On converting sets of tetrahedra to combinatorial and PL manifolds

Marco Attene [a,*], Daniela Giorgi [a], Massimo Ferri [b],
Bianca Falcidieno [a]

[a] *Institute of Applied Mathematics and Information Technology - National Research Council, Genoa, Italy*

[b] *Department of Mathematics - Bologna University, Bologna, Italy*

Abstract

We investigate the problem of removing singularities from a non-manifold tetrahedral mesh so as to convert it to a more exploitable manifold representation. Given the twofold combinatorial and geometrical nature of a 3D simplicial complex, we propose two conversion algorithms that, depending on the targeted application, modify either its connectivity only or both its connectivity and its geometry. In the first case, the tetrahedral mesh is converted to a combinatorial 3-manifold, whereas in the second case it becomes a piecewise linear (PL) 3-manifold. For both the approaches, the conversion takes place while using only local modifications around the singularities. We outline sufficient conditions on the mesh to guarantee the feasibility of the approaches and we show how singularities can be both identified and removed according to the configuration of their neighborhoods. Furthermore, besides adapting and extending surface-based approaches to a specific class of full-dimensional simplicial complexes in 3D, we show that our algorithms can be implemented using a flexible data structure for manifold tetrahedral meshes which is suitable for general applications. In order to exclude pathological configurations while providing sound guarantees, the input mesh is required to be a sub-complex of a combinatorial ball; this makes it possible to assume that all the singularities are part of the mesh boundary.

Key words: tetrahedral mesh, model repair, singularity removal

[*] Corresponding author
Email addresses: `marco.attene@ge.imati.cnr.it` (Marco Attene), `daniela.giorgi@ge.imati.cnr.it` (Daniela Giorgi), `ferri@dm.unibo.it` (Massimo Ferri), `bianca.falcidieno@ge.imati.cnr.it` (Bianca Falcidieno).

1 Introduction

Digital 3D shape models are crucial in many sectors such as industrial design, gaming, simulation and medicine, to cite a few, and their impact on forthcoming multimedia-enabled systems is foreseen to grow significantly.

3D models can be either conceived using computer-aided tools, or reconstructed out of digitized real 3D objects. Recent technological advances have made available cost-effective scanning devices that could not even be imagined a decade ago: it is now possible to acquire 3D data of a physical object in few seconds and produce a digital model of its shape that can be easily shared and distributed in virtual worlds.

Although today 3D shapes can be modelled in several ways (29), the most common approach relies on the so-called *Boundary Representation* (B-Rep), which represents a solid indirectly as the volume bounded by a given, explicit surface. B-Reps are particularly appropriate for both the designer and the computer; NURBS, for example, make it possible for a designer to control the shape of smooth surface patches through few control points, while, from the point of view of the computer, triangle meshes are directly supported by the graphic hardware for rendering complex shapes at exceptional speeds.

Nevertheless, in some cases it is necessary to explicitly model also the inner parts of a shape. Fully solid shape models, for example, come directly from CT or MRI scans in medicine. In this case, the volume is typically represented in raster form as the space being digitized is subdivided into a regular grid of voxels.

In other scenarios, a B-Rep may need to be converted to a volumetric simplicial mesh (i.e. a tetrahedral mesh) in order to apply physically-based simulation techniques. In Computer Graphics, for example, realistic simulation of deformable objects are based on volume meshes (37) while, more in general, in computational sciences numerical solvers for partial differential equations need a discrete domain to apply finite-element or finite-volume methods.

The representation of simplicial meshes in a computer has been widely studied, and a number of data structures have been proposed in the literature (7; 12; 11; 15). The most efficient structures, however, can represent only manifold meshes. Moreover, in a number of applications the manifold condition of the input model is mandatory. Hence a lot of research has been devoted to the conversion of generic simplicial meshes to more efficient manifold meshes. Unfortunately, while state-of-the-art solutions to convert surface meshes are satisfactory, for higher-dimensions some problems have been encountered (16). Hence, in this article we deal with a specific class of volumetric simplicial meshes and propose two solutions for their conversion to combinatorial and

PL manifolds. In order to exclude pathological configurations while providing sound theoretical guarantees, the input simplicial mesh is required to be a sub-complex of a combinatorial ball, which implies that all the singularities can be assumed to be on the boundary.

1.1 Motivation

In medical applications, CT or MRI scans of a patient generate raster 3D images in which the brightness of each voxel is related to the type of tissue sampled in that position. In several cases it is important to *extract* from the 3D image the shape of a particular tissue (e.g. an organ or a bone). This procedure, known as *segmentation*, may generate volumetric simplicial meshes (6; 23). The suitability of these meshes for specific kinds of analysis often requires them to be manifold, as it happens for example in the context of surgical simulation (17).

Similarly, recent variational meshing techniques (1) allow one to easily convert a surface mesh to a volume mesh in which the placement of vertices in inner parts guarantees both a smooth transition in sampling density and well-shaped tetrahedra, which are fundamental conditions for an effective application of the finite-element method. Nevertheless, such volumetric models are not guaranteed to be manifold (see Figure 1). Although the manifold condition is not mandatory for the finite-element method, non-manifold meshes may lead to ill-conditioned equation systems, and instabilities may be observed around singularities.

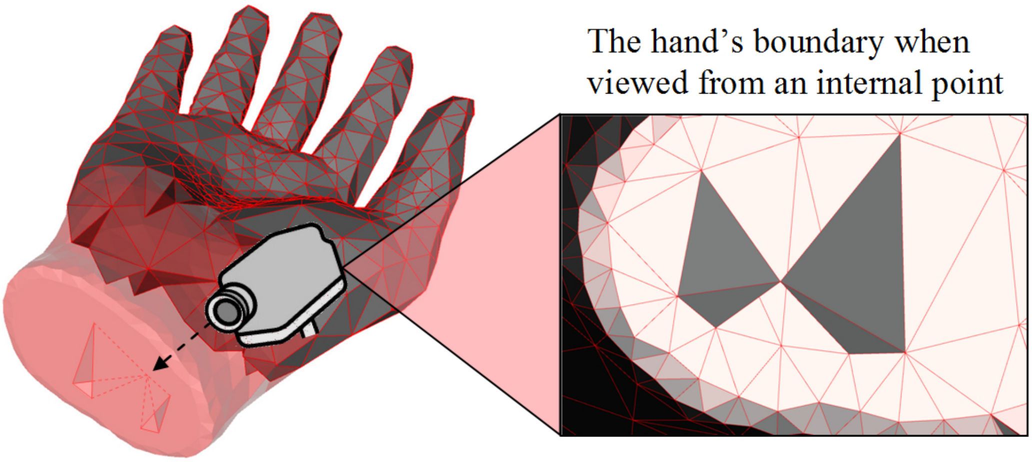

Fig. 1. A tetrahedral mesh obtained through the variational meshing algorithm described in (1) in which a singular vertex is shown. Model courtesy of Pierre Alliez.

Working with non-manifold meshes is also a limitation in several other applications where efficiency is mandatory. On a data structure designed for manifold meshes specifically, in fact, traversal operations are much faster, and this is fundamental to implement efficient algorithms to detect collisions, to perform boolean or morphing operations, to simplify the model, or even simply to render it efficiently.

Note that simplicial 3-complexes have a twofold nature. Indeed they can be seen as either purely combinatorial objects (i.e. abstract simplicial complexes), or as subsets of the Euclidean 3D space (namely *polyhedra*) through the notion of *geometric realization* (see Section 3). Most data structures neatly separate the connectivity of the mesh from its geometry, the latter defining the position of each vertex in the 3D space. For some applications a manifold connectivity may be sufficient, whereas for some others the polyhedron itself is required to be a valid 3-manifold. The former condition does not imply the latter: a polyhedron does not necessarily change as the connectivity is modified and, even if the complex becomes a combinatorial manifold, no guarantees are given on the validity of its geometric realization.

As far as efficiency is sought, it can be granted by the manifold condition on the abstract complex defining the connectivity, because typical traversal operations are independent of the geometry.

In other cases, having combinatorial complexes that define manifold polyhedra is desirable. In surgical simulation, for example, each point of the polyhedron's surface is a candidate for a virtual cut, and existing systems assume that each such point has a well-defined *thick* neighborhood that can be split to simulate the cut itself (17).

Summarizing, it is important to develop both efficient approaches to convert the connectivity of a volumetric simplicial mesh into a manifold abstract complex and algorithms to derive polyhedra that are manifold in the Euclidean space.

1.2 Overview and contributions

In this article two algorithms are presented: (1) a purely combinatorial method to process the connectivity of a tetrahedral mesh so as to make it manifold, and (2) a geometric approach to edit the neighborhood of singularities so that the resulting complex defines a manifold polyhedron. A preliminary version of the combinatorial approach was previously introduced in a conference paper (5), whereas the geometric method along with an extended notation and a deeper discussion are novel contributions of this article. A further additional contribution within the context of the geometric approach is the strategy employed to encode the *history* of the converted shape through an XML wrapper.

Existing related work is described in section 2, where both the most important state-of-the-art algorithms used to process B-reps and the attempts done so far to treat higher dimensional complexes are presented. The mathematical background is then presented in section 3 and, based on this, the problem tackled in the article is stated formally. The combinatorial approach is then described in section 4 both at a conceptual level using a mathematical terminology, and at a practical level using a more computer-driven language and pseudo-code snippets. Similarly, the geometric methodology is introduced in section 5 at the same two levels of abstraction. In section 6 we discuss some of the choices made in the design of the algorithms and outline possible alternative approaches. Finally, we draw our conclusions and the directions of future developments in section 7

2 Related Work

In geometric modeling, *model repair* is the task of removing artifacts from a geometric model to produce an output model that is suitable for further processing by applications that have specific input requirements (9). The conversion to manifold presented in this article can be seen as a specific subtask of model repairing.

Representations for non-manifold objects exist and are useful in several contexts (22; 25; 28). Also, some modelling frameworks allow the representation of manifolds of mixed dimensionalities (33; 31). Nevertheless, the need for efficient algorithms has called for the development of data structures specifically dedicated to the manifold case. Manifold simplicial complexes can be represented in any dimension (10; 30), although most data structures have been proposed specifically for the 2D and 3D cases (13; 21; 27; 29; 11).

In several practical applications, it happens that the simplicial complex resulting from a specific process (e.g. segmentation of a 3D image or digitization of a real object) is mostly manifold, in the sense that only a small percentage of vertices are singular. This fact prompted the development of several algorithms that slightly modify the complex to edit the singularities without changing anything far from them. To achieve this goal, a widely used approach consists of decomposing non-manifold complexes into simpler parts, splitting at those elements (vertices, edges, facets, etc.) where singularities occur (14; 20; 32). The result of such a decomposition is a collection of singularity-free components that can be represented by standard data structures for manifold complexes. Note that in these methods the cuts are local, which means that the topology of the mesh changes only in a neighborhood of the edited singularity; thus splitting at a singular element does not necessarily subdivide the mesh in topologically disjoint components. Most of these works have been proposed for the case of surface meshes, as detailed in the following section.

2.1 Manifold surface meshes

In (32) a method is proposed to convert a non-manifold set of triangles to a set of manifold surface meshes. First, non-manifold edges (i.e. edges having more than two incident faces) are identified, and each edge having $2k$ incident faces is split into k manifold edges, so that if these edges are bent by a small amount in the appropriate direction, the resulting shape will not have self-intersections. This representation is called an *edge-manifold* representation, and may still contain isolated non manifold vertices. Then, to guarantee a manifold topology, one has to identify and duplicate properly the non manifold vertices. A strategy is suggested to produce a minimum number of vertex duplications (32).

In a similar setting, (20) introduces a strategy based on two high level operations: cutting and stitching. The *cutting* operation involves identifying non-manifold edges and cutting the surface along such edges. Two strategies are available for cutting: a global method, operating on all the surface elements, which is appropriate for cuts covering a large portion of the surface, and a local strategy, operating only on a set of marked vertices and edges, which is more efficient in case of a small number of marked elements. The result of the cutting operation is a manifold surface that may contain boundary edges. Hence, a *stitching* operation is performed, which involves joining two boundary edges while guaranteeing that the surface has a manifold topology. There are two greedy strategies for stitching: *pinching* attempts to simply zip boundary edges created during the cutting operation, while *snapping* attempts to stitch along boundaries other than those, and reduces the number of connected components of the surface. Differently from (32), the method in (20) does not address geometric issues such as self-intersecting surfaces.

2.2 Manifold complexes in higher dimensions

In (16), it has been pointed out that the decomposition of a non-manifold complex should not introduce artificial or arbitrary *modifications* to manifold parts. Under these assumptions, a decomposition into manifold components is possible, in general, only for two-dimensional complexes. Therefore, instead of trying to obtain a set of manifold components, the solution proposed in (16) converts a general non-manifold complex to a set of so-called *initial quasi-manifolds*. While in 2 dimensions an initial quasi-manifold is also a 2-manifold (and vice-versa), in higher dimensions this equivalnce does not hold, and models containing singular vertices (Figure 2) may satisfy all the conditions to be initial quasi-manifolds without being actual manifolds. Thus, to efficiently work with initial quasi manifolds, proper data structures are required in which the presence of such singularities do not represent a drawback; in (24) such a data structure has been proposed for the specific 3D case. Though initial quasi-manifolds have several interesting characteristics, and though they can be represented through proper data structures, the presence of singularities make them still not satisfactory for a number of applications.

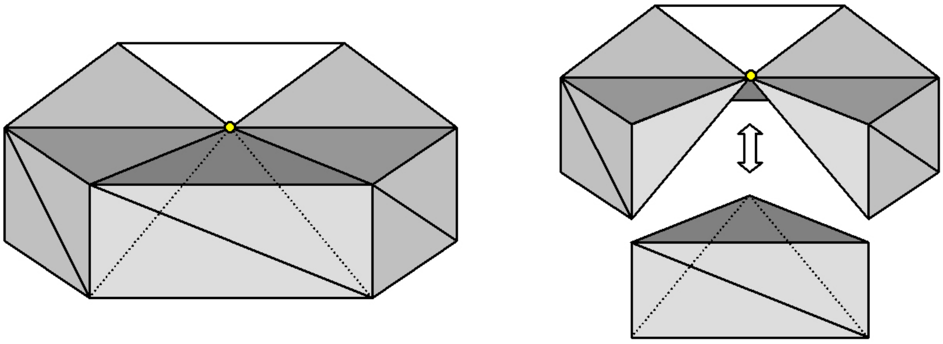

Fig. 2. An example singular vertex in a 3D initial quasi-manifold. On the right, a pair of adjacent tets have been detached to better show the singularity. In (16), this configuration is called a *pinched pie*.

3 Background definitions

3.1 Simplicial complexes

A *k-dimensional simplex*, or *k-simplex*, A^k is a set $V = \{v_0, \ldots, v_k\}$ of $k+1$ objects called *vertices*, together with the set of real-valued functions $\alpha : V \rightarrow \mathbb{R}$ satisfying $\sum_{v_i \in V} \alpha(v_i) = 1$ and $\alpha(v_i) \geq 0$. A function α is called a point of A^k. The values $\alpha(v_0), \ldots, \alpha(v_k)$ are the *barycentric coordinates* of the point α (18).

A (proper) *face* B of A^k, denoted $B < A^k$, is a simplex determined by the (proper) subset $W \subset V$, whose points $\beta : W \rightarrow \mathbb{R}$ are identified with the points $\alpha : V \rightarrow \mathbb{R}$ such that $\alpha(v_i) = \beta(v_i)$ if $v_i \in W$ and $\alpha(v_j) = 0$ if $v_j \in V - W$. If B is a face of A, then A is said to be *incident* at B.

A *finite simplicial complex* K is a finite set of simplices such that:

i) if $A \in K$ and $B < A$, then $B \in K$;
ii) if A, $B \in K$, then $A \cap B$ is either empty or it is a face of both A and B.

From now on, we shall omit the term finite.

The *dimension* of K is the dimension of the largest dimensional simplex belonging to K. A simplicial complex of dimension n is *homogeneous* if it is made of n-simplices and their faces.

The *boundary* ∂A of a simplex A is the complex made of the proper faces of A. The *boundary* ∂K of a homogeneous n-dimensional simplicial complex K is the $(n-1)$-complex obtained as the sum mod 2 of the $(n-1)$-dimensional simplices of the boundary ∂A of each of the n-simplices $A \in K$ plus their faces (18).

L is a *subcomplex* of K if L is a complex and $L \subset K$. For $A \in K$, the (closed) *star* of A in K, $star(A, K)$, is the subcomplex of K made of all simplices of K having A as a face plus all their faces. If $A \in K$, then the *link* of A in K, $link(A, K)$, is the set of simplices in $star(A, K)$ whose intersection with A is empty.

A *geometric realization* $|A^k|$ of a simplex A^k in the Euclidean space $\mathbb{R}^n$, $n \geq k$, can be obtained by defining a bijection between the vertices of A^k and a set of $k+1$ affinely independent points $p_0, p_1, \ldots, p_k$ of $\mathbb{R}^n$, so that $|A^k| = \{(t_0p_0 + t_1p_1 + \ldots + t_kp_k) \in \mathbb{R}^n \mid t_i \geq 0, \sum_i t_i = 1\}$. Thus, $|A^k|$ is the convex hull of $p_0, \ldots, p_k$. In particular, the *standard k-simplex* Δ^k is defined as the convex hull of the points $e_0 = (1, 0, \ldots, 0)$, $e_1 = (0, 1, 0, \ldots, 0)$, $\ldots$, $e_k = (0, 0, \ldots, 1) \in \mathbb{R}^{k+1}$.

The *underlying space* $|K|$ of K is the union $\cup_{A \in K} |A|$ of the geometric realization of its simplices.

A complex L is a *subdivision* of the complex K if $|L| = |K|$ and every simplex of L lies in a simplex of K. In particular, we denote $edgesplit_K(e, \overline{v})$ a subdivision of K in which the edge $e = (v_1, v_2)$ is replaced with two edges $e_1 = (v_1, \overline{v})$ and $e_2 = (\overline{v}, v_2)$. We say that $K' = edgesplit_K(e, \overline{v})$ is obtained from K through an *edge-split subdivision* (see Figure 3).

In this paper we deal with 3-dimensional complexes, and performing an edge-split subdivision on such objects is equivalent to replacing an edge e with two sub-edges e_1, e_2 and, consequently, replacing each tet incident at e with two sub-tets, as shown in the example of Figure 5; calling f the edge opposite to e in the original tet, these two sub-tets are given by the join of f with e_1 and e_2. i.e. the convex hulls of $f \cup e_1$ and $f \cup e_2$. In the remainder of the paper we will simply say that we split an edge of a 3-dimensional complex

If L is a subcomplex of a complex K, the *simplicial neighbourhood* of L in K is $N(L, K) = \{A \mid A \in K, A < B, B \cap |L| \neq \emptyset\}$. In other words, $N(L, K)$ is the smallest subcomplex of K whose realization is also a neighbourhood of $|L|$ in the Euclidean space (34). Hence, the simplicial neighborhood of a vertex coincides with its star, whereas the star of a higher-dimensional simplex is a

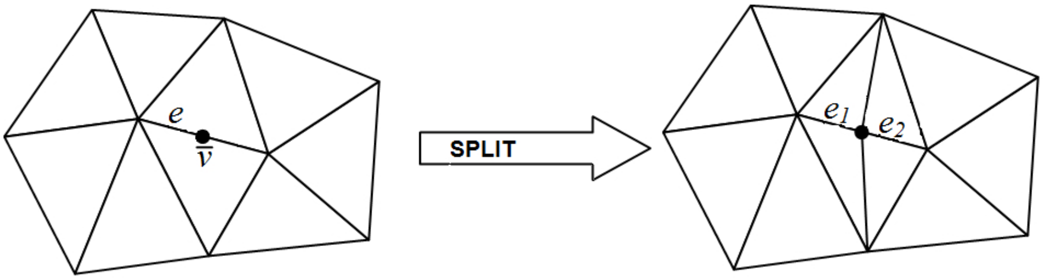

Fig. 3. Example of an edge-split subdivision of a 2-dimensional complex.

proper sub-complex of its simplicial neighborhood. The simplicial neighborhood of a subcomplex L turns out to be the subcomplex, union of the stars of the vertices of L. Though this paper deals with 3-dimensional complexes, simplicial neighborhoods made of tets are difficult to visualize, thus in Figure 4 we show an example simplicial neighbourhood of a 1-simplex in a 2-dimensional complex.

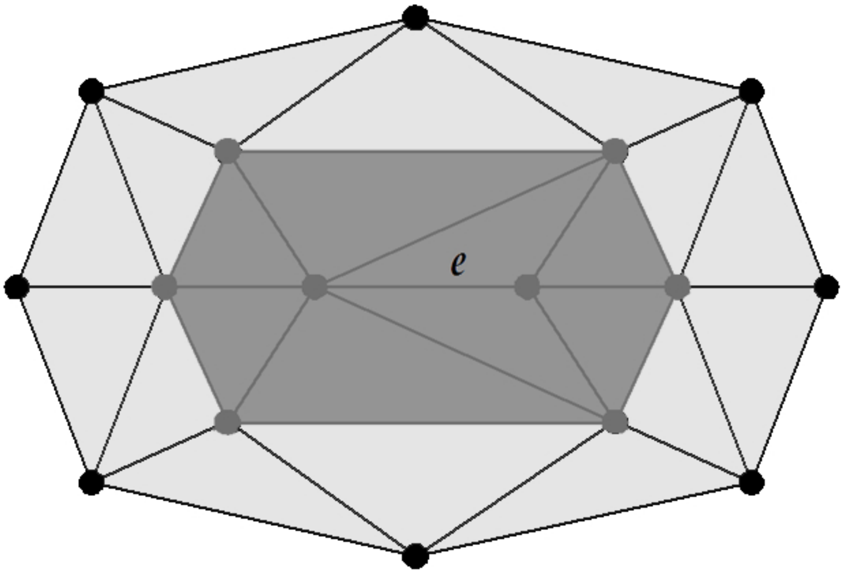

Fig. 4. Simplicial neighbourhood of a 1-simplex e in a 2-dimensional complex.

3.2 Piecewise linear homeomorphisms

Let K and L be simplicial complexes whose sets of vertices are denoted $V(K)$ and $V(L)$, respectively, and let $f : V(K) \to V(L)$ be a bijection such that the vertices $v_0, \ldots, v_p$ of K span a simplex of K if and only if $f(v_0), \ldots, f(v_p)$ span a simplex of L. Then, f is said to be a *simplicial isomorphism* and the induced map $|f| : |K| \to |L|$, taking $v = \sum_i t_i v_i$ to $g(v) = \sum_i t_i f(v_i)$, is called a *piecewise linear homeomorphism.*

Given two simplicial complexes K_1 and K_2, a map $g : |K_1| \to |K_2|$ is a piecewise linear homeomorphism if and only if there exist subdivisions L_1 of K_1 and L_2 of K_2 and a simplicial isomorphism $f : L_1 \to L_2$ such that $g = |f|$. $|K_1|$ and $|K_2|$ are said to be *piecewise linear homeomorphic* if there exists a piecewise linear homeomorphism between them. For the sake of simplicity we shall confuse, with a fairly usual abuse of language, every K with $|K|$, f with $|f|$, and the notion of simplicial isomorphism with that of piecewise linear homeomorphism.

A *combinatorial n-ball* is a complex piecewise linearly homeomorphic with the standard simplex Δ^n. A *combinatorial n-sphere* is a complex piecewise linearly homeomorphic with the boundary $\partial\Delta^{n+1}$ of the standard simplex Δ^{n+1}.

3.3 Combinatorial and PL n-manifolds

A *combinatorial n-manifold* is a homogeneous n-dimensional complex K such that for any vertex v of K, $link(v, K)$ is a combinatorial $(n-1)$-ball if $v \in \partial K$ and a combinatorial $(n-1)$-sphere if $v \notin \partial K$.

A simplex $A^p \in K$ is *regular* in K, where K is a homogeneous n-dimensional complex, if $link(A^p, K)$ is a combinatorial $(n-p-1)$-ball if $A^p \in \partial K$ and a combinatorial $(n-p-1)$-sphere if $A^p \notin \partial K$; otherwise A^p is called a *singular* simplex. It follows that a combinatorial n-manifold is a homogeneous complex in which every vertex is regular. It also holds that in a combinatorial n-manifold all simplices are regular (18).

A *piecewise linear (PL) n-manifold* is a subset of the Euclidean space, which is the underlying space $|K|$ of a combinatorial n-manifold K.

4 Building combinatorial 3-manifolds

In this article, we deal with three-dimensional simplicial complexes. In this case, special names are given to simplices depending on their dimension. Specifically, a 3-simplex is called a *tetrahedron*, or simply *tet*, a 2-simplex is a $facet$, a 1-simplex is an *edge* and a 0-simplex is a *vertex*. We adopt the term *tetrahedrization* to indicate a complex made of tetrahedra and their faces. Recalling the definitions in section 3, a tetrahedrization K is a combinatorial 3-manifold if, for any vertex v of K, $link(v, K)$ is a combinatorial 2-ball if $v \in \partial K$ and a combinatorial 2-sphere if $v \notin \partial K$.

4.1 Problem statement

Le K be a homogeneous, possibly non manifold sub-complex of a combinatorial 3-ball; without loss of generality, we assume that K has no vertices on the boundary of the ball. Our aim is to locally edit K so as to transform it in a manifold complex. Editing operations must influence the neighborhood of the singularities exclusively, while the remaining parts of the complex must remain unmodified. More in detail, supposing, without loss of generality, that K has a single singular vertex (or edge) v, we shall obtain a new complex K' that will be a combinatorial 3-manifold coincident with K everywhere but $star(v, K)$. Note that, differently from (16), we allow the modification of some manifold faces which are incident at singular elements. Specifically, while in (16) only the link of singular elements is modified, our approach is allowed to modify also the link of some faces of singular elements.

Since the aim of this section is to alter the connectivity of the complex, we will limit ourselves to showing a pseudo-realization of K' in $|K|$, that is we will indicate a non-injective simplicial map from K' to K.

4.2 Approach

Roughly speaking, our approach consists of two phases: first we identify and treat singular edges; in a second step, we deal with the remaining vertex singularities adopting two different procedures that depend on the configuration of the link. Note that this is the same sequence employed in (32) to process surface meshes. Also, notice that our assumption that K is a subcomplex of a combinatorial ball, makes it impossible to have singular triangles.

More formally, let L be a combinatorial 3-ball, and K a homogeneous subcomplex of L with no vertices in ∂L. The procedure we adopt to remove singularities located at vertices requires that the link of each vertex has no singularities. To achieve this condition, we treat singular edges first as follows (refer to Figure 5 for an example showing a step of the procedure).

Let $e = \{v_1, v_2\}$ be a singular edge in K. Then $link(e, K)$ is a simplicial 1-complex made of $k > 1$ components $L_i(e)$, with $i = 1, \ldots, k$. We create k new vertices w_i, with $i = 1, \ldots, k$, each positioned at the midpoint of e. For each such vertex, we also create two new edges $e_1^i = \{v_1, w_i\}$ and $e_2^i = \{w_i, v_2\}$. Now, for each edge $l^{i,j} = \{u_1^{i,j}, u_2^{i,j}\}$ in $L_i(e)$, with $i = 1, \ldots, k$, we consider the tet $t^{i,j}$ having both $l^{i,j}$ and e as faces, and replace it with two new tets $t_1^{i,j} = \{v_1, w_i, u_1^{i,j}, u_2^{i,j}\}$ and $t_2^{i,j} = \{w_i, v_2, u_1^{i,j}, u_2^{i,j}\}$. Note that these operations are similar in spirit to what is done in (32) to treat edges; roughly speaking, the process is equivalent to (1) replacing e with k new edges sharing the same end-points and (2) splitting each such new edge at its midpoint. After these operations, the original singular edge e has no remaining incident tets and is removed from the complex.

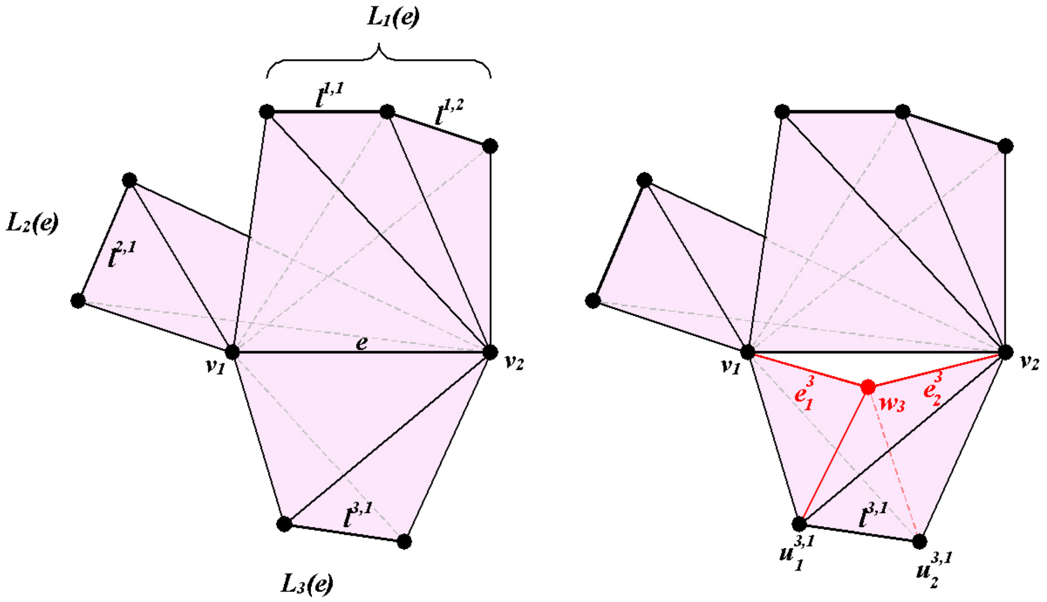

Fig. 5. An example of singular edge e in which one of the components ($L_3(e)$) of its link is considered for the creation of a new vertex w_3 and its incident elements. The algorithm places the new vertex w_3 at the midpoint of the singular edge e; in the figure, however, it is depicted in a displaced position to better illustrate the resulting connectivity.

In the new complex obtained by applying this procedure to all the singular edges, that we still call K, all the newly created w_is are manifold and all the singular vertices have a manifold link.

Each remaining singular vertex v in K can then be treated as follows.

By hypothesis, $link(v, L)$ is a combinatorial 2-sphere while $link(v, K)$ is neither a combinatorial 2-sphere, nor a combinatorial 2-ball. Since $link(v, K)$ is a sub-complex of $link(v, L)$, $link(v, K)$ is the disjoint union of $n \geq 1$ combinatorial 2-spheres with holes. The boundary $\partial link(v, K)$ of $link(v, K)$ is made up of $k > 1$ components, each of which is a combinatorial 1-sphere.

We proceed recursively with respect to k.

Among the combinatorial 1-spheres bounding $link(v, K)$, at least one of them, let it be C, bounds a 2-ball D of either $link(v, K)$ or $link(v, L - K)$ (note that in our setting the Schoenflies conjecture holds (19)).

We add a new vertex w in K at the same position as v, and distinguish two possible cases:

i. If $D \subset link(v, K)$, we replace each tet t_i incident at v and having a face in D with a tet τ_i obtained from t_i by substituting v with w (Figure 6).

ii. If $D \subset link(v, L - K)$, we consider the facets f_i incident at v and having an edge in C and, for each of them, we add a new tet τ_i made of w and the vertices of f_i (Figure 7).

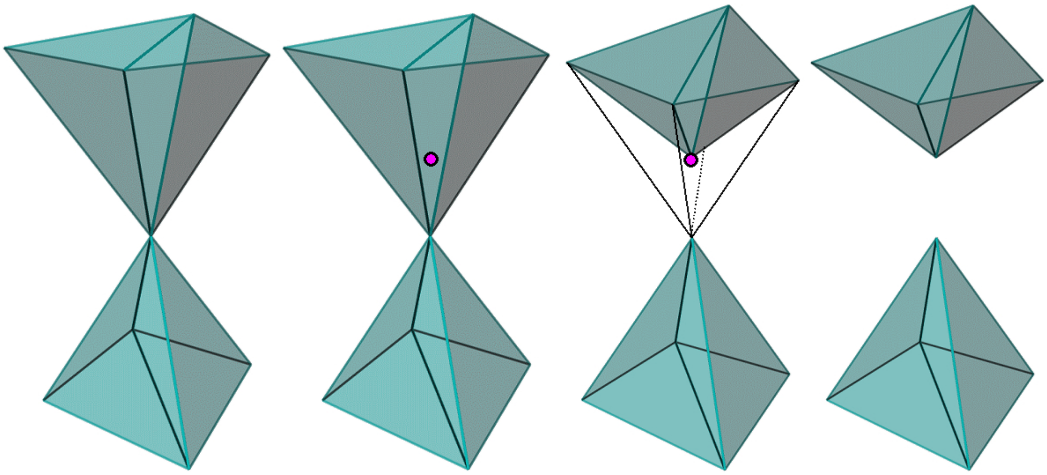

Fig. 6. Vertex duplication and retriangulation of a configuration having two connected components in the link of an isolated singular vertex. The algorithm places the new vertex at the same position of the former singularity; in the figure, however, it is depicted in a displaced position to better illustrate the resulting connectivity.

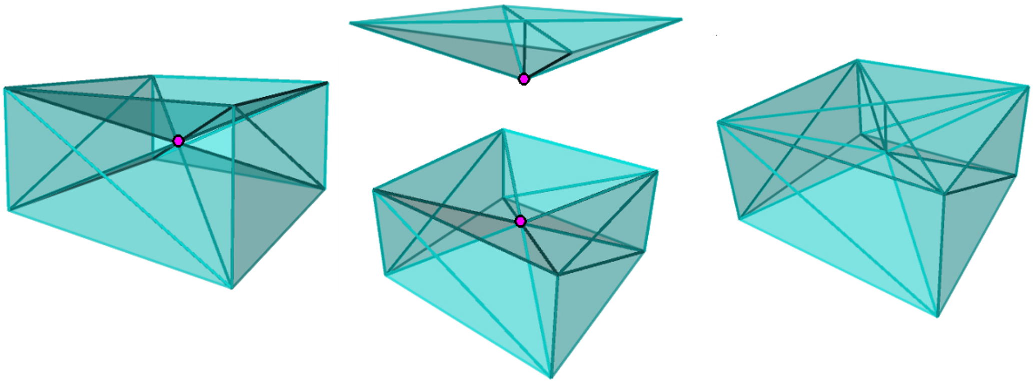

Fig. 7. Conversion of a *pinched pie* configuration to a combinatorial 3-ball. The algorithm places the new vertex at the same position of the former singularity; in the figure, however, it is depicted in a displaced position to better illustrate the resulting connectivity.

Now, the new vertex w is clearly manifold, while the boundary $\partial link(v, K)$ is made of $k - 1$ components, each of which is a combinatorial 1-sphere. Hence, we repeat the procedure as long as the number of components in $\partial link(v, K)$ is greater than 1.

In this way, we obtain a new complex K' in which all simplices are regular, that is a combinatorial 3-manifold.

Notice that, for simplicity, the new vertices w are assigned the same coordinates of the singular vertex v in the Euclidean space. This implies that our method does not produce a geometric realization of K in the Euclidean space, but just a pseudo-realization. Actually, what happens is that a map $f : V(K') \rightarrow V(K)$ induces a simplicial map which is not injective. The complex resulting after our combinatorial algorithm has the characteristic that it is geometrically identical to the input when pseudo-realized within the Euclidean space. This implies that the various copies of each singularity occupy the same region of space: if the input is non self-intersecting, these are the only self-intersections that can occur in the output. The issue of editing the complex so as to obtain a valid, manifold geometric realization (i.e. a PL manifold) is dealt with in Section 5.

4.3 Algorithm

We have implemented the algorithm based on the *tetrahedral data structure* introduced in (11). Within such a structure, the four basic entities of a tetrahedrization (i.e. vertices, edges, facets and tets) are encoded, while only a minimal set of topological relations are stored explicitly. Topological relations describe the connectivity of the complex by linking each simplex to its *boundary* and *co-boundary*, that is, the set of its faces and the set of its incident simplices respectively.

For compactness, the tetrahedral data structure explicitly encodes only the following four *constant* relations, each of which links an element with a constant number of neighboring entities:

- **tet-facet** (TF) which returns the four facets bounding a tetrahedron;
- **facet-tet** (FT) which returns the tets (one or two) incident at a facet;
- **facet-edge** (FE) which returns the three edges bounding a facet;
- **edge-vertex** (EV) which returns the two vertices of an edge;

and the following two special relations, each of which links an element with only one of the neighboring entities:

- **edge-facet** (EF*) which returns one of the facets incident at an edge;
- **vertex-edge** (VE*) which returns one of the edges incident at a vertex.

Provided that the tetrahedrization encoded is combinatorially manifold, from this minimal set it is possible to compute all the other topological relations in optimal time (i.e. in a number of operations linearly proportional to the number of elements involved in the relation being computed). We point the reader to (11) for a detailed description of the algorithms that compute these *implicit* relations.

Note that, if the tetrahedrization T is a sub-complex of a combinatorial ball, each facet can have at least one, and at most two incident tets; in this case, even if T is not a combinatorial manifold, it can be encoded by the tetrahedral data structure and all the constant relations can be still computed in optimal time.

The presence of singularities, however, does not allow to extract all the other relations in optimal time. Therefore, to keep the complexity of the conversion algorithm linear, we pre-compute the link of all the vertices and edges as follows.

Each vertex v is assigned an additional relation *L(v)*, initially empty, that encodes its link. Note that, since such a link is a homogeneous simplicial 2-complex, L needs to explicitly store only the facets; all the other lower-dimensional elements of the link are faces of such facets and can be extracted through the FE and EV relations. Then, for each tet t in the tetrahedrization we consider its four facets, and add each such facet to the link L of its opposite vertex in t. This procedure is sketched in the pseudo-code of Listing 1, where the constant relation FV returning the vertices of a facet is obtained as the combination $FV(f) = EV(FE(f))$, and the constant relation TV returning the vertices of a tet is obtained as the combination $TV(t) = FV(TF(t))$.

The star of v can be computed starting from *L(v)* in optimal time through the FT relation.

Listing 1. Construction of the links of all the vertices

```
CreateVertexLinks(Tetrahedrization $T$)
{
  for each vertex $v$ in $T$
   $L(v)$ := new empty facet list

  for each tet $t$ in $T$
   for each facet $f$ in $TF(t)$
   {
    $v$ := the vertex of $TV(t)$ which is not in $FV(f)$
    Add $f$ to $L(v)$
   }
}
```

Similarly, each edge e is assigned an additional relation *L(e)* that encodes the edges constituting its link, which is a homogeneous simplicial 1-complex. Thus, for each tet t in the tetrahedrization we consider its six edges, and add each such edge to the link *L(e)* of its opposite edge e in t.

Unfortunately, the star of e cannot be computed starting from *L(e)* in optimal time. Hence, during the computation of *L(e)*, we also fill a further relation *T(e)* with the tets t used to build *L(e)* (i.e. the tets incedent at e that, together with their faces, constitute the star of e).

The procedure to compute the links and the stars of all the edges is sketched in Listing 2, where the relation TE is computed as the composition of FE and TF.

Listing 2. Construction of the links and the stars of all the edges

```
CreateEdgeLinksAndStars(Tetrahedrization $T$)
{
  for each edge $e$ in $T$
  {
   $L(e)$ := new empty edge list()
   $T(e)$ := new empty tet list()
  }

  for each tet $t$ in $T$
   for each edge $e$ in $TE(t)$
   {
    $e_2$ := the edge in $TE(t)$ sharing no vertex with $e$
    Add $e$ to $L(e_2)$ and $t$ to $T(e_2)$
   }
}
```

Afterwards, the link *L(e)* of each edge is analyzed, and if it is not connected e is declared to be a singular edge.

In this case, one component of the link remains as it is, while for each additional component $L_i(e)$ of *L(e)* a copy e_i of e is created, and the following topological relations are updated in the data structure:

- EV(e_i) := EV(e)
- for each tet t in $T(e)$ having an edge in $L_i(e)$, and for each facet f of t such that FE(f) contains e, replace e with e_i in FE(f). Take one of these facets as the value of EF*(e_i)

Finally, e and the e_is are split at their midpoints to eliminate duplicated edges in the structure, and the L and T structures are updated accordingly for all the elements involved in the modification.

At this stage, we are guaranteed that the link of each vertex is a combinatorial 2-manifold with some boundaries and possibly more than one connected component.

In particular, if the link *L(v)* of v has more than one boundary component, then v is declared to be singular. In this case, the algorithm proceeds as described in Listing 3.

Once such algorithm terminates, the data structure represents a combinatorial 3-manifold, thus the L and T relations can be deleted, and all the topological relations can now be extracted in optimal time.

Since the tetrahedral data structure is suitable for a wide spectrum of applications, it is worth to implement the conversion algorithm directly on such a structure. Note that the use of this structure has no effects on the overall complexity of the algorithm; all the operations needed to perform the conversion, in fact, can be executed in optimal time.

Listing 3. Conversion algorithm

```
EditSingularVertex(Vertex $v$)
{
 1.  Look for a connected component in $L(v)$ which is a combinatorial 2-ball
 2.  if there is such a component, let it be $D$
 3.    Compute the set $T$ of tets incident at $v$ and having a facet in $D$
 4.    Create a new vertex $w$
 5.    Remove all the tets in $T$ from the tetrahedrization
 6.    Create a new tet {$w$,$w_1^i$,$w_2^i$,$w_3^i$} for each facet {$w_1^i$,$w_2^i$,$w_3^i$} in $D$
 7.    Remove $D$ from $L(v)$
 8.    If $L(v)$ has more than one component go to 1, else go to 15
 9.  else (i.e. if there are no combinatorial 2-balls in $L(v)$)
 10.   Extract a boundary loop $B$ of $L(v)$
 11.   Compute the set $D$ of facets incident at $v$ and having an edge in $B$
 12.   Create a new vertex $w$
 13.   Create a new tet {$w$,$w_1^i$,$w_2^i$,$v$} for each facet {$w_1^i$,$w_2^i$,$v$} in $D$
 14.   Add all the facets {$w$,$w_1^i$,$w_2^i$} to $L(v)$
 15.   If $L(v)$ has more than one boundary component go to 10, else terminate
}
```

5 Building simplicial 3-manifolds

Editing the simplicial complex, as in Section 4.2, to obtain a combinatorial manifold is useful in many contexts in which only a manifold connectivity is required. Nonetheless, there are cases in which having vertices at the same position is a problem, as the underlying space (or realization) of the complex is required to be a valid 3-manifold within the Euclidean topology. In other words, the complex is required to be a PL manifold.

A very simple approach to achieve such a requirement would be the "displacement" of those new vertices that have been inserted in the complex while making it a combinatorial manifold. This approach would require to move the newly inserted vertices to new positions, so that the underlying space does not have non-manifold points (also called singularities). Unfortunately, such a strategy requires to perform several consistency checks because the new position of a vertex might cause the realized complex to self-intersect or to contain degenerate or inverted tetrahedra. While checking for self-intersections in a topological neighborhood is relatively easy, it turns out to be computationally expensive for parts of the complex which are combinatorially far but potentially close when realized in the Euclidean space. The problem can be tackled more efficiently if the complex's complementary part is available, but unfortunately this is not the case for most practical scenarios.

Thus, our solution is based on a completely different approach in which the "geometric" nature of the complex is considered from the very beginning.

5.1 Problem statement

As with the combinatorial approach of section 4, let K be a homogeneous, possibly non manifold sub-complex of a combinatorial 3-ball L. Having assumed that L can be geometrically realized, we want to devise a strategy to locally edit K so as to obtain a PL manifold. In other words, we seek for a combinatorial 3-manifold K', with a valid, non self-intersecting and manifold underlying space $|K'|$. As an additional requirement, editing operations must modify only simplices which are incident at singular elements and must not introduce additional singularities to be processed.

5.2 Approach

Roughly speaking, our approach aims at removing small amounts of material around singular points, so as to resolve the singularities while keeping changes local.

A singular point p in $|K|$ may be either the image (through the map defining the geometric realization of simplices in K) of a vertex $v \in K$, or a point within the image of an edge $e \in K$. Let $\sigma \in K$ denote this pre-image simplex, whose dimension can be either 0 or 1.

If $p \in |K|$ is singular, we want to remove from $|K|$ all the points of a small neighborhood of p. Defining the shape and size of the neighborhood to be removed is not trivial due to the requirements imposed so far. Indeed, we want the editing operations to modify only simplices in K that are incident at the pre-image σ. Removing plain Euclidean neighborhoods such as balls, for example, is not appropriate. The removal of a ball $B(p, \epsilon)$ around the singular point p, indeed, requires the choice of a radius $\epsilon > 0$ that guarantees that all the simplices whose realization intersects B are also incident at σ. Thus, to verify that ϵ is small enough, also simplices which are combinatorially far from σ must be checked for a potential intersection, and this is computationally expensive.

Hence, our solution is to remove from $|K|$ only points around p whose pre-image belongs to the simplicial neighbourhood $N(\sigma, K)$ of the simplex σ in K. Being the simplicial neighbourhood of σ in K the smallest subcomplex of K whose realization is also a neighbourhood of $|\sigma|$ in the Euclidean space (see section 3), this guarantees that the removal is local in both combinatorial and geometric senses.

Note that the complete removal of a tetrahedron is not allowed as it might introduce new singularities in the complex, thus we have to guarantee that at least a portion of each realized tetrahedron remains in $|K|$ after the neighbourhood removal. To achieve this, we perform a subdivision of the the simplicial neighbourhood $N(\sigma, K)$ based on edge-splits, as detailed in the following section.

5.2.1 Editing Operations

Let p be a singular point whose pre-image is the simplex $\sigma \in K$ (a singular vertex or a singular edge), and let $E(\sigma)$ be the set $\{e_1, ..., e_n\}$ of all the edges which are incident or adjacent to σ. Denote with p_i the point of $|e_i|$ having distance from $|\sigma|$ equal to ϵ (see the example in Figure 9 where σ is the edge e), with $0 < \epsilon < \epsilon_{max}(\sigma)/2$. The threshold $\epsilon_{max}(\sigma)$ is the minimum distance of $|\sigma|$ from the boundary of $N(\sigma, K)$. Then, let K' be the complex obtained by splitting each e_i at p_i. By definition, K' is a subdivision of K, and $N(\sigma, K')$ is a subcomplex of K'. Thus we can simply remove $N(\sigma, K')$ from K' to get rid of the singular simplex σ while respecting the editing constraints. We call *erosion* of σ the sequence of operations including the subdivision of K into K' and the subsequent removal of $N(\sigma, K')$ (Figure 8).

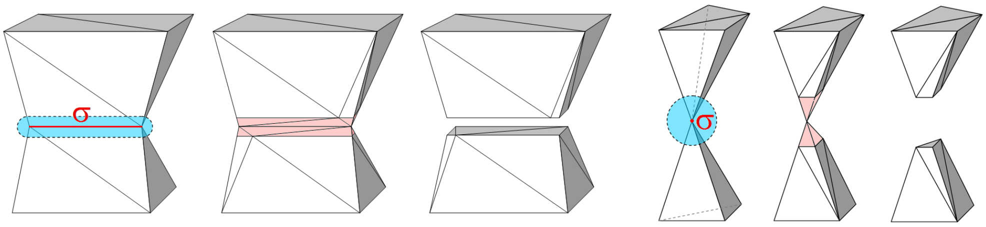

Fig. 8. Erosion of a singular simplex σ in K. Left: σ is a singular edge. Right: σ is a singular vertex. From left to right (for both the cases): the set of points having distance from $|\sigma|$ less than ϵ; the subdivision K' of K; the removal of $N(\sigma, K')$.

Note that for each singular simplex to be eroded, ϵ is less than the minimum distance of the simplex from the corresponding simplicial neghborhood's boundary, thus only elements within such simplicial neighborhood are modified.

Having this in mind, we also point out that, in principle, a different value of ϵ might be used for each singular simplex, provided that it is strictly positive and does not exceed the threshold. Nevertheless, when singularities are adjacent (eg. two singular edges sharing a common vertex) it is possible to reduce the number of splits by using a common value of ϵ. For this reason, in our prototype implementation we compute an overall threshold $\epsilon_{max}(K) = min_{\sigma \in K}\{\epsilon_{max}(\sigma)\}$ and let the user choose a value for ϵ within the range $(0, \epsilon_{max}(K)/2)$. A default value of $\epsilon_{max}(K)/4$ is used if the conversion is required to be fully automatic.

5.3 Algorithm

Also in this case, we have implemented the algorithm based on the *tetrahedral data structure*. After having pre-computed the link of each vertex and edge as described in section 4.3 we tag the identified singularities and process them one by one starting from edges. For each such singular edge $e = (v_1, v_2)$, we compute the set $E(e)$ of all the edges which are incident at either v_1 or v_2; $E(e)$ can be computed by composing the relation L (see Listing 1) with the constant relations FT, TF and FE. Each edge $e_i \in E(e)$ incident at v_1 (resp. v_2) must be split at the point p_i whose distance d_i from v_1 (resp. v_2) depends on the angle α between e and e_i (see Figure 9). Specifically, d_i can be computed as follows:

$$d_i = \begin{cases} \epsilon, & \text{if } \alpha > \frac{\pi}{2} \\ \\ \frac{\epsilon}{sin(\alpha)}, & \text{otherwise} \end{cases} \tag{1}$$

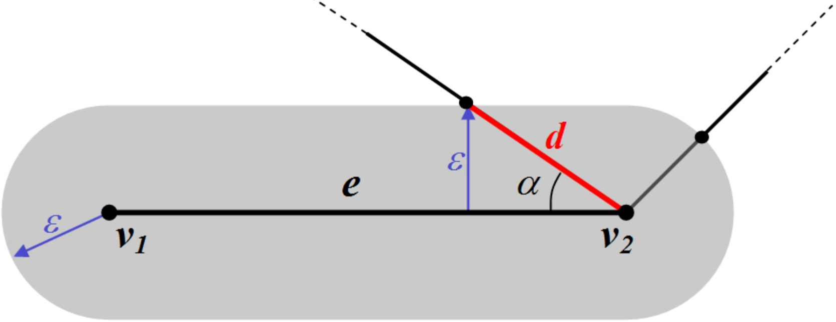

Fig. 9. Computation of the splitting points to process singular edges.

Simplicial edge-splits are standard operations widely discussed in the literature (2) and, as such, we do not explain here implementation details; however, it is worth to specify that in our setting any implementation must also take care of updating the temporary relation L (Listing 1) because, since the complex is not a combinatorial manifold, this relation cannot be re-computed in optimal time after splitting. Also, after having split an edge e which is tagged as *singular*, the two resulting sub-edges must be tagged as well.

After having processed all the singular edges, each remaining singular vertex can be processed by simply splitting its incident edges $E(v)$ that, as for singular edges, can be computed by composing the relation L with the constant relations FT, TF and FE. Each such edge e_i must be split at a distance $d_i = \epsilon$ from the singular vertex.

The whole algorithm is described in the pseudo-code of Listing 4, while an example of result is shown in Figure 10.

Listing 4. Conversion algorithm. Simplices are stored in lists which are scanned from head to tail through the "for each" statements. Edge-splits append newly created simplices to the tail of the lists; so, if a split is performed within a "for each" block, newly created elements will be processed properly.

```
ToPLManifold(Tetrahedrization $K$)
{
    CreateVertexLinks($K$)
    CreateEdgeLinksAndStars($K$)
    for each edge $e$ of $K$
        if $L(e)$ is not connected
           tag $e$ and its vertices as $singular$
    for each vertex $v$ of $K$
        if $L(v)$ is not simply connected
           tag $v$ as $singular$
    Compute $\epsilon$
    for each $e = (v_1, v_2)$ of $K$
        if $e$ is tagged as $singular$
           for each $e_i \neq e$ of $E(v_1) \cup E(v_2)$
               split $e_i$ at $p_i$
           for each $t$ of $FT(L(v_1) \cup L(v_2))$
               if $t$ is incident at $v_1$ or $v_2$
                  remove $t$ from $K$
    for each $v$ of $K$
        if $v$ is tagged as $singular$
           for each $e_i$ of $E(v)$
               split $e_i$ at $p_i$
           for each $t$ of $FT(L(v))$
               if $t$ is incident at $v$
                  remove $t$ from $K$
}
```

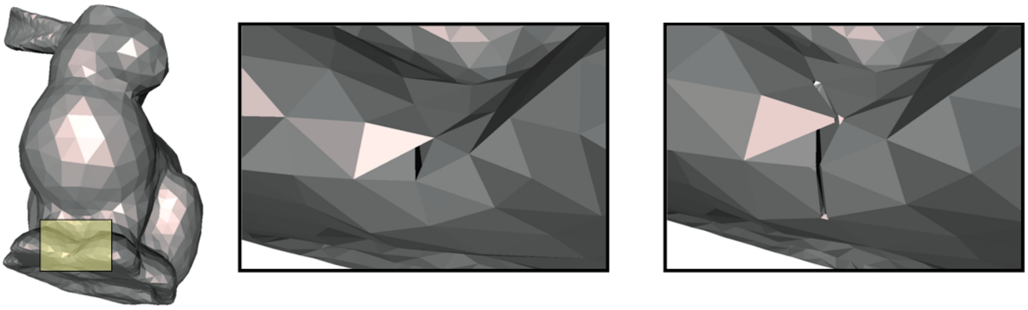

Fig. 10. The bunny model (left) has some singular simplices (center). After having run our algorithm, the sigularities have been properly eroded (right).

Note that the input to our process is a subcomplex of a combinatorial ball, which means that degenerate elements are not allowed. Thus, the two vertices of each edge have necessarily different coordinates, and the vertices inserted to split edges are computed as linear combinations of pairs of different vertices. In principle, our choice of ϵ guarantees that the new vertices do not coincide with existing ones, but in practice this may happen due to numerical approximation, and this may lead to the creation of degenerate elements. This problem never occured during our experiments, but it must be considered. Hence, though in theory it is sufficient that no degenerate elements are present in the input, in practice we need to verify that the computation of ϵ (see Section 5.2.1) does not lead to a too small value.

5.4 Geometric content preservation

We foresee that in most scenarios our conversion algorithm will be used as a pre-processing step for subsequent processing and analysis. Although it is a local operation, the erosion of a singularity may substantially alter the topology of the mesh (e.g., disconnect two components that were formerly attached) or, more in general, remove material in areas where such material had a particular meaning (e.g., in shapes resulting from optimization processes in CAD). While these problems can be negligible in some contexts, we may not assume that this is true in the range of all the possible applications, and we want to ensure that the editing operations do not jeopardize the targeted use of the model. In other words, in some scenarios it is necessary to reason on the “intended” nature of the complex rather than on its explicit representation.

To achieve this, additional knowledge must be encoded together with the edited tetrahedrization. In (4) high-level knowledge is linked to the geometry through references within XML files, hence we inspired from such a previous work. Specifically, in our prototype we have chosen to save the converted model as an XML file containing references to two other files: one containing the actual PL manifold, and another containing information about the editing process. Specifically, to keep track of the modifications introduced by the conversion, all the simplicial neighborhoods that were removed during the editing process are stored along with the manifold polyhedron (see Figure 11). This means that the XML file contains all the necessary information to recover the original polyhedron by merging the PL manifold and the realizations of the removed parts so that, if necessary, one can unambiguously reason about the original geometry of the shape.

Clearly, if only the manifold polyhedron is required, the corresponding file can be accessed directly without using the XML wrapper.

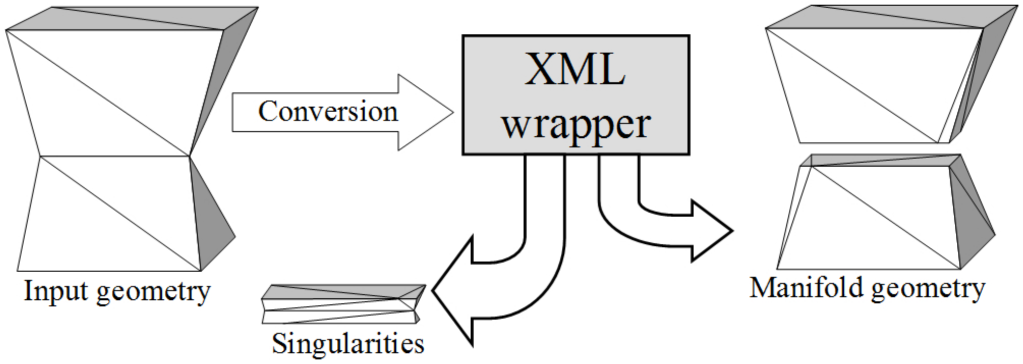

Fig. 11. Schematic summary of the output produced by our prototype to keep track of the modifications introduced by the conversion.

6 Discussion

The conversion of a complex to a PL manifold as described in Section 5 leads to a resulting simplicial complex which is a combinatorial manifold with a valid geometric realization. In other words, the conversion described in Section 5 is more general than the purely combinatorial algorithm presented in Section 4, which produces a combinatorial manifold but does not provide its geometric realization. Nonetheless, if the targeted application has requirements on the connectivity only, the method of Section 4 should be preferred as it introduces less modifications and keeps the geometry unaltered.

Both the algorithms presented assume that the input mesh is a homogeneous subcomplex of a combinatorial ball, which guarantees that singular simplices are on the boundary. Note that this condition is not too restrictive. In medicine, for example, segmented 3D images can generate tetrahedral meshes by simply triangulating the voxels; though not necessarily manifold, the resulting mesh is clearly a subcomplex of a combinatorial ball (i.e. the fully tetrahedrized 3D image). Also, our algorithm is applicable to the tetrahedrizations produced by popular meshing methods (e.g. (1)) based on a constrained Delaunay tetrahedrization of the input's convex hull.

The approach we have chosen to build combinatorial manifolds is inspired by existing methods for surface meshes. In the case of surfaces, however, all the isolated singular vertices can be treated using the same scheme (32). Conversely, for tetrahedral meshes we need to use different procedures depending on the configuration of the link of each singularity, and each such procedure has a different impact on the type of connectivity of the mesh.

Note that, in general, existing methods such as (32) or (20) cannot be used to achieve the objective of our combinatorial algorithm. To resolve configurations such as the pinched-pie shown in Figure 1, for example, it is not sufficient to duplicate the singular vertex; after the duplication, in fact, it is necessary to re-triangulate the part between the two copies with new tets to fill the resulting "combinatorial hole".

Thus, to treat a singular vertex we need to analyze its link and recursively apply a proper procedure to each component of the link. If the component is a combinatorial 2-ball we duplicate the singular vertex by changing the type of connectivity of its star (i.e. its fundamental group); otherwise, if the component is a 2-ball with n holes, we *emboss* the singularity n times without modifying the fundamental group of its star. These two strategies, namely *i.* and *ii.* in section 4.2, are illustrated in Figure 6 and Figure 7 respectively.

Though the modification of the fundamental group deriving from our approach is suitable for most real-world practical applications, alternative solutions can be implemented. For example, the two combinatorial cones depicted in Figure 6 can be merged together instead of being separated; similarly, the pinched pie configuration in Figure 7 can be converted to a solid torus instead of becoming a ball. The difference between the results of these approaches can be seen in the real-world example illustrated in Figure 12.

Note that the same arguments have been dealt with for the case of surface meshes in (20), where surface patches connected through singular points can be either separated or *regularly* merged.

For the geometric treatment of singularities (section 5) we have introduced the *erosion* operation. In this case an alternative solution would be the *addition* of material instead of the erosion, but this might produce self-intersecting meshes. Unfortunately, as we mentioned, typical tetrahedral meshes do not include explicit information about the complementary part of the solid, and this makes self-intersection checks computationally expensive. This is one of the reasons that made us exclude the possibility of displacing the copies of the singularities to transform combinatorial manifolds to PL manifolds. Furthermore, we cannot guarantee that such a *displacement* strategy is feasible for all

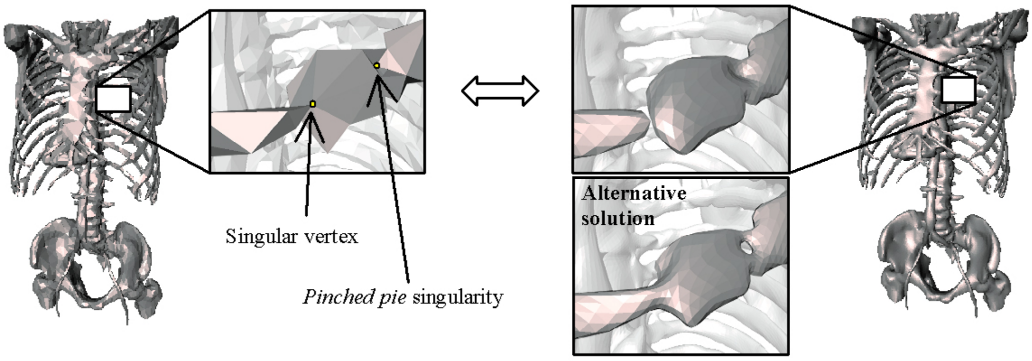

Fig. 12. Through our approach, a tetrahedral mesh resulting from a segmentation of a medical image (left) is converted to a combinatorial manifold. Its boundary has been subdivided twice using Loop's scheme (26) (right) to show the resulting connectivity around formerly singular vertices (magnified). An additional magnification shows the result of the same part subdivided after being processed by the alternative approaches discussed in section 6.

the possible configurations. Let us consider, for example, the untetrahedrizable Schönhardt prism (35). If we contract one of the side edges of the prism, thus making the two bases touch at one vertex v, the resulting polyhedron is made of 5 vertices, and thus it is guaranteed to be tetrahedrizable (35). Now, consider such a tetrahedrized polyhedron to be a *slice* of a pinched-pie configuration having v as singular vertex. After having duplicated the singular vertex v, let us consider the set of the allowed displacement directions for the copy v' of v. Clearly, v' cannot be moved towards the interior of the pinched-pie becuase this would lead to *inverted* tetrahedra having negative volume. On the other hand, we argue that the cone of the outward displacement directions may happen to contain only points that make our slice become again a sort of Schönhardt prism, which means that any displacement within this cone would necessarily produce self-intersections.

7 Conclusions

We have shown that, under specific conditions, a tetrahedral mesh with singularities can be converted either to a combinatorial 3-manifold or to a PL 3-manifold by using only local modifications.

Based on well-established mathematical concepts, we have outlined sufficient conditions that make such a conversion possible, and developed novel conversion algorithms. Moreover, we have shown that these algorithms can be implemented on a data structure designed for manifold complexes; in this way, the algorithms can be implemented as part of more general applications dealing with manifold tet meshes without the need of introducing additional structures.

In our future work, we wish to investigate procedures to minimize the topological complexity of the complex resulting from the conversion to a combinatorial manifold; in principle, in fact, the system may decide whether to *emboss* a singularity or to disconnect the components around it depending on the Betti numbers characterizing the result. Also, based on results obtained so far for surface meshes (8; 3), in our future research we wish to target a broader class of complexes, allowing the removal of possible degenerate elements prior to their conversion to PL-manifolds.

Finally, thanks to recent advances in boundary-constrained 3D mesh generation (36), we plan to implement algorithms to compute the complementary part of a tetrahedrization within its convex hull, which will allow us to propose efficient alternative solutions also for the geometric approach, in analogy with the combinatorial treatment.

Acknowledgement

This work is supported by CNR research activity ICT-P03, by the AIM@SHAPE EU NoE (IST Contract N. 506766), by the FOCUS-K3D EU CA (Contract N. ICT-2007.4.2) and by the FIRB - SHALOM project (Contract N. RBIN04HWR8). The work has also been performed within the activity of ARCES “E. De Castro”, University of Bologna, under the auspices of INdAM-GNSAGA.

References

[1] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):617–625, 2005.

[2] M. Attene. *Surface remeshing and applications.* PhD thesis, Genoa University, 2004.

[3] M. Attene and B. Falcidieno. ReMESH: An Interactive Environment To Edit And Repair Triangle Meshes. In *Procs of Shape Modelling International (SMI'06)*, IEEE Computer Society Press, pages 271–276. 2006.

[4] M. Attene, F. Robbiano, M. Spagnuolo and B. Falcidieno. Semantic Annotation of 3D Surface Meshes based on Feature Characterization. In *Lecture Notes in Computer Science, 4816 (SAMT'07 Procs.)*, pages 126–139. 2007.

[5] M. Attene, M. Ferri, and D. Giorgi. Combinatorial 3-manifolds from sets of tetrahedra. In *Cyberworlds '07 Proceedings, spec. sess. on NASAGEM workshop*, pages 367–375. IEEE Computer Society Press, 2007.

[6] S. Battiato, C. Bosco, G. Farinella, and G. Impoco. 3D CT segmentation for clinical evaluation of knee prosthesis operations. In *Procs of 4th Eurographics Conference - Italian Chapter*, Feb 2006.

[7] B. G. Baumgart. A polyhedron representation for computer vision. In *Procs of National Computer Conference*, pages 589–596, 1975.

[8] M. Botsch and L. Kobbelt. A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes. In *Procs of Vision, Modeling and Visualization*, pages 283–290, 2001.

[9] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Levy, S. Bischoff, and C. Rossl. Geometric modeling based on polygonal meshes. In *SIGGRAPH Course Notes*, 2007.

[10] E. Brisson. Representing geometric structures in d dimensions: topology and order. In *Procs of 5th ACM Symp. on Computational Geometry*, pages 218–227, New York, Jun 1989. ACM Press.

[11] E. Bruzzone and L. D. Floriani. Two data structures for building tetrahedralizations. *The Visual Computer*, 6(5):266–283, 1990.

[12] S. Campagna, L. Kobbelt, and H. P. Seidel. Directed edges - a scalable representation for triangle meshes. *ACM Journal of Graphics Tools*, 3(4), 1998.

[13] D. Dobkin and M. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 5(4):3–32, 1989.

[14] B. Falcidieno and O. Ratto. 2-manifold cell decomposition of r-sets. *Computer Graphics Forum*, 11(3):391–404, 1992.

[15] L. D. Floriani and A. Hui. Data structures for simplicial complexes: an analysis and a comparison. In *Third Eurographics Symposium on Geometry Processing*. Eurographics Association, jul 2005.

[16] L. D. Floriani, M. M. Mesmoudi, F. Morando, and E. Puppo. Decomposing non-manifold objects in arbitrary dimensions. *Graphical Models*, 65:2–22, 2003.

[17] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from manifold tetrahedralisation: application to real-time surgical simulation. *Madical Image Analysis*, 9:113–122, 2005.

[18] L. C. Glaser. *Geometrical Combinatorial Topology*, volume 1. Van Nostrand Reinhold, 450 West 33rd Street, New York, 1970.

[19] L. C. Glaser. A proof of the most general polyhedral schoenflies conjecture possible. *Pacific Journal of Mathematics*, 38(2):401–417, 1971.

[20] A. Gueziec, G. Taubin, F. Lazarus, and B. Horn. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 7(2):136–151, 2001.

[21] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[22] E. Gursoz, Y. Choi, and F. Prinz. Vertex-based representation of non-manifold boundaries. In M. J. Wozny, J. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 107–130. Elsevier,

North Holland, 1990.

[23] U. Hartmann and F. Kruggel. A fast algorithm for generating large tetrahedral 3D finite element meshes from magnetic resonance tomograms. In *Procs. of the IEEE Workshop on Biomedical Image Analysis*, page 184, 1998.

[24] A. Hui, L. Vaclavik, and L. D. Floriani. A decomposition-based representation for 3D simplicial complexes. In *Fourth Eurographics Symposium on Geometry Processing*. Eurographics Association, jul 2006.

[25] S. Lee and K. Lee. Partial entity structure: a fast and compact non-manifold boundary representation based on partial topological entities. In *Procs of 6th ACM Symp. on Solid Modeling and Applications*, pages 159–170, New York, 2001. ACM.

[26] C. Loop. Smooth subdivision surfaces based on triangles. *Ms.C Thesis, University of Utah, USA*, 1987.

[27] H. Lopes and G. Tavares. Structural operators for modeling 3-manifolds. In *Procs of 4th ACM Symp. on Solid Modeling and Applications*, pages 45–60, New York, May 1997. ACM.

[28] Y. Luo and G. Lukacs. A boundary representation of form features and non-manifold solid objects. In *Procs of 1st ACM Symp. on Solid Modeling and Applications*, pages 45–60, New York, Jun 1991. ACM.

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

[30] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, 1993.

[31] J. Rossignac. Structured Topological Complexes: A feature-based API for non-manifold topologies. In *Procs of the ACM Symposium on Solid Modeling*, pages 1–9, 1997.

[32] J. Rossignac and D. Cardoze. Matchmaker: manifold breps for non-manifold r-sets. In *Procs of 5th ACM symposium on Solid modeling and applications*, pages 31–41, New York, NY, USA, 1999. ACM Press.

[33] J. Rossignac and M. O'Connor. SGC: A Dimension-Independent Model for Pointsets with Internal Structures and Incomplete Boundaries. In *Procs of the IFIP Workshop on CAD/CAM*, pages 145–180, 1989.

[34] C. P. Rourke and B. J. Sanderson. *Introduction to piecewise linear topology*, volume 1. Springer-Verlag Berlin Heidelberg New York, 1972.

[35] E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Annalen*, 89:309–312, 1927.

[36] H. Si and K. Gaertner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163, 2005.

[37] Y. Zhuang and J. Canny. Real-time and physically realistic simulation of global deformation. In *SIGGRAPH99 Sketches and Applications*, New York, NY, USA, aug 1999. ACM Press.