ReMESH: An Interactive Environment to Edit and Repair Triangle Meshes

Marco Attene and Bianca Falcidieno
IMATI CNR, Dept. of Genova, Via De Marini, 6 – Genova – Italy
{attene, falcidieno}@ge.imati.cnr.it

Abstract

Polygonal meshes obtained from acquisition of real-world objects may easily exhibit topological or geometrical defects, which often prevent subsequent processing and analysis to provide satisfactory results.
This paper describes the foundations of ReMESH, a user-friendly graphical tool which incorporates several mesh-repairing features, and allows to perform a kind of low-level editing which is often missing in most existing software packages. We show how state-of-the-art techniques have been adapted and extended to form an intuitive and integrated environment, and introduce some optimizations and novel ideas that make ReMESH particularly efficient. The main application in which the tool proves to be extremely useful is the post-processing of scanned surface models. In this context, ReMESH represents a valid support for the production of certified quality meshes.

1. Introduction

Triangle meshes are becoming a de-facto standard in most application areas, mostly due to their simplicity and to the increasing support of hardware producers. In many cases the process that produces the mesh is generic, and does not take into account particular context-dependent requirements. On the other hand, however, systems that use these models are designed to work on meshes with particular characteristics and, if these are not met, software tools are likely to fail or produce unsatisfactory results. The process of adapting the raw data to a specific application context is usually referred to as *polygon mesh post-processing* [1].
As an example, if the targeted application of a huge mesh is its inspection at interactive speed, we may count on efficient simplification algorithms that reduce the mesh complexity down to acceptable limits [12][16]. While simplification is included in most mesh editing systems, however, several other post-processing methods are not yet diffused enough, though they are extremely useful. When dealing with huge scanned models, for example, interesting post-processing pipelines might include a procedure to turn a generic surface mesh into a manifold and oriented triangulation, or a method to remove degenerate or overlapping faces; in both the cases, one may want to introduce as less distortion as possible and only where strictly necessary. Methods exist in the literature, but each of tehm tackles one particular aspect of the problem and there is a lack of tools that implement and integrate repairing algorithms in an intuitive and flexible framework.

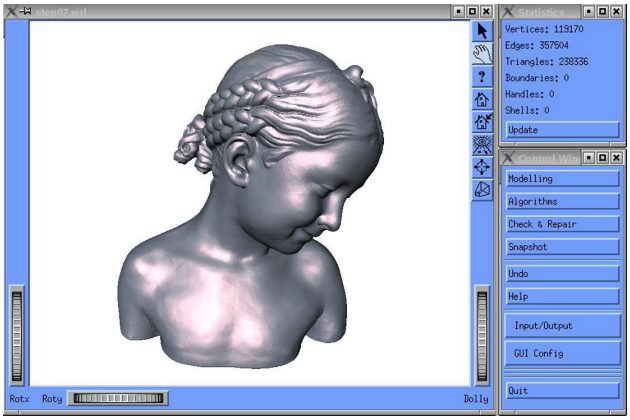

Figure 1: The graphical interface of ReMESH 1.1

With ReMESH (Figure 1), we propose a user-friendly environment for this kind of mesh post-processing. Besides a number of automatic procedures that fix typical defects of scanned models, ReMESH provides interactive tools to let the user have a complete control on the geometry and the connectivity of the mesh. As an example, it gives the possibility to swap an edge by simply mouse-clicking on it, or to automatically find and zoom on degenerate elements so that the user can interactively modify the local geometry and connectivity, again, through simple mouse clicks and drags. For completeness, however, ReMESH also provides a selection engine, and several high-level operations such as simplification, refinement, subdivision, and many others.
Summarizing, ReMESH is a triangle mesh editor that can be used for several sorts of processing, but its very distinguishing characteristic is the complete and flexible set of tools provided to repair scanned models with their typical defects. An exhaustive description of all ReMESH's features would bring us far beyond the scope of this paper (see [3] and [4] for a complete overview), so here we concentrate the focus on its *repairing* and *low-level editing* capabilities.

2. Terminology

When developing ReMESH, we took particular care to maintain a neat separation between connectivity and geometry. For this reason we make use of some notation adapted from [19], and denote a **triangle mesh** as a pair (P,K), where P is a set of N point positions $p_i = (x_i, y_i, z_i) \in$

R^3 with $1 \le i \le N$, and K is an abstract simplicial complex which contains all the topological information. The complex K is a set of subsets of $\{1, ..., N\}$. These subsets are called simplices and come in 3 types: vertices $v = \{i\}$, edges $e = \{i,j\}$, and triangles $t = \{i,j,k\}$, so that any non-empty subset of a simplex of K is again a simplex of K, e.g., if a triangle is present so are its edges and vertices.

The abstract simplicial complex K describes a topology [22], or connectivity, on P. We refer to P as to the **geometry** of the triangle mesh $M=(P, K)$, while we call **connectivity**, or topology, of M the connectivity defined on P through K. We say that M is *combinatorially manifold* iff K is a combinatorial manifold [10]. In its turn, K is a combinatorial manifold iff all its vertices are manifold, and a vertex of K is manifold if its neighborhood is homeomorphic to a disk in the topology of K.

A simplex σ of cardinality k+1 is also called a *k-simplex*. For each k-simplex σ we define a function φ:

$$\varphi : [0,k] \subset \mathrm{N} \to \mathrm{N} \text{ s.t. } \sigma = \bigcup_{i \in [0,k]} \varphi(i) .$$

Now, for each k-simplex σ in K, let us consider the subset of R^3 formed by the points x that can be expressed as the convex combination of the vertex positions of σ:

$$x = \sum_{i=0}^{k} l_i p_{\varphi(i)} \text{, with } \sum_{i=0}^{k} l_i = 1, l_i \ge 0$$

We refer to the union of all such subsets as to the *geometric realization* $S \subset R^3$ of the triangle mesh $M=(P,K)$. Thus, the geometric realization is a set of points of R^3 for which an Euclidean topology exists, and we say that S is manifold iff the neighborhood of each point in S is homeomorphic to a disk. Throughout the reminder of this paper we say that M is *geometrically* manifold, or manifold in the Euclidean sense, if S is manifold with respect to the Euclidean topology.

Note that a triangle mesh may be manifold in the combinatorial sense and not in the Euclidean one, for example when the mesh self-intersects. Also, a geometrically manifold mesh may be not combinatorially manifold. To obtain such a model, for example, start from a triangle mesh which is both combinatorially and geometrically manifold, pick an edge $e = \{i,j\}$, add a new triangle $t = \{i,j,k\}$ and set $p_k = p_j$.

If we relax the requirement of homeomorphism with a disk to the weaker condition of homeomorphism with a disk **or** with a half-disk, we say that M is manifold *with boundary*, which holds both in the Euclidean and in the combinatorial sense.

We define an *orientation* of an edge as an ordering of its two vertices. Furthermore, we call an *orientation* of a triangle an equivalence class of ordering of its vertices where $(v_1,v_2,v_3) \sim (v_{\psi(1)},v_{\psi(2)},v_{\psi(3)})$ are equivalent orderings if the parity of the permutation ψ is even. Two triangles sharing an edge e are *consistently oriented* if they induce different orientations on e. A triangle mesh is *orientable* iff all its triangles can be oriented consistently.

3. Previous Work

3.1 Fixing the connectivity

In many application contexts the input mesh is required to be manifold and orientable, while several commonly used graphic formats (VRML, STL, OFF, IV, …) may represent sets of polygons which do not necessarily constitute manifold and orientable surfaces.

In [26] a method is proposed to convert non-manifold regular sets to manifold surface meshes by identifying non-manifold edges (i.e., edges having more than two incident faces); each such edge is replicated a number of times sufficient to assign at most two incident faces to each copy. In a second step, non-manifold vertices are also identified and duplicated properly. In a similar setting, [13] identifies non-manifold edges and cuts the surface along such edges, that is, each non-manifold edge having k incident faces is turned into k boundary edges having 1 incident face each. In a second step, the user may choose whether to *pinch* or to *snap* such boundary edges; in the former case, each boundary loop is simply *zipped*, or closed, while in the latter case pairs of neighboring boundaries are merged together.

3.2. Fixing the geometry

As far as the geometry is concerned, two main approaches have been investigated in the literature. A first class of methods is based on the construction of an intermediate volumetric representation, from which the resulting surface is completely re-built. Methods of this kind fix both the geometry and the connectivity, and include [23] and [24], in which a uniform voxelization of the space is constructed and contoured [21] to extract a clean manifold surface. In [17], an adaptive space subdivision (i.e., an octree) is used, and a dual contouring approach [18] makes the method able to reconstruct sharp features.

Another important class of algorithms is based on a direct processing of the input mesh, without the need of intermediate representations. In [28], mesh zippering was introduced as a tool for merging partially overlapping range images. Tiny holes and surface cracks may be removed through patching triangles, as described in [6][8]. If bigger holes need to be fairly filled, the method introduced in [20] is able to produce patches in which both the sampling density and the normal field of the surrounding surface is reproduced. If the surface has small unwanted handles or tunnels (sometimes referred to as *topological noise*), these can be located and removed following the approach described in [15]. By inserting vertices and collapsing edges properly, [9] proposes an automatic procedure to remove degenerate triangles.

Typically, volume-based methods provide certified quality results, while direct methods may fail for many reasons. Filling non-planar holes with non self-

intersecting triangles, for example, is not always possible and determining a valid triangulation, if any, is an NP-complete problem [7] and is tackled through heuristics.
On the other hand, volumetric methods are more demanding in terms of computational resources, and introduce an error everywhere regardless of the local mesh quality. In most situations an input mesh has few local defects, while most of the triangles are correctly connected to each other. For this reason, we chose to implement a direct method in ReMESH; If the repairing cannot be completed automatically, a set of interactive tools are provided to cut, merge, remove and fill faulty regions clearly displayed by the software. In any case, however, the user may run a volumetric repairing based on the Marching Intersections approach [24].

4. ReMESH

ReMESH provides two sets of repairing operations. Operations of the first set are mandatory, are performed automatically, and serve to convert the set of input polygons into a set of manifold and oriented triangle meshes that can be properly described by ReMESH's internal data structure. The second set of operations has been designed to make the mesh more robust and utilizable in a wider range of applications.

4.1. Mandatory Repairing

While loading, an internal data structure [3] is initialized. Such a structure has been optimized to efficiently manage manifold and oriented meshes, possibly with boundary. Most graphic formats supported by ReMESH (VRML, OFF and IV), however, may represent non-manifold and/or non-orientable sets of polygons. In this case the loader runs the algorithm described in [13]. If the resulting manifold surface is not oriented, however, ReMESH assigns an orientation to one triangle for each connected component, and propagates the orientation to neighboring triangles; once all of the triangles have been visited, the mesh is cut along edges having non-consistently oriented incident triangles. Further operations include the triangulation of non-triangular faces, the removal of isolated vertices, and the duplication of non-manifold vertices.
All the operations described are performed automatically by the loader, so that the user always works on manifold and oriented meshes.

4.2. Optional Repairing

Once the data structure is properly filled with a mesh which is manifold in the combinatorial sense and oriented, various sorts of improvements may be still applied. Among its features, ReMESH provides automatic and user-assisted functionalities to remove *defects* that would cause subsequent processing to fail.

Removal of small and isolated components

In many cases a 3D application expects the input mesh to be made of a single connected component; on the other hand, however, several mesh generation techniques often create tiny disconnected patches [21][24] near a main surface, and removing all but the biggest connected component is a functionality of clear interest. In ReMESH, such a functionality has been implemented by simply counting the number of triangles forming each component, that is, the *biggest* component is the one with the highest number of triangles.

Hole Filling

Also, many applications require the input to be watertight (i.e., without surface holes). Filling holes with ReMESH is a very flexible and controllable operation. It is possible to triangulate a single hole by clicking on one of its boundary edges, or to fill all the holes without user-assistance. In both the cases, the hole(s) may be simply triangulated [6] or it(they) may be patched by inserting new vertices so as to reproduce the sampling density and the normal field of the neighboring surface [20]. In this latter case, we have analyzed the method proposed by Liepa in [20] and adapted it for a more general case.
Essentially, Liepa's hole-filling procedure is subdivided in three steps: *triangulation*, *refinement* and *fairing*. In the first step the hole is triangulated as suggested in [6]; In the second step, new vertices are placed within the newly inserted triangles to produce the desired vertex density. Specifically, each new triangle which is too big wrt the neighboring faces is subdivided in three sub-triangles by inserting a new vertex at the barycenter. At the end, edges are iteratively swapped so as to verify *in-sphere* tests in a Delaunay-like manner. Finally, in the third step, the new vertices are moved to positions that minimize an appropriate functional and make the patch's normal field fairly continuous with respect to the neighboring surface.
In the context of repairing big scanned models, we found it necessary to perform some little improvements and extensions to the original method.
First, we have observed that when the hole is big with respect to the average length of its bounding edges, the initial triangulation is inevitably made of some long and thin triangles, and the Delaunay-like optimization does not always converge to a satisfactory result due to limited robustness when computing circum-spheres. In these cases one might use exact arithmetics [27], but for big models this would lead to unacceptably long execution times. We have found that minimizing the length of internal edges provides excellent results, and represents a much faster and robust procedure. Thus, in the iterative optimization, we swap an edge only if its length after the swap decreses.
Our main innovation, however, consists of a very simple but effective approach to merge together pairs of disconnected boundaries (Figure 2). Given two boundary loops, b_1 and b_2, the user clicks on a pair of vertices belonging to the two boundaries. After connecting them through a new edge that we call the *gate*, at each step the algorithm attaches a new triangle t to the gate and to an

edge of one of the two boundaries, and the newly created edge becomes the new gate. This operation is repeated until all the edges belonging to both b_1 and b_2 become internal. For each new such triangle t, we choose its third vertex (i.e., the one which is opposite to the gate) either on b_1 or on b_2, and such a choice is driven by the attempt to complete the two boundaries roughly at the same *speed*. More formally, let E_1 and E_2 be the sets of edges belonging to b_1 and b_2 respectively, and let l_i be the total length of edges in E_i. Also, we make use of the set $B_i = \{e \in E_i : e \text{ is on boundary}\}$ so that, initially, $B_i = E_i$. Let n_i be the total length of edges in B_i. At each step, the third vertex of the new triangle is chosen so as to minimize the quantity $c = |n_1 l_2 - n_2 l_1|$.

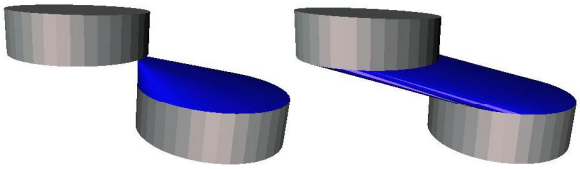

Figure 2: Left: boundaries connected as suggested in [20]. Right: connected using our method.

The other two steps of our modified Liepa's hole-filling do not require the patching triangulation to be simply connected, so we just use them on our cylindircal patch.
Our minimization of c makes the algorithm proceed at about the same *speed* on both the boundary loops, and provides much more satisfactory results than simply adding two triangles to change the topology and then using the triangulation algorithm (see Figure 2), as suggested in [20].

Removal of tiny handles

Another typical source of malfunctions for many 3D applications is the presence of unwanted handles or tunnels. Mostly often, such topological defects are due to the mesh generation process, and do not correspond to actual topological entities in the original 3D data. In [15] such defects have been conceptualized under the term *topological noise*, and in the same work the authors propose a procedure to identify and remove them. In ReMESH, we implemented an automatic procedure to identify and select tiny handles. First, the user clicks on the mesh and, by dragging the mouse, draws a semi-transparent sphere on it. Then, all the handles that fit within the sphere are automatically selected for further processing. The selected triangles, for example, may be removed and the resulting holes patched, with the result of reducing the genus of the mesh.
The algorithm implemented in ReMESH is based on a depth-first, triangle-based visit of the mesh, as the one used in [25]. For the sake of explanation, we imagine to remove one triangle at a time, starting from a seed one, in a depth-first manner. In some cases, for example when wrapping around a cylinder, expanding the hole requires its boundary to split. In particular, it happens that the next triangle to be removed has only one boundary edge but all its three vertices on the boundary. In this case, we use the notation introduced in [25] and label such a triangle with an S symbol (see Figure 3). Let e_1 and e_2 be the two non-boundary edges of such a triangle; after removing it, the removal proceeds from the only triangle attached to e_1, while e_2 is put onto a stack for future processing. If there are no handles, the surface between e_1 and e_2 is disconnected, so we remove the component containing e_1, and then the other one separately. If, when removing triangles from the first component we reach e_2, we went across a handle. In this case, we tag the S triangle with a particular symbol S'. Actually, there is no need to remove triangles, while it is sufficient to mark them as visited.

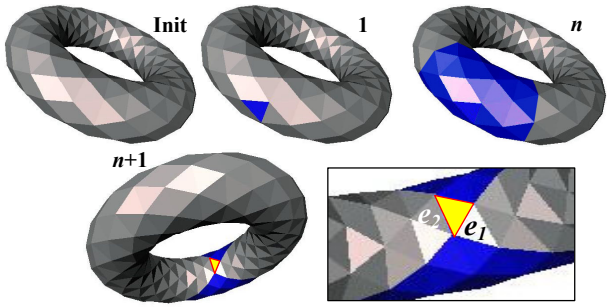

Figure 3: Example of depth-first visit starting from a seed face and tagging of an S triangle.

Typically, topological noise is present on meshes obtained from laser scanner data [15]. We have experienced that for such class of meshes, in which the variation in edge length is not excessive, S' triangles are geometrically close to actual handles. Through this observation, we can locate most of the tiny handles by simply analyzing the topology of a spherical neighborhood (whose radius was interactively selected by the user) of each such triangle. Using this procedure, ReMESH locates tiny handles much more quickly than the method proposed in [15]. As an example, we run ReMESH on an old P3 450MHz, and we could locate 98 of the 104 handles of the famous Buddah model (1087716 faces) in about 32 seconds, which is twenty times less than what reported in [15] for the same mesh on a faster PC (the comparison is fair because the time-consuming part of [15] is the location, and not the simplification of the handles).

Degenerate triangles

The problem of robustness of geometric algorithms has been receiving a lot of attention for more than 15 years [14][11], and it remains an important area of research. When designing a geometric algorithm, in fact, a researcher uses the theory of real numbers and their **exact** arithmetic. At implementation time, however, real numbers must be approximated with finite precision representations and the error introduced cannot always be neglected, as it may cause a wrong branching in the process pipeline and, as a consequence, it may cause a

topological inconsistence or a failure of the algorithm. Typically, such problems are originated from degenerate or nearly degenerate triangles having extreme angles, also referred to as *skinny* triangles [9].

In order to produce *robust* meshes, we chose to implement a strategy based on the Epsilon Geometry introduced in [14]. ReMESH provides the possibility to remove all the triangles with angles smaller than ε or bigger than π-ε, where ε is a customizable threshold angle. The removal is carried out through swapping and contraction of edges, inspired from ideas of [9]. Specifically, triangles having a nearly flat angle are treated by swapping the edge opposite to such angle, while triangles having a nearly null angle are removed by collapsing the edge opposite to such angle to its mid-point (checks are performed in this order). The default value of ε is $arcsin(10^{-5})$; by experiment, this value proved to be a good compromise between precision and robustness.

Even if the strategy implemented does not guarantee robustness in all the cases, we have found that it avoids nearly all of the most common problems when dealing with non robust applications. All the check and repairing tasks can be performed by the user after selecting a new value for ε. This can be useful when the model being edited must become the input for a less robust system.

Sharp feature recovery

During a 3D acquisition process, it is extremely hard to align the scanning pattern with sharp edges and corners of the object being digitized. As a result, in the digital model such sharp features are replaced by irregular chamfer triangles. ReMESH provides an implementation of the EdgeSharpener algorithm that, after an identification of chamfer triangles, inserts new vertices on them and reconstructs sharp edges and corners based on an extrapolation of neighboring smooth regions [5] (see Figure 4).

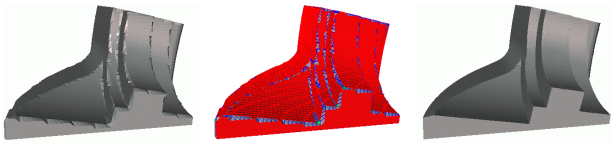

Figure 4: From left to right: original; smooth regions detected; sharpened model.

4.3. Manual repairing by low-level editing

In some cases, all the automatic functionalities provided by ReMESH may fail to remove some defects. This may happen, for example, when too many degenerate triangles are adjacent to form surface spikes (see Figure 5). In these cases, ReMESH provides a button that looks for degeneracies; the last problem detected, if any, is automatically visualized by properly pointing the camera, and the tool provides a number of interactive operations that let the user choose what to do in each particular situation. By simply clicking and dragging the mouse, it is possible to swap edges, or to remove one triangle at a time, or to select a region and re-triangulate it using a Delaunay-like method, or to locally run some iterations of Laplacian smoothing, and so on.

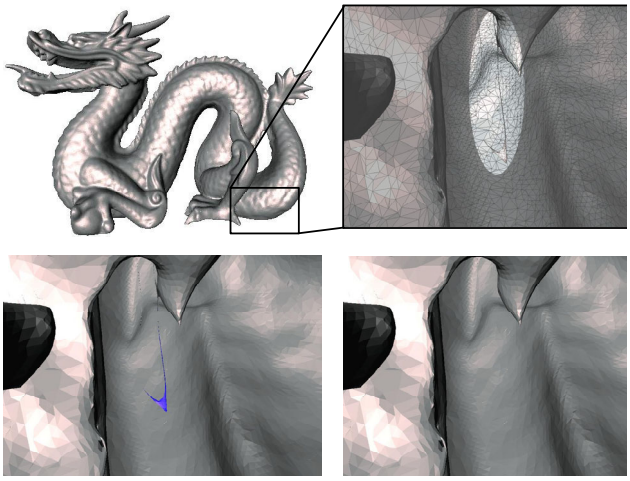

Figure 5: Automatic location of a spike incorporating degenerate triangles (top row). Selected region (bottom-left) and removal (bottom-right).

Selections

In ReMESH it is possible to define regions of interest, or selections. A selection is a set of mesh triangles on which further operations have their effect. By clicking on a point of the surface, the user selects the center of a sphere whose radius is interactively controlled by dragging the mouse without releasing the button. When the button is released, ReMESH considers all the triangles entirely contained in the sphere. These triangles may constitute several disconnected components; the component containing the center of the sphere is selected and highlighted. To refine a selection, the status of each triangle may be toggled by clicking on it, that is, a selected triangle becomes unselected and vice-versa. Also, it is possible to grow, shrink and invert the current selection, to add another selection by holding the shift key, to remove or intersect selections and so on.

5. Conclusions

In this paper we have introduced the foundations of ReMESH, a graphical tool integrating most of the existing techniques developed so far to repair polygonal meshes. Besides presenting the toolbox, we discussed some improvements and extensions to state-of-the-art methods which have been implemented in ReMESH.

In version 1.1, ReMESH has been extensively tested on several scanned models ranging from thousands to millions of faces, and provided extremely good results in all the cases.

ReMESH was used to post-process raw geometry and produce several high-quality models for AIM@SHAPE's Shape Repository [2]. The main goal of such repository is

to become a reference point for persons looking for "certified" shapes, in which the plain geometry is endowed with metadata specifying, among the others, interesting characteristics such as manifoldness, orientation, watertightness, genus, and so on.

The model shown in Figure 1 was digitized through a Minolta Vivid 910 laser scanner. After aligning and merging the range images, the resulting mesh was not manifold and not orientable, had numerous degeneracies, tiny disconnected components and plenty of surface holes due to visibility occlusions. All these problems have been fixed through ReMESH, so that the model could become a proper input for further processing. Such a fixed model is made of more than one million faces, and became the *representative* certified mesh of AIM@SHAPE's Shape Repository.

Acknowledgements

This work is partially supported by the EU Project AIM@SHAPE (Contract # 506766). Thanks are due to Dr. Michela Spagnuolo for her encouragement in writing the paper, to all the members of the Shape Modeling Group of the IMATI-GE/CNR, to the authors of the papers that inspired this work, and to the people that, through discussions and suggestions, made it possible to develop ReMESH. The models used in the illustrations are courtesy of AIM@SHAPE (Figure 1), H. Hoppe (Figure 4) and Stanford Univ. (Figure 5).

References

[1] Adamson, A., Alexa, M. and Attene, M. 2004. *Post-Processing.* In AIM@SHAPE's survey on Acquisition and Reconstruction (Technical Report), 146-152.

[2] AIM@SHAPE, FP6 IST NoE 506766. *Shape Repository v2.0*. http://shapes.aim-at-shape.net/

[3] Attene, M. 2004. *ReMESH: An Interactive and User-friendly Environment for Remeshing Surface Triangulations*. IMATI-GE/CNR Technical Report 06/2004.

[4] Attene, M. 2005. *ReMESH 1.1 - User Manual*. http://www.ima.ge.cnr.it/ima/personal/attene/PersonalPage/Remesh/1.1/manual.pdf

[5] Attene, M., Falcidieno, B., Rossignac, J. and Spagnuolo, M. 2003. *Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces*. In Proceedings of the 1st Eurographics Symposium on Geometry Processing, 63-72.

[6] Barequet, G. and Sharir, M. 1995. *Filling Gaps in the Boundary of a Polyhedron*. Computer-Aided Geometric Design, 12, 2, 207-229.

[7] Barequet, G., Dickerson, M. and Eppstein, D. 1998. *On triangulating three-dimensional polygons*. Computational Geometry 10, 155-170.

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

[9] Botsch, M. and Kobbelt, L. P. 2001. *A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes*. In Proceedings of Vision, Modeling and Visualization, 283-290.

[10] De Floriani, L., Morando, F. and Puppo, E. 2003. *Representation of Non-manifold Objects through Decomposition Into Nearly manifold parts.* In Proceedings of ACM Solid Modeling '03, 304-309.

[11] Fortune, S. 1996. *Robustness issues in geometric algorithms*. In Proceedings of the 1st Workshop on Applied Computational Geometry (WACG '96), 9-14.

[12] Garland M. and Heckbert, P.S. 1997. *Surface simplification using quadric error metrics*. In Proceedings of ACM SIGGRAPH '97, 209-216.

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

[14] Guibas, L., Salesin, D. and Stolfi. J. 1989. *Epsilon geometry: building robust algorithms from imprecise computations*. ACM Symposium on Computational Geometry, 5, 208-217.

[15] Guskov, I. and Wood, Z. 2001. *Topological noise removal*. In Proceedings of Graphics Interface '01, 19-26.

[16] Hoppe, H. 1997. *View-dependent Refinement of Progressive Meshes.* In Proceedings of ACM SIGGRAPH '97, 189-198.

[17] Ju, T. 2004. *Robust Repair of Polygonal Models.* ACM Transactions on Graphics, 23, 3 (Procs. of SIGGRAPH 2004), 888-895.

[18] Ju, T., Losasso, F., Schaefer, S. and Warren, J. 2002. *Dual Contouring of Hermite Data*. ACM Transactions on Graphics, 21, 3 (Proceedings of SIGGRAPH '02), 339-346.

[19] Lee, A. W. F., Sweldens, W., Schröder, P., Cowsar, L. and Dobkin, D. 1998. *MAPS: Multiresolution adaptive parameterization of surfaces*. In Proceedings of ACM SIGGRAPH '98, 95-104.

[20] Liepa, P. 2003. *Filling Holes in Meshes*. In Proceedings of the Eurographic Symposium on Geometry Processing, 200-206.

[21] Lorensen, W. and Cline, H. 1987. *Marching Cubes: a high resolution 3D surface construction algorithm*. In Proceedings of ACM SIGGRAPH '87, 163-169.

[22] Munkres, J. R. 2000. *Topology.* Prentice Hall, New Jersey, USA.

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

[24] Rocchini, C., Cignoni, P., Ganovelli, F., Montani, C., Pingi, P. and Scopigno, R. 2001. *Marching Intersections: an efficient resampling algorithm for surface management*. In Proceedings of Shape Modeling International (SMI '01), 296-305.

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

[26] Rossignac, J. and Cardoze, D. 1999. *Matchmaker: Manifold Breps for non-manifold r-sets*. In Proceedings of the ACM Symposium on Solid Modeling, 31-41.

[27] Shewchuk, J. R. 1997. *Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates*. Discrete & Computational Geometry, 18, 305-363.

[28] Turk, G. and Levoy, M. 1994. *Zippered polygon meshes from range images*. In Proceedings of ACM SIGGRAPH '94, 311–318.