Mesh Repair with Topology Control

Franck Hétroy, Stéphanie Rey, Carlos Andujar, Pere Brunet, Alvar Vinacua

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Mesh repair with topology control

Franck Hétroy — Stéphanie Rey — Carlos Andújar — Pere Brunet — Àlvar Vinacua

N° 6535

Mai 2008

Thème COG

ISSN 0249-6399 ISRN INRIA/RR--6535--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
GRENOBLE - RHÔNE-ALPES

Mesh repair with topology control

Franck Hétroy * † , Stéphanie Rey * , Carlos Andújar ‡ , Pere Brunet ‡ , Àlvar Vinacua ‡

Thème COG — Systèmes cognitifs
Équipe-Projet Evasion

Rapport de recherche n° 6535 — Mai 2008 — 20 pages

Abstract: In this research report, we propose a new method to convert a triangular mesh with geometrical and topological defects into a 2-manifold, whose topology (genus and number of connected components) is controlled by the user. We start by converting the input mesh into a thin layer of face-connected voxels; then the topology of this voxel set can be modified by the user thanks to morphological operators of different orders; at last the fixed voxel set is converted back into a triangular mesh, which both is a 2-manifold and have the desired topology.

Key-words: mesh, topology, morphology, erosion, dilatation, opening, closing, discrete membrane, isosurface

Part of this work was done while the first author was visiting the Universitat Politècnica de Catalunya in Barcelona with a Lavoisier grant from French Ministry of Foreign Affairs.

* Grenoble Universités
† INRIA
‡ Universitat Politècnica de Catalunya, Barcelona

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Réparation de maillages avec contrôle de la topologie

Résumé : Nous proposons dans ce rapport de recherche une nouvelle méthode permettant de convertir un maillage triangulaire comportant des défauts géométriques et topologiques en une 2-variété, dont la topologie (genre et nombre de composantes connexes) est contrôlée par l'utilisateur. Pour cela nous commenons par convertir le maillage initial en un ensemble face-connecté de voxels, d'épaisseur 1; la topologie de cet ensemble de voxels peut ensuite être modifiée par l'utilisateur grâce à des opérateurs morphologiques de différents ordres ; finalement l'ensemble de voxels est converti à nouveau en un maillage triangulaire, qui à la fois est une 2-variété et possède la topologie recherchée.

Mots-clés : maillage, topologie, morphologie, érosion, dilatation, ouverture, fermeture, membrane discrète, isosurface

1 Introduction

Mesh repair consists in the transformation of a mesh with singularities into a "nice" one. "Nice" often means *two-manifold*: resulting surface S is made of vertices, edges and faces such that each point on S has a neighbourhood on S homeomorphic to $\mathbb{R}^2$ (in particular, a two-manifold is a closed surface). But other conditions are sometimes required. A singularity can refer to very different things. We distinguish here three different kinds of surface singularities.

- *Combinatorial singularities* prevent the mesh, seen as a combinatorial object, to be a two-manifold. Amongst them, we list:
  - singular edges (edges with at least three incident faces);
  - boundary edges (edges with only one incident face);
  - isolated edges (edges with no incident face);
  - singular vertices (vertices whose neighbourhood, made of vertices and edges – which is called the *link* of the vertex, is not homeomorphic to a cycle or a chain);
  - boundary vertices (vertices whose link is homeomorphic to a chain but not a cycle);
  - isolated vertices (vertices which are not endpoints of any edge).

  Precise definition of combinatorial singularities can be found in [17]. See Figure 1 for an example.

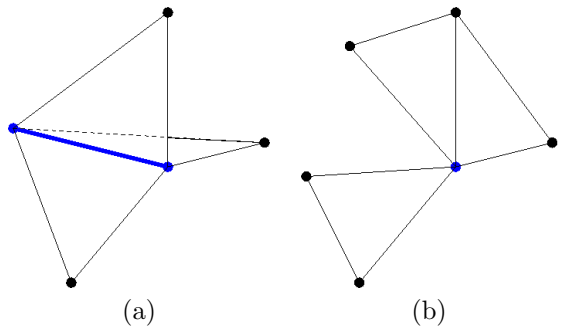


Figure 1: (a) Singular edge (its endpoints are singular vertices), (b) singular vertex.

- *Geometrical singularities* prevent the mesh, seen as the *embedding* of a surface into $\mathbb{R}^3$, to be the boundary of a three-dimensional object. For example, two triangles whose interiors intersect each other create a geometrical singularity.
- *Topological singularities* prevent the surface to get the desired genus or the desired number of connected components. As an example, complex meshes often contain small undesired handles, creating multiple small tunnels in the object they bound.

s

Another common singularity created by modern acquisition process is complex holes with (possibly) tiny islands within (see e.g. [11], and Figure 7). This singularity can be seen either as a set of combinatorial singularities (boundary edges and boundary vertices), or as a geometrical singularity, since it prevents the surface to be closed.

Mesh repair is important mainly for two reasons. First, meshes are widely used to represent (the surface of) 3D objects in computer graphics, because their flexibility is well-adapted to several tasks, such as visualization, manipulation or computations. Second, the acquisition process from a real object (for example, using a scanner) often creates inconsistent meshes, that is to say meshes having the previously cited singularities. These are due to the unavoidable limitation of the hardware's measures, but also sometimes to the inner geometry of the object: for example, the hidden part of an object cannot be reconstructed correctly by a scanner. This could not be important, but unfortunately, many applications require as input mesh a nice two-manifold. Consequently, many people have tackled this problem and tried to remove singularities from a mesh.

2 Related work

We review here different kinds of mesh repair methods. We have chosen not to classify them by *how* they work, but by *what kind of problem* they solve. First subsection describes several topological and geometrical singularity removal algorithms. Following subsection presents topology modification techniques. Last subsection briefly introduces a related problem, which is surface reconstruction.

2.1 Combinatorial and geometrical singularity removal

Mesh repair methods can be split into two categories: surface-based methods and volume-based methods. First ones operate directly on the input mesh, while second ones first convert the mesh into a set of voxels. Note that a comprehensive overview of existing works can be found in [9].

2.1.1 Surface-based methods

Amongst recent surface-based methods are [17] and [8]. Guéziec et al. propose a method to solve combinatorial singularities [17]. Their method converts a set of polygons into a 2-manifold by applying local operators. It has several advantages: since it does not handle the geometry, coordinates are not important and there is no approximation error; moreover, attributes such as colours, normals and textures can be preserved, and this algorithm works in linear time. Unfortunately, it removes only combinatorial singularities, and the user intervention is often strongly required. Borodin et al. remove several combinatorial and geometrical artefacts such as unwanted gaps and cracks using a vertex-edge contraction operator and a progressive boundary decimation algorithm [8]. Unfortunately, this method does not handle the very complex holes (with possible tiny "islands" within) produced by modern acquisition hardware. Creating a closed mesh that fills these holes is sometimes known as the *surface completion*

problem. Davis et al.'s method [11] was one of the first to tackle this problem. Unfortunately, in some cases it can produce excessively curved regions. Since then, a lot of other works have been proposed [25, 33, 31]. Surface-based methods are often automatic, but fail to repair geometrical singularities such as self-intersecting polygons: for example, in order to fill holes, these methods only consider the neigbourhood of theses holes, and do not prevent the patches they create to intersect the surface away from them.

2.1.2 Volume-based methods

Volume-based methods generate consistent surfaces, since their output will be the boundary of a volume. Moreover, they provide accurate error bounds between original and final models. One of the first volume-based method is, to our knowledge, Murali and Funkhouser's [26]. This method uses a BSP tree to represent the original surface, but is quite expensive. Recently, Ju proposed a new volume-based algorithm to convert a "polygon soup" into a 2-manifold [21], using an octree to guarantee the creation of a closed surface. This method is robust in the sense it preserves detailed geometry and sharp features of the original model. However, it does not handle correctly thin structures and does not remove topological singularities. Podolak and Rusinkiewicz recently presented a volume-based method for mesh completion [30]. The volume is represented by a graph, which is subsequently separated into two sub-graphs representing the interior and the exterior of the model. This method allows different ways to fill some holes, depending on the object desired topology. Bischoff et al. [6, 7] presented a method to remove combinatorial, geometrical and topological singularities from a CAD model, using an octree. As far as we know, this is the first work which solves all three kinds of singularities; unfortunately, it is designed mostly for CAD models, since it generates an approximation of the original model (the model is resampled), where sharp features are preserved. Moreover, holes in the mesh are closed only if greater than a user-defined threshold, whereas the value of a relevant threshold may differ from one part of the model to another, depending on the geometry.

2.2 Topology simplification

Removing topological singularities from a mesh is often seen as a different problem. Apart from methods simplifying both topology and geometry [14, 2, 4], a few methods try to simplify topology while preserving the geometry of a model. Guskov and Wood use a local wave front traversal to cut small handles [18], but they cannot detect long thin handles. Moreover, they need a 2-manifold as input. Similarly, the recent method proposed by Wood et al. [34] operates only on 2-manifolds. It finds handles using a Reeb graph, then measures their size in order to select the ones to be removed. This last task is quite slow, leading to a computation time relatively high. Nooruddin and Turk proposed a method based on a volumetric representation and on morphological operators to repair and simplify the topology of a mesh before simplifying also its geometry [28]. They first voxelize the model, using several scanning directions (the exact number is not given), then apply open and close operators to simplify the topology, extract an isosurface, and then simplify it. Whether they can completely control the topology of the final object or not remains unclear. Recently Zhou

et al. proposed a fast and robust method to break the smallest handles of a model [35]. This is done using a volumetric representation of the model, which is thinned to a topological skeleton. Smallest handles are removed by breaking skeleton cycles and then growing the modified skeleton accordingly. The same year, the authors proposed in another paper an original approach in which the user can control the location of handle removal or hole filling by sketching lines [22].

2.3 Surface reconstruction from point clouds

Apart from mesh repair, a huge amount of methods have been proposed to create manifold meshes, from various types of input data. For example, several algorithms create 2-manifolds from point clouds [1, 10, 29, 12, 20, 32]. These methods can be applied to mesh repair, only considering the vertices of the mesh. However, topology control is in general more difficult in this scope, since there is a lack of information (namely, the neighboring relations between the vertices).

3 Method overview

We propose a new method to convert a triangular mesh with geometrical, combinatorial and topological singularities into a 2-manifold whose topology is supervised by the user. It combines volume-based 2-manifold creation and adapted topology modification:

1. first, the input surface is converted into a set of voxels, named *discrete membrane*;
2. then morphological operators (openings and closings) are applied to this discrete membrane in order to detect areas which can change the topology (hole creation or filling, shell connection or disconnection);
3. the user selects the areas to be added or deleted from the set;
4. finally, the modified voxel set is converted into a 2-manifold with guaranteed topology, which is subsequently smoothed.

The pipeline of our algorithm is depicted on Figure 2.

We have chosen to use a volumetric intermediate model to be sure to remove all combinatorial and geometrical singularities: the output model is guaranteed to be a 2-manifold. Our method can be related to Nooruddin and Turk's [28] since we also use morphological operators to control the topology; however our classification between inside and outside voxels seems more coherent thanks to the discrete membrane, and we allow the user to monitor the topology modification step. As in [22], topology modification is interactive: as the user often knows the topology of the object (including the location of handles and holes), it allows a better repair than a fully automatic method. However and contrary to [22], we have chosen to guide the user during the process, by indicating topologically ambiguous areas.

Our contributions are:

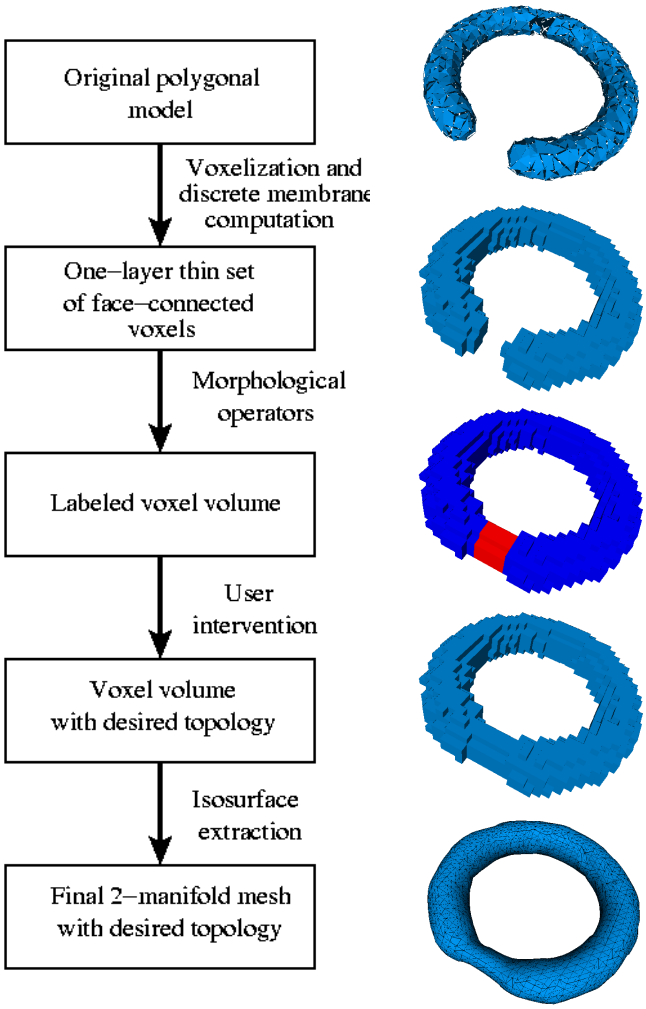


Figure 2: Algorithm overview.

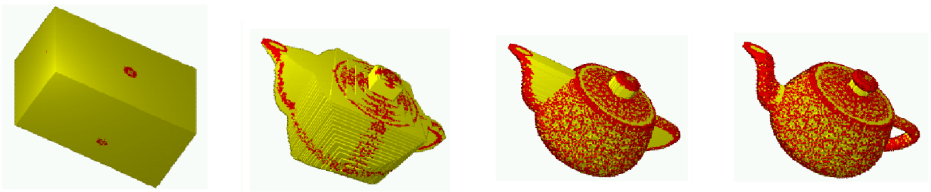
- our method solves all kinds of singularities (the input mesh can be a "soup" of triangles), and is not restricted to some particular models;
- the topology of the final mesh is completely controlled by the user;
- there is no threshold about feature size: for example, the user can choose to fill some wide holes while not filling smaller ones.

Following sections describe each stage of our algorithm.

4 Voxelization and discrete membrane creation

To construct a voxel set representing the input model, we use the algorithm described in [15]. This algorithm takes as input a cloud of points, with variable density, and computes a *discrete membrane* of voxels containing these points

(see Figure 3). It starts by voxelizing the space containing the data point set; then it contracts a set of face-connected voxels (the discrete membrane), initialized as the boundary of the voxelization, using plates of reducing size. The voxels containing the input points locally stop the shrinking; the process is over when the membrane cannot be contracted anywhere. Finally, the discrete membrane is relaxed to obtain a smoother surface afterwards. Note that the size of the voxelization is a user-defined parameter, but can also be automatically estimated.



(a) (b) (c) (d)

Figure 3: From a cloud of points to a discrete membrane [15]: (a) voxelization of the 3D space (voxels containing input points are shown in red), (b,c) silhouette shrinking with reducing plate size, (d) final discrete membrane.

Since our input is not a cloud of points but a triangulated mesh, we have modified this algorithm: we do not only compute the voxels containing the input points, but also the voxels intersecting the faces. This is done using a small recursive procedure, described below (Algorithm 1).The main advantage of this procedure is that the computation is done very fast: about 11s for a buddha model with 300.000 faces and a voxelization size of more than 3.500.000 voxels on a low-end computer.

5 Interactive topology modification using morphological operators

The construction of a discrete membrane of voxels can ensure us to get a consistent 2-manifold, if we then use relevant isosurface computation algorithms. But the surface(s) we extract from the discrete membrane may not have the topological type we want: the number of isosurfaces we extract and their genus (number of holes or, similarly, handles) may be wrong. To control the topology of the output of our algorithm, we will apply morphological operators on a volume. The volume will not be the discrete membrane itself, but the discrete membrane plus the interior of the object it bounds, which is automatically known from the discrete membrane construction. In the following sections, this voxel set is denoted as S. It is a 3-manifold: the neighbourhood of each point in its interior is homeomorphic to a sphere.

Algorithm 1 Computation of the intersection between the input mesh and the voxelization

procedure *RecComputeIntersection*($Voxel, Triangle$)
  if V is not labeled as *red* but intersects T **then**
    Label V as *red*;
    for each 6-neighbouring voxel V' of V **do**
      RecComputeIntersection(V', T);
    end for
  end if
end procedure

function *ComputeIntersection*($Voxelization, Mesh$)
  Label as *red* each voxel containing a vertex of the mesh;
  for each triangle T of the mesh **do**
    Compute the voxel V containing its barycenter;
    RecComputeIntersection(V, T);
  end for
  Return all voxels labeled as *red*;
end function

5.1 Topology of discrete volumes

The topology of a 3-manifold can be characterized by three numbers, named *Betti numbers*. j^{th} Betti number β_j is defined as the rank of the j^{th} homology group H_j (an introduction to homology groups, with precise definitions of Betti numbers, can be found in [13]). What is more interesting for our study is that Betti numbers correspond to numbers of connected components (β_0), tunnels (β_1) and voids (also called cavities, β_2) of the volume. Betti numbers are also related to the Euler characteristic of the volume χ (χ is defined as the alternate sum of the numbers of vertices, edges, faces and – in our case – voxels of the volume: $\chi = Ve - E + F - Vo$): we have the relation $\chi = \beta_0 - \beta_1 + \beta_2$.

Computing Betti numbers is not a trivial task. The number of connected components β_0 can be computed from various ways. Since in our case, β_2 is equal to 0 (we construct our volume such that it does not contain any cavity), only β_1 is not easily found. It will be computed thanks to previous relations $\chi = \beta_0 - \beta_1 + \beta_2$ and $\chi = Ve - E + F - Vo$. To compute χ efficiently (without keeping track of faces, edges, etc.), we exploit the fact that χ is additive (as did for example [5]): $\chi(A \cup B) = \chi(A) + \chi(B) - \chi(A \cap B)$. Since each vertex of the voxelization belongs to 8 voxels, χ will be eighth the sum of the local Euler characteristics around each vertex of the voxelization. The local Euler characteristics can be computed using a lookup table, since only 256 2×2 voxel configurations can occur (in fact, up to isomorphism, we only have 22 different configurations). Moreover, since χ is additive, we do not need to compute the Euler characteristic around each vertex at each step of our algorithm. Each time we will add or remove voxels, we only need to update local Euler characteristics around corresponding vertices. Once χ is computed, we immediately have the

number of tunnels in our volume: $\beta_1 = \beta_0 - \chi$.

The topology of the final surface is linked to the topology of our voxel set, since it will correspond to its boundary: the number of connected components of the surface will be equal to β_0, and its genus (number of holes) will be equal to β_1. Thus, in order to get a surface with a desired topology, we only need to compute a voxel set with the "same" topology, then to use a topology-preserving isosurface creation algorithm.

5.2 Morphological operators

In order to track down areas of the object where topology is wrong (that is to say, irrelevant handles creating tunnels or connecting different components, or on the contrary missing tunnels or bridges between several parts of a connected component), we use morphological operators. Basic operators are *erosion* and *dilatation*. The erosion operator $\mathcal{E}$ transforms the set of voxels S into the set $\mathcal{E}(S) = \{s \in S, s \text{ is not 26-neighbour of any voxel of } \mathbb{Z}^3 \backslash S\}$. The dilatation operator $\mathcal{D}$ transforms S into the set $\mathcal{D}(S) = \{w \in \mathbb{Z}^3, w \text{ is a 26-neighbour of some voxel of } S\}$ [5]. Combination of these two operators are called *opening* and *closing*: $\mathcal{O} = \mathcal{D}o\mathcal{E}$ and $\mathcal{C} = \mathcal{E}o\mathcal{D}$. Erosion and opening can widen holes and disconnect parts, while dilatation and closing can close holes and connect previously disconnected parts of the volume. Figure 4 shows these four operators applied on an example.

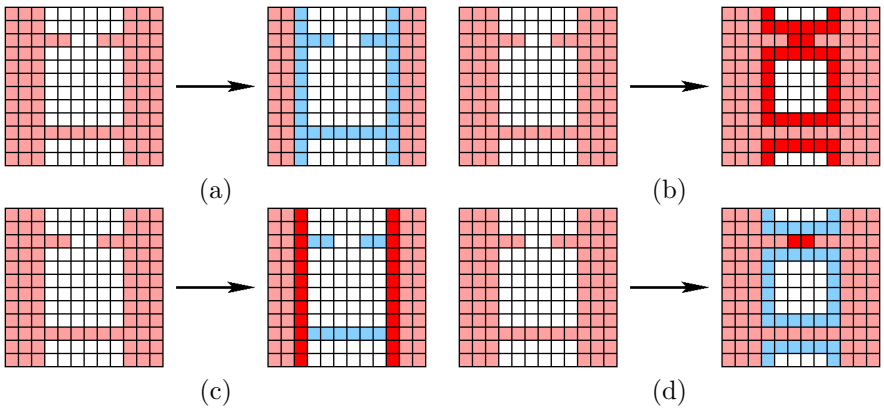


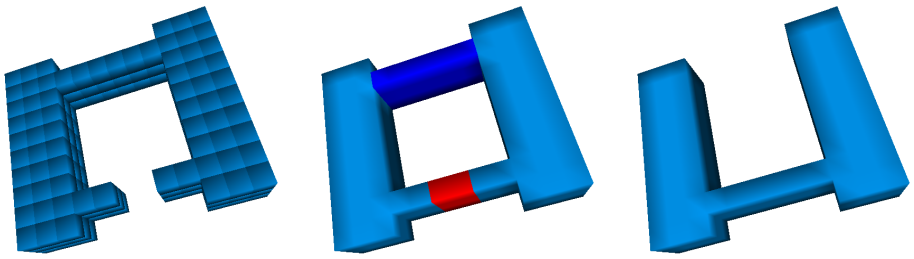
Figure 4: Morphological operators applied on a 2D set of pink pixels: (a) erosion, (b) dilatation, (c) opening, (d) closing. Removed pixels are in blue while added pixels are in red.

To go further and detect bigger topologically critical areas, we can iterate this process. Since opening and closing are idempotent operators (that is to say $\mathcal{O}o\mathcal{O} = \mathcal{O}$ and $\mathcal{C}o\mathcal{C} = \mathcal{C}$), we call *opening of order n* (noted $\mathcal{O}^n$) a sequence of n erosions followed by n dilatations, and *closing of order n* (noted $\mathcal{C}^n$) a sequence of n dilatations followed by n erosions, $n \geq 1$.

An interesting question is: should we use erosion and dilatation or opening and closing ? We choose to use these two last operators, even if they are a bit slower to compute, because they avoid shrinkage or expansion of the model, contrary to erosion and dilatation.

5.3 Algorithm

We start from the voxel set S (the discrete membrane plus its interior volume). The discrete membrane is computed as described in section 4 (see Figure 5 (a)). In order to detect topologically critical areas, we apply openings and closings to S (the order n is selected by the user). We then compute the set $o(S)$ of voxels which belong to S and not to $\mathcal{O}^n(S)$ and the set $c(S)$ of voxels which belong to $\mathcal{C}^n(S)$ and not to S. We cluster voxels of $o(S)$ and $c(S)$ in 6-connected components; for each component c, we then compute the Betti numbers of the new set of voxels $S \backslash c$ or $S \cup c$ (remember that the new Euler characteristic can be computed simply by adding or removing local Euler characteristics of vertices of c to or from the Euler characteristic of S), and compare it to the Betti numbers of S: if one of them changes, we have detected a topologically critical area, which we call a *critical component* of the voxel set. In this case, c is labelled with a special tag: "possible removal" or "possible addition". The discrete membrane together with the critical components are displayed into a visualization interface, in which the user can select to remove and/or add some critical components to the voxel set (Figure 5 (b) and (c)).



(a) (b) (c)

Figure 5: (a) Discrete membrane (b) Discrete membrane with critical components: possibly removable voxels are shown in dark blue and possibly addable voxels are shown in red (c) New voxel set, when both critical components have been selected. In this example, we applied an opening and a closing of order 1.

Implementation note: to compute 6-connected components in an effective way, we scan the voxel set once, giving each voxel V in $S \backslash \mathcal{O}^n(S)$ or $\mathcal{C}^n(S) \backslash S$ a tag. This tag is an integer (positive if the voxel belongs to $S \backslash \mathcal{O}^n(S)$, otherwise negative) indicating the number of the connected component the voxel belongs to. Tags are stored in tree structures, each tree corresponding to a connected component of voxels: if none of the 6 neighbouring voxels[1] has a tag, we create a new tag for V; if one neighbouring voxel has a tag, we give V the same tag;

[1] actually, we only need to look at the three previously studied neighbours

if at least two neighbouring voxels have a tag and if these tags are different, we give V a new tag which is stored as the parent node of neighbouring tags. Then, to keep the connected components which modify the topology of the voxel set and remove the others, we only need to scan the voxels once more.

6 Isosurface computation

Once we get a discrete volume with desired topology, we go back to a surface by using a Marching Cubes-like algorithm. We use the dual of the voxelization as grid; each vertex of this grid is labeled as *inside* the surface if it corresponds to a voxel of S, otherwise it is considered as *outside* the surface. This will lead us to a surface whose enclosed volume roughly corresponds to the volume of S. This volume may be a bit greater than the volume inside the input mesh; however, the surface will then be shrunk during the smoothing step (see section 6.2), thus reducing this volume.

6.1 Topology preservation

The standard Marching Cubes algorithm [24] is not suitable for our purpose, because of the ambiguous configurations: resulting surface can be non-manifold, or its topology may not correspond to the topology of the modified voxel set S (we want the same number of connected components as S, and any tunnel in S should correspond to a hole/handle in the surface). Moreover, because the input mesh may be very noisy, we cannot use any additional information (such as normals, or additional vertices) to help solve these ambiguities, as is requested in most "topological consistent" Marching Cubes improvements [23, 27, 19].

Therefore, we prefer to use a global approach to solve topological ambiguities [3]. This method is well-suited for our purpose because it gives even more control to the user over the topology of the result, without much effort: he only needs to tell if he wants to maximize or minimize the number of connected components and the genus of the output. An interesting fact to note is that ambiguous configurations (see [3], Figures 1 and 2) correspond to voxels of the voxel set that are 26- (vertex-) or 18- (edge-) connected, but not 6- (face-)connected, see Figure 6 for an example. Since in section 5 we used 6-connectedness, we choose as default option never to connect ambiguous configurations. According to [3], this corresponds to the policy: *"maximize the number of connected components while minimizing the genus"*.

6.2 Smoothing

The previous stage generates a 2-manifold whose vertex coordinates have been estimated in a very simple way: each vertex lies in the midpoint of an edge of the grid. In fact, since the input mesh is noisy, we cannot rely on the positions of its vertices. In case a smooth mesh is expected as output, some postprocessing is required. The smoothing method must fulfill the following requirements:

- it should not require special information (such as expected normals), since none is available;

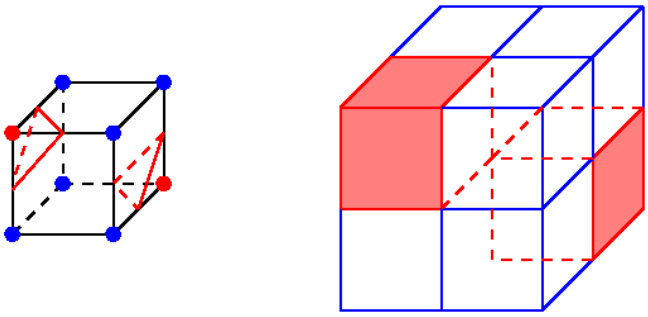


Figure 6: Marching Cubes ambiguous case (left), and the corresponding voxelization (right). Voxels of S (in red) are 26-connected, but not 6-connected.

- it should preserve features as much as possible while correctly smoothing the sharp edges introduced by the previous method;
- most importantly, it must neither change the topology of the mesh nor create new singularities, such as auto-intersections.

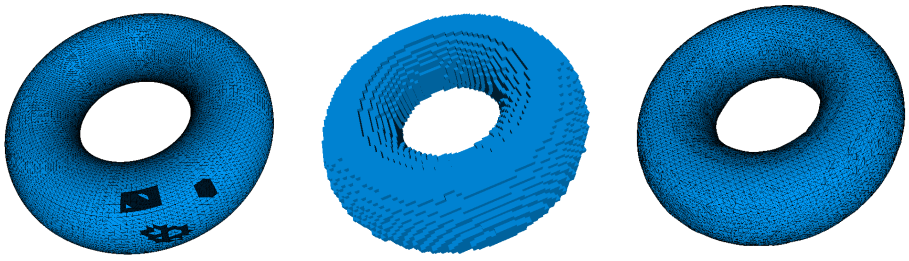
In our implementation, we have chosen to apply the bilateral mesh denoising method of [16], which is fast and satisfies the previous conditions: in practice, no singularity is created as long as the smoothing does not destroy geometrical features; the strength of the smoothing can be controlled with very simple parameters. Only one parameter has been kept in our implementation: the number of iterations. The normal to the surface at a vertex is computed using the 2-ring neighborhood of the vertex, because the mesh to be smoothed is very noisy by construction. The neighborhood used for the computation of the other parameters is set to the 1-ring neighborhood; this is a valid approximation because the aspect ratio of the faces is, by construction, uniform over the mesh.

7 Results and comments

Figure 2 shows the entire process on a model with all kinds of singularities: this model is a soup of triangles, some of them overlapping, with a hole-like region in the background. The computation of the discrete membrane leads to a voxel set with one connected component and no tunnel. Closing of order 2 enable us to create a tunnel, which leads to a torus-shaped final 2-manifold surface: one connected component, genus 1.

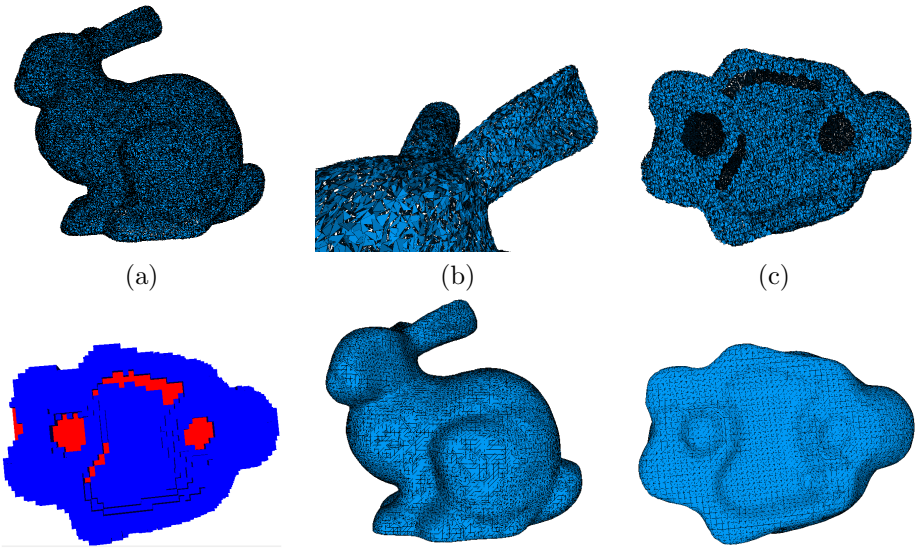
7.1 Repair of combinatorial and geometric singularities

Models with complex holes can be repaired without problem with our method, as can be seen on Figure 7. Very noisy models with holes can also be transformed into smooth 2-manifold, such as the bunny in Figure 8. In this example, closings of order 3 allowed to fill the four holes on the bottom of the original model.



(a) (b) (c)

Figure 7: (a) Input model, containing complex holes with islands within. (b) Computed discrete membrane. (c) 2-manifold result.



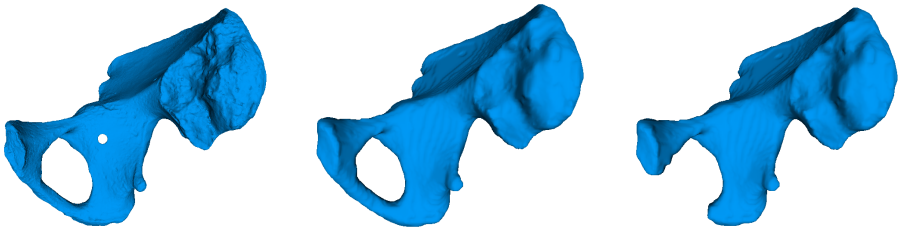
(a) (b) (c)

(d) (e) (f)

Figure 8: (a) Input model: the Stanford Bunny with all faces randomly noised. (b) Close-up of the input model. (c) Bottom view of the input model. (d) Detected critical components (in red). (e) 2-manifold result. (f) Bottom view of the result.

7.2 Repair of topological singularities

The same model can be repaired different ways, depending on the expected topology, as shown on Figures 9 and 10. On the hip model, closing of order 1 allows to fill the hole while opening of order 2 is necessary to break the handle. On the statue model, openings of order 3 are necessary to modify the topology.



(a) (b) (c)

Figure 9: (a) Input model, with genus 2. (b) Genus 1 output mesh. (c) Genus 0 output mesh.

(a) (b) (c)

Figure 10: (a) Input model, with genus 4. (b) Genus 3 output mesh. (c) Genus 1 output mesh.

7.3 Timings

Table 1 gives computation times for the four models presented in this section, on a desktop PC with a 2.4 GHz Pentium 4 processor. Input model sizes are 25,300 faces (torus), 69,450 faces (bunny), 265,000 faces (hip) and 483,000 faces (statue). The voxelization sizes are respectively 66x23x66, 69x68x55, 87x75x95 and 113x83x45 voxels. 10 iterations were performed for the smoothing stage, except on the hip model (5 iterations only). No opening or closing was performed for the torus model, since the discrete membrane alone gave the desired topology.

8 Conclusion

In this report we have presented a method to repair meshes with combinatorial, geometrical or topological singularities, or even the all of them. This method is

Model	**Membrane**	**Morpho**	**Isosurface**	**Total**
Torus	< 1	0	10	< 11
Bunny	15	8	27	50
Hip (genus 1)	14	20	19	53
Hip (genus 0)	14	18	12	44
Statue (genus 3)	3	25	34	62
Statue (genus 1)	3	25	27	55

Table 1: Computation times (in seconds) for the models of Figures 7, 8, 9 and 10.

fast and produces 2-manifolds whose topology is controlled by the user. It runs in four stages:

1. creation of a discrete volumetric model, which is a 3-manifold;
2. application of morphological operators to detect topologically critical areas;
3. selection by the user of the areas to remove or add to the model;
4. reconstruction of a smooth 2-manifold;

Apart from the selection of the topologically critical areas, the user can control some parameters (actually, one for each of the other stages):

1. voxelization size;
2. strength of the morphological operators to apply;
3. strength (number of iterations) of the surface smoothing stage.

Even if this method is better suited for smooth models because of the final smoothing stage, it has no requirement about the input model, which can be as simple as a polygon soup.

Possible enhancements of this work include:

- trying to use an octree for the first stage, to speed up the computation of the final voxel set;
- investigating ways to better control the size and shape of topologically critical components;
- modifying the surface reconstruction stage in order to possibly fit some geometrical features of the input model, if the user wants to (e.g. sharp edges).

Acknowledgements

References

[1] N. Amenta, S. Choi and R.K. Kolluri, *The Power Crust, Unions of Balls, and the Medial Axis Transform*, Computational Geometry, 19(2-3):127–153, 2001.

[2] C. Andújar, P. Brunet and D. Ayala, *Topology-Reducing Surface Simplification using a Discrete Solid Representation*, ACM Transactions on Graphics, 21(2):88–105, 2002.

[3] C. Andújar, P. Brunet, A. Chica, I. Navazo, J. Rossignac and À. Vinacua, *Optimizing the Topological and Combinatorial Complexity of Isosurfaces*, Computer-Aided Design 37(8): 847–857, 2005.

[4] C. Andújar, P. Brunet, M. Fairen and V. Cebollada, *Error-Bounded Simplification of Topologically-Complex Assemblies*, 4th Workshop on Multiresolution and Geometric Modelling, pp. 355–366, 2003.

[5] S. Bischoff and L. Kobbelt, *Isosurface Reconstruction with Topology Control*, Proceedings of Pacific Graphics, pp. 246–255, 2002.

[6] S. Bischoff and L. Kobbelt, *Structure Preserving CAD Model Repair*, Computer Graphics Forum (Eurographics 2005 Proceedings), 24(3):527–536, 2005.

[7] S. Bischoff, D. Pavic and L. Kobbelt, *Automatic Restoration of Polygon Models*, ACM Transactions on Graphics, 24(4):1332–1352, 2005.

[8] P. Borodin, M. Novotni and R. Klein, *Progressive Gap Closing for Mesh Repairing*, Advances in Modelling, Animation and Rendering, Springer-Verlag, pp. 201–213, 2002.

[9] M. Botsch, M. Pauly, C. Rössl, S. Bischoff and L. Kobbelt, *Geometric Modeling Based on Triangle Meshes*, SIGGRAPH 2006 Course Notes.

[10] R. Chaine, *A Geometric Convection Approach of 3D Reconstruction*, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 218–229, 2003.

[11] J. Davis, S.R. Marschner, M. Garr and M. Levoy, *Filling Holes in Complex Surfaces Using Volumetric Diffusion*, Proceedings of First Symposium on 3D Data Processing, Visualization, and Transmission, pp. 428–438, 2002.

[12] T.K. Dey, *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*, Cambridge University Press, 2006.

[13] T.K. Dey and S. Guha, *Computing Homology Groups of Simplicial Complexes in* $\mathbb{R}^3$, Journal of the ACM, 45:266–287, 1998.

[14] J. El-Sana and A. Varshney, *Topology Simplification for Polygonal Virtual Environments*, IEEE Transactions on Visualization and Computer Graphics, 4(2):133–144, 1998.

[15] J. Esteve, P. Brunet and À. Vinacua, *Approximation of a Cloud of Points by Shrinking a Discrete Membrane*, Computer Graphics Forum, 24(4):791–807, 2005.

[16] S. Fleishman, I. Drori, D. Cohen-Or, *Bilateral Mesh Denoising*, ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings), 22(3):950–953, 2003.

[17] A. Guéziec, G. Taubin, F. Lazarus and W. Horn, *Cutting and Stitching: Converting Sets of Polygons to Manifold Surfaces*, IEEE Transactions on Visualization and Computer Graphics, 7(2):136–151, 2001.

[18] I. Guskov and Z. Wood, *Topological Noise Removal*, Proceedings of Graphics Interface, pp. 19–26, 2001.

[19] C.-C. Ho, F.-C. Wu, B.-Y. Chen, Y.-Y. Chuang and M. Ouhyoung, *Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data*, Computer Graphics Forum (Eurographics 2005 Proceedings), 24(3):537–545, 2005.

[20] M. Kazhdan, M. Bolitho and H. Hoppe, *Poisson Surface Reconstruction*, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 61–70, 2006.

[21] T. Ju, *Robust Repair of Polygonal Models*, ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings), 23(3):888–895, 2004.

[22] T. Ju, Q.-Y. Zhou and S.-M. Hu, *Editing the Topology of 3D Models by Sketching*, ACM Transactions on Graphics (SIGGRAPH 2007 Proceedings), 26(3):42, 2007.

[23] T. Lewiner, H. Lopes, A.W. Vieira and G. Tavares, *Efficient Implementation of Marching Cubes' Cases with Topological Guarantees*, Journal of Graphics Tools, 8(2):1–15, 2003.

[24] W. Lorensen and H. Cline, *Marching Cubes: a High Resolution 3D Surface Construction Algorithm*, Computer Graphics (SIGGRAPH 1987 Proceedings), 21(4);163–170, 1987.

[25] P. Liepa, *Filling Holes in Meshes*, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 200–205, 2003.

[26] T. Murali and T. Funkhouser, *Consistent Solid and Boundary Representations from Arbitrary Polygonal Data*, Proceedings of Symposium on Interactive 3D Graphics, pp. 155–162, 1997.

[27] G. Nielson, *On Marching Cubes*, IEEE Transactions on Visualization and Computer Graphics, 9(3):282–297, 2003.

[28] F.S. Nooruddin and G. Turk, *Simplification and Repair of Polygonal Models Using Volumetric Techniques*, IEEE Transactions on Visualization and Computer Graphics, 9(2):191–205, 2003.

[29] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.P. Seidel, *Multi-Level Partition of Unity Implicits*, ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings), 22(3):463–470, 2003.

[30] J. Podolak and S. Rusinkiewicz, *Atomic Volumes for Mesh Completion*, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp 33–42, 2005.

[31] A. Sharf, M. Alexa and D. Cohen-Or, *Context-Based Surface Completion*, ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings), 23(3):878–887, 2004.

[32] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo and D. Cohen-Or, *Interactive Topology-Aware Surface Reconstruction*, ACM Transactions on Graphics (SIGGRAPH 2007 Proceedings), 26(3):43, 2007.

[33] J. Verdera, V. Caselles, M. Bertalmío and G. Sapiro, *Inpainting Surface Holes*, Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 903–906, 2003.

[34] Z. Wood, H. Hoppe, M. Desbrun and P. Schröder, *Removing Excess Topology from Isosurfaces*, ACM Transactions on Graphics, 23(2):190–208, 2004.

[35] Q.-Y. Zhou, T. Ju and S.-M. Hu, *Topology Repair of Solid Models Using Skeletons*, IEEE Transactions on Visualization and Computer Graphics, 13(4):675–685, 2007.

Contents

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399