Re-meshing Techniques for Topological Analysis

Marco Attene, Silvia Biasotti, Michela Spagnuolo
Istituto per la Matematica Applicata
Consiglio Nazionale delle Ricerche
{attene, biasotti, spagnuolo}@ima.ge.cnr.it

Abstract

A method for the extraction of the Extended Reeb Graph (ERG) from a closed 3D triangular mesh is presented. The ERG codes the relationships among critical points of the height function associated to the mesh, and it can represent isolated as well as degenerate critical points. The extraction process is based on a re-meshing strategy of the original mesh, which is forced to follow contour levels. The occurrence and configuration of flat areas in the re-triangulated model identify critical areas of the shape, and their relationships allow the reconstruction of the global topological structure of the shape.

1. Introduction

The combination of geometry and topology provides interesting insight into several computer application problems that involve shape modelling and understanding. In particular, computational topology has been proposed to identify a new field of research in which topological problems are approached taking into specific consideration the discrete domain and the computational requirements which characterise digital applications [1], [2]. Among the different disciplines related to topology, differential topology deals with the relations among shape characteristics and properties of the surface defining the shape. In particular, Morse theory states that the topology of a given manifold can be described by analysing the critical points of a smooth function defined on the manifold itself [3][4]. In other words, global properties and connectivity information of a shape can be computed provided that the critical points of a smooth function defined over the shape are known.

In this context, an interesting topological structure, known as Reeb graph, codes the surface shape by storing the evolution of contours obtained by intersecting the surface with a plane, which sweeps along a predefined direction. A method for the extraction of Reeb graphs from triangular meshes representing bi-variate surfaces (2.5D surfaces), has been presented in [5][6]. An efficient solution to automatically compute the Reeb graph of a mesh is to compute a number of parallel slices in the given direction, and then study the triangulation of the contour lines, using the contours as line constraints. First of all, it can be easily seen that flat areas of the constrained mesh localize surface critical points. Then, since vertices of the constrained mesh may only belong to contours, the topological adjacency of critical points can be reconstructed using edge-based adjacency among contours. The method can be applied to a wide class of surfaces, in particular surfaces whose critical points are not isolated, a useful advantage when dealing with real objects.

In this paper, the extension of this approach to closed 3D meshes is presented. The main difference consists in the intermediate step of constrained triangulation. The reconstruction of a constrained triangulation from the computed contour lines would require, indeed, knowledge of the shape topology for solving correspondence and branching problems [7][8][9], giving rise to a classical "chicken and egg" problem. Therefore, we have adopted an approach based on a re-meshing of the original surface, so that all its vertices will eventually lye on the computed contour levels. Once the shape is re-meshed, flat regions can be again used to locate critical points, using a slightly modified classification scheme as for

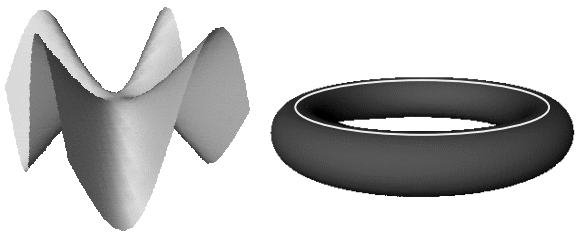
the 2.5D case, and their relationships will be easily tracked using the contour constraints.

The reminder of the paper is organized as follows. First, the theoretical background of this work is briefly described with reference to previous work in this area. Then, the approach for constructing the Reeb graph is outlined and the re-meshing strategy adopted to extend the approach to 3D closed meshes is presented. Results and some discussions on the advantages and limits of the method conclude the paper.

2. Background

In applications related to geometric modelling, it is quite natural to choose the height function and its criticalities to study a given shape. Intuitively, the height function h of a smooth manifold M, embedded into the usual three-dimensional Euclidean space, is the real function which associates to each point on the surface its elevation, that is, $h(P)=h((x_P,y_P,z_P))=z_P$ for every $P \in M$.

The *critical points* of h, and in general of any real smooth function defined on M, are the points of M at which the gradient is zero, i.e. $\nabla(h(P))=0$ or, stated differently, the tangent plane is horizontal. Moreover, h is called Morse if all of its critical points are non-degenerate, that is, if the Hessian matrix H of the second derivatives of h is non-singular at that point. In particular non-degenerate critical points are isolated; therefore, surfaces with plateaux or volcano rims do not comply with the definition of Morse function. Figure 1(a) depicts an example of degenerate fourth order saddle while in (b) an example of non-isolated critical points is shown.



(a) (b)

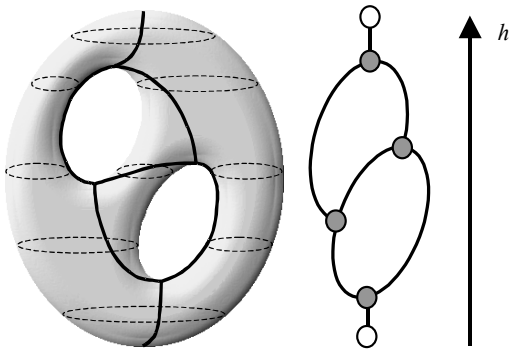
Figure 1. Examples of degenerate critical points: a quadriped saddle (a) and a line of maximum points (b).

Critical points of the height function, commonly known as peaks, pits and passes, are useful for shape characterization as they are considered among the basic elements by which shapes are perceived and organised [10]. The configuration of critical points has been used for shape description by several authors for different applications. In [11], for example, the configuration of critical points is used to code and classify the shape of a smooth function, while in [12] the configuration of critical points is used as a constraint for the simplification of meshes while preserving the topology of the shape. In [13], a wave traversal algorithm is proposed to navigate a triangular mesh according to distance criteria among elements in the simplicial mesh. A similar wave-front propagation has been used to deduce the global topology from volume data set in order to extract correct polygonal meshes representing iso-surfaces [14].

Starting from the Morse theory, an interesting topological structure has been defined by Reeb, which represents a smooth surface by coding the evolution of its contour levels [15]. More precisely, the Reeb graph of a manifold M with respect to a real valued function f is defined as follows:

Definition: Let $f : M \rightarrow \Re$ be a real valued function on a compact manifold M. The *Reeb graph* of M wrt f is the quotient space of $M \times \Re$ defined by the equivalence relation "~", given by:

$(X_1, f(X_1)) \sim (X_2, f(X_2)) \Leftrightarrow f(X_1) = f(X_2)$ and X_1 and X_2 are in the same connected component of $f^{-1}(f(X_1))$



(a) (b)

Figure 2. The Reeb equivalence classes on a bi-torus (a) and its graph representation (b).

In practice all points of a compact manifold having the same value under a real function and whose pre-

image belongs to the same connected component are collapsed into one element. Since the contour topology changes only in correspondence of critical levels of the height function, the Reeb quotient space can be described as a graph [3] (see Figure 2).

The use of Reeb graphs has been already addressed in computer graphics [16], [17], but it has been generally limited to Morse mapping functions. Therefore, degenerate critical points are not allowed or handled by local perturbations of the surface, which introduces artefacts not corresponding to any shape features, thus leading to imprecise shape description [16].

3. Critical areas and graph construction

Given a triangular mesh T representing a single-sided two-manifold surface without boundary, we want to compute a topological graph with the same properties as the Reeb graph. The Extended Reeb Graph representation (ERG) for 2.5D surfaces represented by triangular meshes has been fully described in [5]. The ERG derives from the direct application of the Reeb graph definition to degenerate as well as non-simple height functions, which allows us to consider a broader class of surfaces. The main idea is to consider critical *areas* instead of critical points and identify, starting from them, the smallest area on the mesh whose behaviour is topologically equivalent to the critical area (influence zones). Influence zones are the starting points for identifying the portion of the mesh contributing to the definition of Reeb graph arcs.

More precisely, a *sufficiently* dense number of sections of a triangulated mesh T is computed and the mesh T^* is defined as the triangulation of the contours, constrained to the contours themselves, i.e. the edges of T^* never cross a contour. In T^*, maximum and minimum points of T are localised by flat regions whose vertices belong to the same contour [5] [18]. Saddle points are localised either by flat areas having vertices at the same elevation but belonging to different contours (see Figure 3(a)), or by regions composed of two adjacent non-flat triangles such that their common edge is flat (see Figure 3(b)). These regions of T^* are defined to be the *critical areas* of T and they may be simply as well as multiply-connected: in Figure 3(c), an example of multiply-connected critical area of maximum is shown.

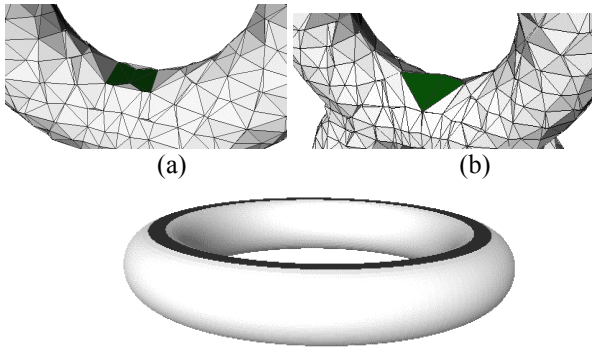


Figure 3. Examples of critical areas.

The classification of critical areas as maximum, minimum and saddle is done by checking the number of non-constrained edges in the boundary and by analysing the ascending/descending direction of the surface across their boundary. In Figure 4, a bi-torus is depicted with its contour levels and the resulting critical areas (only the visible ones are highlighted, but there are also two symmetrical saddles and a minimum).

Figure 4. The constrained mesh and the resulting critical areas.

The notion of critical area is sufficient to fully describe the topological behaviour of the surface around maximum and minimum areas but it is not enough for saddle ones. In order to give a better characterization of the surface's topology, the notion of influence zone is introduced, as the smallest area of the mesh having the same topological behaviour as the critical area. The influence zone of minimum or maximum areas correspond to the critical areas themselves, while for saddle ones it is the saddle-like portion of the mesh within which the topology of the contours changes. Therefore, the influence zone of a saddle area is composed by the corresponding critical

areas and by the part of the mesh contained in the contours nearest to the critical area. Note that, saddle critical areas at the same elevation may have the same influence zone. An example is shown in colour plate 1, where the influence zones of saddle areas are depicted in yellow. If the influence zone of different saddle areas coincide, this indicates that the height function is non-simple at that level.

Based on these concepts, the ERG is defined as a graph whose *nodes* correspond to the influence zones of simple critical areas, while *macro-nodes* are used to represent complex critical areas. The ERG *arcs* represent the topological adjacency among critical areas and are defined using a region-growing process, as described by the following pseudo-code.

Let *N* be the set of nodes and *A* the set of arcs of the ERG, then the ERG construction algorithm is:

```
CharacteriseCriticalAreas();
N:=DetectInfluenceZones();
A:=LinkNodesbyInfluenceZone(N),
for (each node in N)
  {for (each non visited growing direction)
    {while (!(ReachAnotherInfluenceZone))
       UpperLevelExpansion(node);
     ConnectAreas(node, newarea);
    }
  }
```

First of all, the function *CharacteriseCriticalAreas()* computes the critical areas and order them by elevation. The function *DetectInfluenceZones(N)* determines the influence zones, which define the set of nodes *N* (see Figure 5(a)). Then, a first set of ERG arcs is extracted by expanding the influence zones of simple maximum or minimum areas until the boundary of another influence zone is reached. This latter step corresponds to the function *LinkNodesbyInfluenceZone(N, A)*. In this manner, the arcs connected to terminal nodes of the ERG are identified (see Figure 5(b)). To complete the ERG construction, the links between saddles and complex areas have to be determined. These arcs are determined by expanding, if possible, the influence zones following the free directions in their outmost boundary component (see Figure 5(c)). In other words, all directions that do not correspond to an already identified arc are checked and the expansion is done in the ascending directions (*UpperLevelExpansion(node)*) until another influence zone, i.e. node, is reached. In this case, an arc is defined between the starting node and the reached one (*ConnectAreas(node, newarea)*). When all the free ascending directions of each node are considered, the ERG construction process ends.

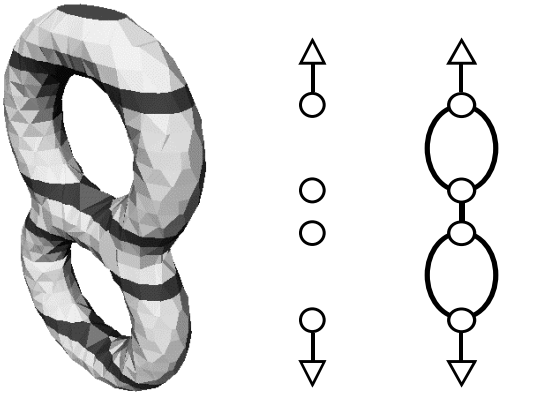


Figure 5. The first step of the ERG construction (b) and the final one (c) computed for the surface in (a).

Some comments can be made. First of all, since we use the height function to map the surface shape, the extracted ERG depends on the orientation of the height direction. Anyway, the global topology of the shape is captured by the ERG and it can be proven that even the Morse relationship among critical areas and Euler characteristics is still valid, except for meshes containing degenerate critical points such as the monkey-saddle [5]. The dependence of the ERG from the orientation makes it unsuitable for shape classification or recognition, which require unique models for shape description. Nevertheless, it is important to underline that the ERG furnishes a topological framework for constructing morphological skeletons of a given shape and it has been shown in [19] that the ERG can be effectively used to render the topology of a shape at a minimal level of detail.

Another important point concerns the density of the sweeping planes, which determines the scale of the shape features that will be. In order to correlate the distance of the sweeping planes to the feature size, a reasonable criterion is to use the minimum distance among vertices in the chosen direction to tune the distance of the cutting planes.

4. Re-meshing strategy for constraint insertion

The use of the triangulation constrained to contours makes it easy to efficiently compute the ERG of the mesh. To extend this approach to 3D, it is necessary to process the original mesh to ensure that contours are properly inserted and that vertices not belonging to the contours are removed from the mesh.

With regard to the contour extraction, plenty of algorithms can be found in the scientific literature, mainly designed for GIS applications and generally developed for 2.5D meshes [23][24][25]. In our context, since the contours have to be inserted as constraints, a method has been implemented which computes contours and inserts them in the mesh in a single step. In a second step, vertices that do not belong to the constraints are removed from the mesh. To start the process, the only parameter required is the number n of parallel planes to be intersected with the mesh T. For simplicity, indeed, the contours are always considered parallel to the XY plane of the coordinate system in which T is represented. A rotation of the whole mesh is performed for computing the slicing in any user-defined direction.

At each level, the intersection of the mesh with the corresponding plane is represented by a set of closed connected components. To avoid critical intersections, in particular with superior and inferior extremes of T, the distance among planes may be slightly adjusted locally to prevent these to occur. Therefore, inserting all the resulting connected components for each level, as explained in the following section solves the problem.

4.1. Contour computation and insertion

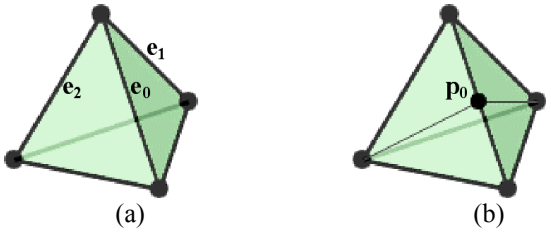
Since the triangulation represents a closed surface, it is possible to track and insert each contour level in a quite simple way, with basic operations involving only the adjacency relations stored in the data structure. More precisely, for each plane Π_i corresponding to a given elevation z_i a list L is defined which contains the edges of T that have a non-empty intersection with Π_i. Note that an edge with one of its vertices on the plane is considered to have a non-empty intersection with it. The first element of L is chosen as the first seed for computing a contour component. Once the tracking of this component is complete, the list L is scanned until the next seed edge is found, not yet been marked as processed. If such an edge exists, then it is used as the starting edge to insert a new connected component. The described process is repeated until the list L is empty.

Each connected component is inserted starting from the seed and tracing the contour vertex by vertex. Vertices may be either vertices of T already on the contour, or they have to be created every time there is an intersection between an edge and the plane. In this case, the mesh is locally updated with the insertion of the appropriate number of new triangles and edges, and the local adjacency relations are updated as well. From this process, a sequence $v_0, v_1, ..., v_k$ of vertices is obtained such that an edge of T exists for each pair (v_i, v_{i+1}), as well as for (v_k, v_0).

More precisely, at each step of the contour construction an active vertex, v, is defined and the next vertex is searched by analyzing the edges in $VT(v)$ which is the set of the triangles adjacent to v. Only one of these two situations may occur:

1. There exist one edge e in $VT(v)$ that entirely lies on the plane; the next vertex of the contour will be the other vertex of e, which becomes the current vertex; edges incident in v are marked as *visited*.
2. Otherwise, the next edge to be processed is the edge e in $VT(v)$, which intersects the plane, it is opposite to v and it is not marked as *visited.* In this case e is split at the intersection point p, where a new vertex v is created, and the local geometry and topology is updated; the new vertex and all its incident edges are marked *visited* (see Figure 6).

The process starts at the seed edge that is either in one of the two situations described above, or it has only one vertex on the plane. In this latter case, the intersection vertex will be defining the first active vertex.



(a) (b)

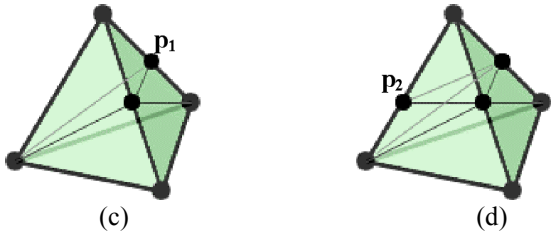


Figure 6. Inserting a connected component onto a tetrahedron.

The process is recursively repeated for each new extracted edge until it is impossible to get a next edge. The algorithm is described in Figure 6, with a simple tetrahedron as starting mesh (a). In the first step, the new vertex is inserted at p_0 and the next edge selected is e_1 (b). Also e_2 could have been selected as next edge, depending on the order with which triangles are stored in the VT(v) relation of the new vertex. In this case, the contour would be traced in the opposite direction. Then, the next new vertex is inserted at p_1 and the next edge processed is e_2 (c). Finally, the last new vertex is inserted and the process terminates because all the edges of the influence polygon are marked as visited (d).

4.2. Vertex removal

Once the contours have been inserted, it is necessary to remove two sets of vertices: vertices of the original triangulation that do not belong to section planes (original vertices), and those generated as a consequence of the sequential insertion of contours (redundant vertices). Redundant vertices are caused by the sequential insertion of contours in the mesh, as shown in Figure 7. Here, one triangle of the original triangulation is shown with two section planes which cross it. The result we would like to obtain is shown in the upper sequence (a), while the result of the sequential insertion of contours is depicted in (b).

In general, the problem of removing a vertex v from T corresponds to finding a re-triangulation of the *influence polygon* of v, that is, the polygon defined by the vertices in the star-neighbourhood of v. The removal of redundant vertices is trivial because they are all associated to flat influence polygons by definition. The vertex removal is therefore handled as in a simple 2D case: once the vertex and all its incident triangles have been removed, any polygon triangulation scheme can be used, provided that an edge is maintained on the section plane.

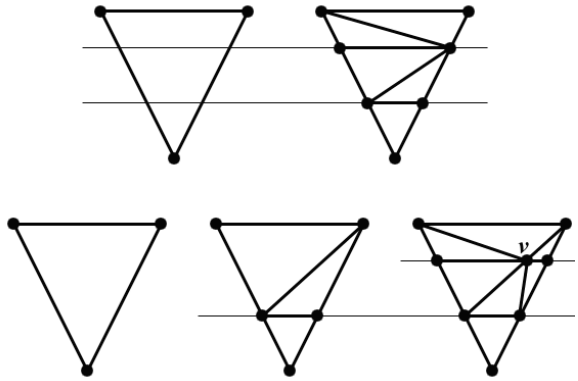


Figure 7. The generation of a redundant vertex.

The removal of vertices belonging to the original triangulation but not lying on the contours requires more care because the associated influence polygon may be not flat and this could cause self-intersection during the re-triangulation phase [22][26]. In this case, a support plane is used onto which the polygon is projected and triangulated as in the 2D case following the approach in [27]. We decided to use the so-called average plane of the star-neighbours of v as the projection plane, as defined in [27]. The re-triangulation is then based on the two Euler operators *MakeEdgeTriangle* and *MakeTriangle*; these insert new triangles in the mesh, which should not intersect with other elements of the triangulation. In Figure 8, three examples are shown in which the vertex v has to be removed, and the meshes are represented by polygons on the 2D plane. In (a) the polygon can be triangulated, while in cases (b) and (c) the polygon would self-intersect.

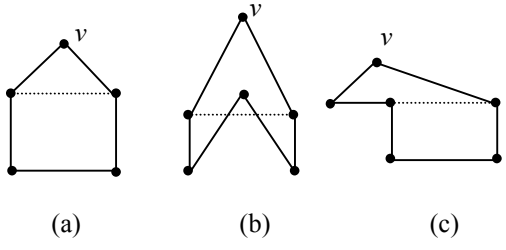


Figure 8. Examples of vertices to be removed.

Therefore, without additional hypotheses, an intersection test is necessary between each newly created element and the triangulation, to guarantee the validity of the model. The implementation of full self-intersection tests would require a number of operations **linear** in the number of mesh vertices. If n is the total number of vertices, the removal of m vertices requires $O(m*n)$ operations and, since in our context m is almost equivalent to n, the algorithm with self-intersection tests would require $O(n^2)$ operations. To keep the complexity within acceptable limits, we decided to implement a partial set of local tests, as described in the following.

First of all, given two triangles t_1 and t_2 incident at the edge e the *edge-angle* on e is the angle between the normal vectors n_1 and n_2 respectively of t_1 and t_2 (see Figure 9). This angle is computed for each non-boundary edge of newly created triangles to evaluate if they could intersect or not with existing triangles, especially to avoid folding of triangles over themselves.

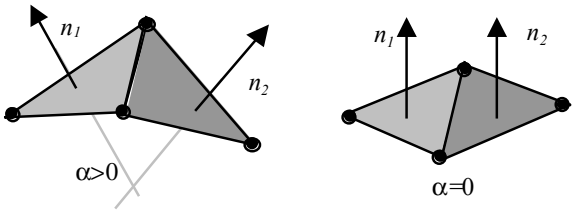


Figure 9. Example of edge-angles.

If one or more edges of a new triangle is above a given threshold value, the *existence condition* test for this new triangle is set to false. The maximum value for an *edge-angle* is π, which indicates that the model is surely not valid because the two triangles overlap. Therefore, it is possible to determine if the insertion of the new triangle causes the mesh to fold over itself, as in Figure 8(c), in a number of operations which is linear in the number of vertices of the influence polygon. As far as the cases of Figure 8(b) are concerned, the *existence conditions* are true even if the insertion causes self-intersection. In the context of this paper it is important to consider that, if the original surface is smooth enough, and if the number of section planes is great enough, we cannot find vertices to remove that fall in the case of Figure 8(b). However, this is only a conjecture suggested by experimental results. Future improvements will surely consider the definition of a formal framework to a priori establish the minimum number of contours to insert in order to grant the absence of vertices in such conditions. In the remainder we will consider this as an assumption.

More precisely, let $P = [e_1, \ldots, e_k]$ be the circular list, counter-clockwise sorted, of the edges of the influence polygon projected onto the support plane. Let us indicate with *RotateList* the move of the last element of P in the first position (e.g. the first rotation will result in $P = [e_k, e_1, \ldots, e_{k-1}]$). The following pseudo-code describes the algorithm used for removing vertices from T.

REMOVAL_LOOP {
IF $|P| > 3$ {
Let $e_a = (v_1, v_2)$, $e_b = (v_2, v_3)$ be the first two elements of P

IF (e_a,e_b form a concave angle with respect to the interior of P) and IF (*not* $\exists e \in E : e = (v_3, v_1)$ *or* $e = (v_1, v_3)$) THEN {
MakeEdgeTriangle($e_c = (v_3, v_1)$; $t = (e_a, e_b, e_c)$)

IF the *existence conditions* of t are not verified THEN *Undo MakeEdgeTriangle*(e_c,t) and *rotate the list* ELSE remove e_a and e_b from P and replace them with e_c.
}
ELSE *rotate the list*

IF the list P has been rotated $|P|$ times and no element has been added GO TO *UNREM* ELSE GO TO *REMOVAL_LOOP*.
}

IF $|P| = 3$ {
Let e_a, e_b, e_c be the three edges of P. *Maketriangle*($t = (e_a, e_b, e_c)$).

IF the *existence conditions* of t are not verified GO TO UNREM ELSE terminate.
}
}*END_NEXT*
UNREM Restore the initial state, the vertex can not be removed.

The *removability* of a vertex depends on the geometry of other mesh elements and, unfortunately, the sequential removal of vertices may also generate unstable configurations like the one in Figure 8(c). Due to the sequential removal, vertices, which were

impossible to remove, may become removable at the end of a removal loop. Therefore, the process continues with an optimization step and then another removal loop. Unstable configurations can be indeed improved by *edge-swapping*. The effect of a generic edge-swap operation is depicted in Figure 10, and for 2D triangulation, the validity of the model after a swap is guaranteed if the four angles of the quadrilateral formed by the two triangles (t_1 and t_2 in the figure) are less than 180°.

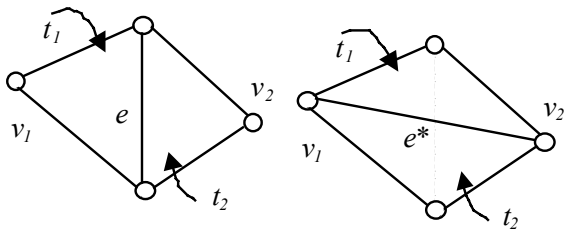
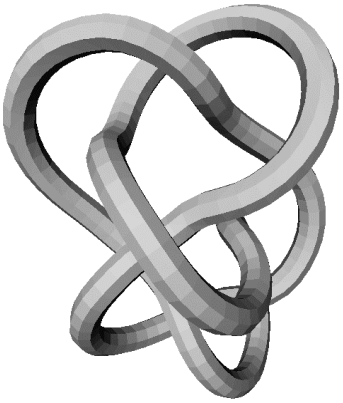
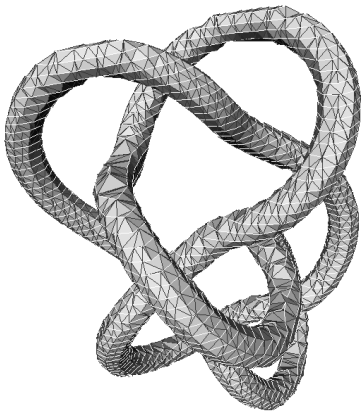


Figure 10. Edge swapping.

Unfortunately, the extension to the 3D can be applied only if the edge-angle of e is zero, in other words, the two triangles lie on the same plane. In all other cases, it is necessary to check, again, that the swap does not cause intersections among elements of the triangulation. A simplified test, based on the analysis of edge-angles, has been implemented in this case as well: after swapping, the edge-angles of e and of those of the quadrilateral edges are computed. If one ore more are greater than a fixed threshold value, then the initial state is restored. It is not difficult to prove that, if applied in two dimensions, this test implies the 2D case condition, and therefore it can be considered as an extension to the three-dimensional case.



(a)



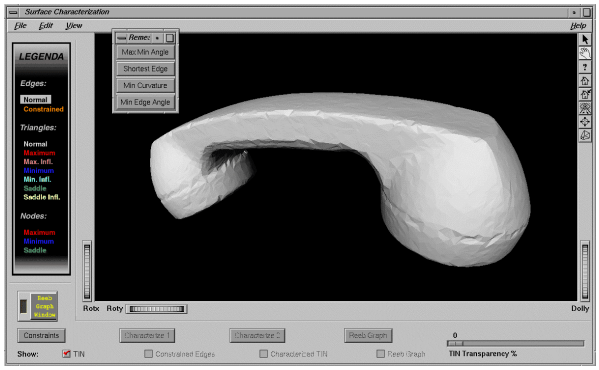
(b)

Figure 11. The contouring of a complex shape.

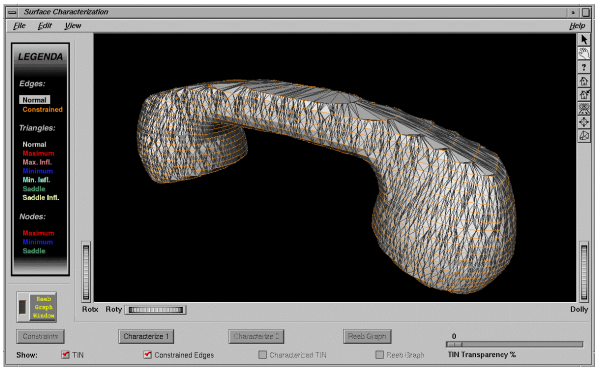
In Figure 11, an example of re-meshing of a rather complex shape is shown.

5. Results and discussion

Based on the described techniques, a prototype system has been implemented which performs the contour computation and insertion, and the Reeb graph extraction. The original mesh can be swept along any user-defined direction and with arbitrary number of sections. The whole process is depicted in Figure 12 and in the corresponding colour plate. In (a), the original mesh is shown. In (b), the mesh is shown after the contouring step, which makes it ready for the shape characterisation. The critical areas are depicted in the colour plate, with a colouring scheme, which associates to minimum areas the blue colour, to maximum the red, and to saddle the green one. Also, the influence zones are depicted in two different views.



(a)



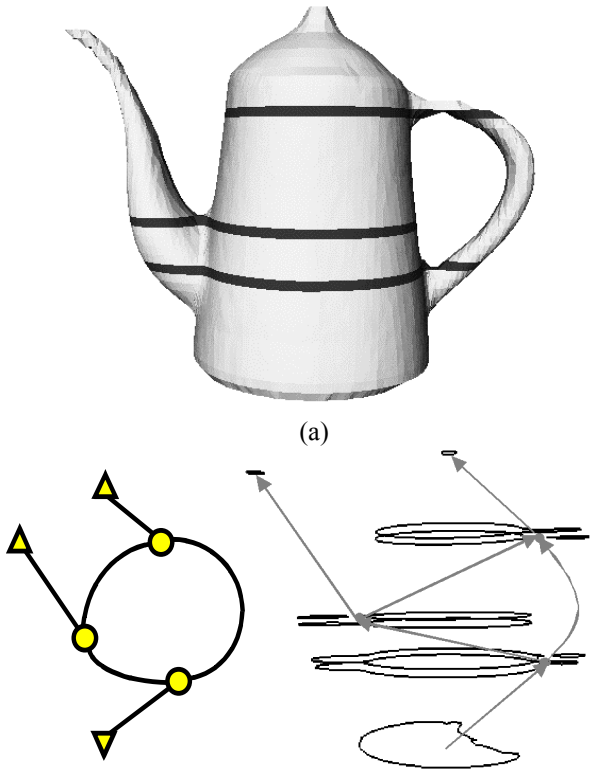
(b)

Figure 12. The ERG extraction process applied to the phone handset (see the corresponding colour plate).

The global complexity of the re-meshing algorithm can be given as a function of the maximum value between the number of vertices of the original triangulation, n, and the number of the constrained ones, m. Moreover, it can be seen that the number of edges and triangles are of the same order as the number of vertices. In the slicing step, the edge ordering pre-processing requires $O(\max(m,n\log n))$ operations. Then, $O(n\log n)$ operations are needed to sort the edges and $O(\max(m,n))$ is the number of intersection tests. Inserting the whole set of constraints requires $O(m)$ edge splits. Finally, the complexity of the vertex removal process is $O(m)$, which includes both the original vertices and the flat-region ones.

With regard to the computational complexity of the ERG extraction, the recognition of critical areas and the detection of influence zones require $O(t)$ operations, where t is the number of triangles. The complexity of the arc completion step is expressed by $O(t\log t)$, so that the total computational cost of the ERG extraction is $O(m\log m)$. Therefore, the whole process, starting from a generic triangulation, requires $O(max(m\log m, n\log n))$ operations.

Future developments of this method mainly concern the definition of a morphological structure to be merged with the ERG, which codes also the main morphological changes among contours. With reference to Figure 13, the shape of the original surface can be restored using contour blending techniques, especially if the ERG is augmented with more sections along the arcs that identify significant changes of the geometry of the contours. In this sense, we are currently working on the use of the ERG as the reference structure to compress and decompress shape models [19].



(a)

(b)

Figure 13. The ERG of the teapot: critical areas and corresponding influence zones (a) and the relations among critical sections (b).

6. Acknowledgements

The authors would like to thank all the people of the Computer Graphics Group at IMA-CNR, and the reviewers for their helpful comments.

7. References

[1] T. K. Dey, H. Edelsbrunner, and S. Guha, "Computational Topology", In *Advances in Discrete and Computational Geometry*, eds: Chazelle, B., Goodman, J. E., Pollack, R., *Contemporary Mathematics* 223, AMS, Providence, 1999, pp. 109-143.

[2] J.C.Hart, “Computational Topology for Shape Modelling”, *In Proceedings of Shape Modelling International ’99*, Univ. Aizu, Japan, 1999.

[3] J. Milnor, *Morse Theory*, Princeton University Press, New Jersey, 1963.

[4] V.Guillemin, and A.Pollack, A, *Differential Topology*, Englewood Cliffs, NJ: Prentice-Hall, 1974.

[5] S. Biasotti, B. Falcidieno, and M. Spagnuolo, “Extended Reeb Graphs for Surface Understanding and Description” to appear in *Proceedings of 9th Discrete Geometry for Computer Imagery conference*, *LCNS*, Springer Verlag, Uppsala, 2000.

[6] S. Biasotti, B. Falcidieno, and M. Spagnuolo, “Shape Abstraction Using Computational Topology Techniques”, in Proceedings of the Seventh Workshop GEO-7 organized by the IFIP Working Group 5.2, Parma, October 2000.

[7] H. Fuchs, Z. M. Kedem, and S.P. Uselton, "Optimal surface reconstruction from planar contours2. *Communications of the ACM*, 20(10):693-702, October 1977.

[8] Meyers D. "Reconstruction of surfaces from planar contours" Ph.D Dissertation, University of Washington, 1991.

[9] J.-M. Oliva, M. Perrin, S. Coquillart. "3D reconstruction of complex polyhedral shapes from contours using a semplified generalised Voronoi diagram", Proc. of Eurographics 96, J. Rossignac and F. Sillion (eds), Blackwell publishers, Vol. 15 (1996) N°3.

[10] P. Pentland, “Perceptual organization and representation of natural form”, *Artificial Intelligence*, Vol.28, 1986, pp. 293-331.

[11] L.R.Nackman, “Two-dimensional Critical Point Configuration Graphs”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 4, 1984, p. 442-450.

[12] C. Bajaj, D. R. Schikore, "Topology preserving data simplification with error bounds", Computer & Graphics, 22(1), 1998, pp. 3-12.

[13] U. Axen, and H. Edelsbrunner, “Auditory Morse Analysis of Triangulated Manifolds”, *Mathematical Visualization*, Springer Verlag, 1998, pp. 223-236.

[14] Z. J. Wood, M. Desbrun, P. Schröder, D. Breen, "Semi-Regular Mesh Extraction from Volumes", in *IEEE Visualization Conference*, Salt Lake City, October 2000.

[15] G. Reeb, “Sur les points singuliers d'une forme de Pfaff completement integrable ou d'une fonction numèrique”, *Comptes Rendus Acad. Sciences*, Paris, 1946, 222:847-849.

[16] Y. Shinagawa, and T. L. Kunii, “Constructing a Reeb graph automatically from cross sections”, *IEEE Computer Graphics and Applications*, 11(6), 1991, pp 44-51.

[17] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien, “Surface Coding Based on Morse Theory”, *IEEE Computer Graphics & Applications*, 1991, pp 66-78.

[18] G. Aumann, H. Ebner, and L. Tang, “Automatic derivation of skeleton lines from digitized contours”, ISPRS Journal of Photogrammetry and Remote Sensing, 46, 1991, pp. 259-268.

[19] S. Biasotti, M. Mortara, and M. Spagnuolo, “Surface Compression and Reconstruction using Reeb graphs and Shape Analysis”, *Spring Conference on Computer Graphics*, Bratislava, 2000, pp. 174-185.

[20] R. Engelking, and K. Sielucki, *Topology: a geometric approach*, Sigma series in Pure Mathematics, Volume 4, Heldermann Verlag, Berlin, 1992.

[21] T. Banchoff, "Critical points and curvature for embedded polyedral surfaces", *American Mathematical Monthly*, 77:475-485, 1970.

[22] P. Cignoni, C. Montani, R. Scopigno “A Comparison of Mesh Simplification Algorithms”, Computer & Graphics, 22(1):37-54, 1998

[23] C.L. Bajaj, V. Pascucci, D.R. Schikore, “Fast isocontouring for improved interactivity”, in *Proceedings 1996 SPIE Symposium on Volume Visualization*, San Francisco, 1996, pp.39-46.

[24] Y. Livnat, H.W. Shen, C.R. Johnson, “A Near optimal isosurface extraction algorithm using the span space”, in *IEEE Transactions on Visualization and Computer Graphics*, 2, 1996, pp.73-84.

[25] M. Van Kreveld, “On Quality paths on polyhedral terrains”, in *Proceedings IGIS’94: Geographic Information Systems, Nievergelt,* J., Roos, T., Schack, H.J., Widmayer, P. (Eds.), *Lecture Notes in Computer Science*, 884, Springer-Verlag, 1994, pp.113-122.

[26] Alan D. Kalvin, Russell H. Taylor, “Superfaces: Polygonal Mesh Simplification with Bounded Error”, in *IEEE Computer Graphics and Applications,* vol 16, no. 3, 1996.

[27] W. J. Schroeder, J. A. Zarge, W. E. Lorensen , “Decimation of Triangle Meshes”, in *Computer Graphics (SIGGRAPH ’92 proceedings)*, vol. 26, no. 2, 1992.

[28] G. Turk “Re-Tiling Polygonal Surfaces”, in *Computer Graphics (SIGGRAPH ’92 proceedings)*, vol. 25, 1992.

[29] G. Barequet, M. Dickerson, D. Eppstein, “On triangulating three-dimensional polygons”, in *Computational Geometry*, 10 (1998) pp. 155-170.