignac, A. Safonova, A. Szymczak, and G. Tavares. Edgebreaker: a simple compression for surfaces with handles. In *7th ACM Siggraph Symposium on Solid Modeling and Application*, pages 289–296, 2002.

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

[19] E. Medeiros, L. Velho, and H. Lopes. A topological framework for advancing front triangulations. In *Proceedings of the XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003), to appear*. IEEE Press, October 2003.

[20] K. Mehlhorn, S. Naher, and C. Uhrig. The LEDA platform of combinatorial and geometric computing. In *Automata, Languages and Programming*, pages 7–16, 1997.

[21] J. Milnor. *Morse Theory*. Annals of Mathematics Study 51. Princeton University Press, 1963.

[22] M. H. A. Newman. On the foundations of combinatorial analysis situs. *Proc. Royal Acad.*, 29:610–641, 1926.

[23] U. Pachner. Pl homeomorphic manifolds are equivalent by elementary shellings. *Europ. J. Combinatorics*, 12:129–145, 1991.

[24] H. Poincaré. Sur la géneralisation d'un théoréme d'euler relatif aux poliédres. *C. R. Acad. Sci. Paris*, (117):437–464, 1893.

[25] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, /1999.

[26] J. Rossignac and A. Szymczak. Wrap&zip: Linear decoding of planar triangle graphs, 1999. Computational Geometry: Theory and Applications.

[27] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148, pages 203–222. 1996.

[28] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 95*, pages 351–358, August 1995.

[29] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, 1998.

[30] L. Velho. Mesh simplification using four-face clusters. In *Proceedings of SMI 2001*. Instituto per la Matematica Applicata - CNR, IEEE Computer Society, May 2001. International Conference on Shape Modeling and Applications.

[31] L. Velho. Using semi-regular 4–8 meshes for subdivision surfaces. *Journal of Graphics Tools*, 5(3):35–47, 2001.

[32] L. Velho and J. Gomes. Variable resolution 4-k meshes: Concepts and applications. *Computer Graphics forum*, 19:195–212, 2000.

[33] L. Velho and D. Zorin. 4-8 subdivision. *Computer-Aided Geometric Design*, 18(5):397–427, 2001. Special Issue on Subdivision Techniques.

[34] A. W. Vieira, L. Velho, H. Lopes, G. Tavares, and T. Lewiner. Fast stellar mesh simplification. In *Proceedings of the XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003), to appear*. IEEE Press, October 2003.

[35] K. Weiler. Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, 1985.

[36] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *Proceedings of SIGGRAPH 94*, pages 247–256, July 1994.