Curvature Tensor Based Triangle Mesh Segmentation with Boundary Rectification

Lavoué Guillaume
LIRIS FRE 2672 CNRS
43, Bd du 11 novembre
69622 Villeurbanne Cedex, France
glavoue@liris.cnrs.fr

Dupont Florent
LIRIS FRE 2672 CNRS
43, Bd du 11 novembre
69622 Villeurbanne Cedex, France
fdupont@liris.cnrs.fr

Baskurt Atilla
LIRIS FRE 2672 CNRS
43, Bd du 11 novembre
69622 Villeurbanne Cedex, France
abaskurt@liris.cnrs.fr

Abstract

This paper presents a new and efficient algorithm for the decomposition of 3D arbitrary triangle meshes into surface patches. The algorithm is based on the curvature tensor field analysis and presents two distinct complementary steps: a region based segmentation, which is an improvement of that presented in [1] and which decomposes the object into known and near constant curvature patches, and a boundary rectification based on curvature tensor directions, which corrects boundaries by suppressing their artifacts or discontinuities. Experiments were conducted on various models including both CAD and natural objects, results are satisfactory. Resulting segmented patches, by virtue of their properties (known curvature, clean boundaries) are particularly adapted to computer graphics tasks like parametric or subdivision surface fitting in an adaptive compression objective.

Keywords: Segmentation, 3D-mesh, Curvature tensor, Classification, Region growing, Region merging, Boundaries, CAD.

1. Introduction

Recent advances in the field of computer graphics (tools for acquisition, modelers, graphics hardware, etc…) have contributed to an amazing growth in the amount of 3D-models created and stored. With the expansion of the Internet, the need for transmission of these 3D-contents is more and more acute, this problematic results in the research of adaptive and multi-resolution compression methods, particularly for 3D-meshes which is the most widespread representation for 3D-objects.

In this context, the decomposition of 3D-objects, into surface patches, becomes attractive since it simplifies compression complexity and because it brings adaptiveness to algorithms. Within this framework, we present a curvature tensor based triangle mesh segmentation method, particularly adapted to optimized triangulated CAD objects, which decomposes a 3D-mesh into known and near constant curvature regions with clean and smooth boundaries. Resulting patches are particularly adapted to computer graphics tasks such as subdivision or parametric surface fitting in an adaptive compression objective.
Section 2 details the related work about mesh segmentation, whereas the overview of our method is presented in section 3. Sections 4 and 5 deal with the two distinct steps of our method: the region segmentation and the boundary rectification.

2. Related work

Only few studies concern triangle mesh segmentation. Garland et al. [2] present a face clustering of which aim is to approximate an object with planar elements; this algorithm is especially adapted for radiosity or simplification. Several approaches use discrete curvature analysis combined with the Watershed algorithm [3][4][5]. In the same way Zhang et al. [6] use the sign of the Gaussian curvature to mark boundaries, and process a part decomposition. These approaches extract only regions surrounded by high curvature boundaries and fail to distinguish simple curvature transitions. Lavoué et al. [1] present a classification based method which allows to detect these transitions; the first part of this paper is an improvement of this work. In a different way, Li et al. [7] use skeletonization to obtain nice segmentation results but their method induces a smoothing effect which can make disappear certain features.

Most of these cited approaches have a major shortcoming: the boundaries between patches are not correctly handled because they represent a minor problematic in these algorithms. As a result, either they are fuzzy (because only vertices are considered) [4][6], or they are jagged and present artifacts [1][3][5], or they are too straight and do not fit to the model [7]. Only Katz et al. [8] specifically handle the boundaries by using a fuzzy decomposition. Their method, based on geodesic distances and convexity, is well adapted for natural objects; but considering CAD object, their decomposition is not keen enough to be adapted to our surface fitting objective.

3. Method overview

We present a decomposition algorithm of arbitrary triangle meshes into known and almost constant curvature surface patches with clean and smooth boundaries. We address particularly the problem of CAD parts. Our approach is based on two steps:

A curvature based region segmentation: firstly, a pre-processing step identifies sharp edges and vertices (see Section 4.1). This information is necessary for the continuation of the algorithm, particularly in the case of optimally triangulated meshes. Then the curvature tensor is calculated for each vertex according to the work of Cohen-Steiner et al. [9]. Then vertices are classified into clusters (see Section 4.2), according to their principal curvatures values *Kmin* and *Kmax*. A region growing algorithm is then processed (see Section 4.3) assembling triangles into connected labelled regions according to vertex clusters. Finally a region adjacency graph is processed and reduced in order to merge similar regions (see Section 4.4) according to several criteria (curvature similarity, size and common perimeter).

A boundary rectification: firstly, boundary edges are extracted from the previous region segmentation step. Then for each of them, a *boundary score* is processed (see section 5.2) which notifies a degree of correctness. According to this score, estimated correct boundary edges are marked and are used in a contour tracking algorithm (see section 5.3) to complete the final correct boundaries of the object.

4. The region segmentation process

4.1. Sharp features detection.

Our segmentation algorithm is based on the analysis of the curvature of each vertex. Prior to start the algorithm we must detect and take into account sharp edges, especially for CAD object. Indeed even if, in practice, a curvature value is associated to sharp edges, the curvature is not theoretically defined on these features. We cannot consider a sharp edge like any other high curvature edge; it defines only a boundary and not a region. That is why we process a sharp features detection. A *sharp edge* is defined as follow: an edge shared by two triangles whose normal vectors make an angle higher that a given threshold. Vertices that belong to a *sharp edge* are considered as *sharp vertices* (but an edge shared by two *sharp vertices* is not necessarily a *sharp edge*).

This *sharp features* detection is useful within the region growing process (see section 4.3) and as a pre-processing step to process a mesh enrichment on bad tessellated objects, particularly optimized triangulated CAD objects with contain a very small triangle number. For each triangle associated with three *sharp vertices*, we could not reasonably evaluate its curvature or associate it with a region; it ties up with the "*no hard boundary*" problematic raised by Razdan and Bae [5]. Therefore we subdivide these *sharp triangles* by adding a new vertex at the center (see Fig.6). The region segmentation is thus applied on this modified mesh and added vertices are removed at the end of the algorithm.

4.2. Vertex classification

Vertices of the mesh are classified according to their principal curvatures *kmin* and *kmax*. Moreover the boundary rectification process (see section 5) needs principal curvature directions *dmin* and *dmax*, thus we have to calculate this information for each vertex of the input mesh.

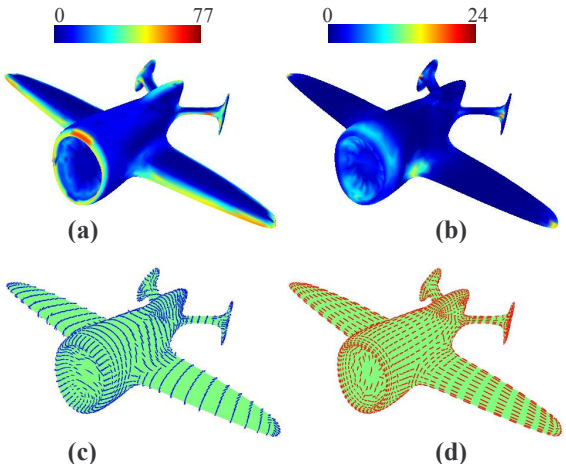

Figure 1. Curvature fields for the 3D object "Plane". (a) *Kmax*, (b) *Kmin* (absolute value), (c) *dmax*, (d) *dmin*.

4.2.1. Discrete curvature estimation. A triangle mesh is a piecewise linear surface, thus the calculation of its curvature is not trivial. Several authors have proposed

different evaluation procedures for curvature tensor estimation [9] [10] [11].

We have implemented the work of Cohen-Steiner et al. [9], based on the Normal Cycle. This estimation procedure relies on solid theoretical foundations and convergence properties. Moreover the tensor can be averaged over an arbitrary geodesic region, like in [12], therefore it is independent of the sampling and we have the possibility to filter noisy objects or consider only a queried size of details extraction for the segmentation method. For each vertex, the curvature tensor is calculated and the principal curvatures values *kmin*, *kmax* and directions *dmin*, *dmax* are extracted. They correspond respectively to the eigenvalues and eigenvectors of the curvature tensor, with switched order (the eigenvector associated with *kmin* is *dmax* and vice versa).

Fig.1 presents samples of these fields for the "Plane" object. On the edges of the wings, we have a high maximum curvature, whereas *kmin* is null, it is a parabolic region. *Kmin* is positive on elliptic regions, like at the end of the wings, and negative in hyperbolic regions like at the joints between the wings and the body of the plane. We have represented the absolute value of *kmin* on the figure because its sign has no importance in our algorithm. The principal curvature directions have signification only on anisotropic regions (elliptic, parabolic and hyperbolic) where they represent lines of curvature of the object. On isotropic regions (spherical, planar), they do not carry any information.

4.2.2. Curvature classification. Vertices are classified according to the values of their principal curvatures *Kmin* and *Kmax* (see Fig.2), associated with the Euclidian distance (in the curvature space). This classification is independent of the spatial disposition of the vertices. More complex and complete comparative measures exist between two tensors [13] [14] but for our purpose we just need to consider a basic curvature information and not complex tensor features like shape or orientation. Moreover *Kmin* and *Kmax* carry complementary information. *Kmin* can be negative, but we consider only its absolute value, it is not necessary to differentiate positive and negative values in our classification. The clustering is done via a K-Means algorithm (a usual unsupervised fast classification method) [15], completed by a cluster regularization (merging of small or similar clusters).

At the end of the algorithm each vertex is associated with a Cluster C_i and an associated classified curvature value c_i (c_i is in fact a two scalars vector which contains classified values for *Kmin* and *Kmax*). The number of clusters K, in the curvature space, is fixed by the user, but is not critical for the final segmentation result because of the region growing and merging steps. Fig.2 shows the vertex classification process applied to the "Plane" object (2506 vertices). The number of clusters in the curvature space was fixed to 5 for this example (clusters colors are yellow, orange, blue, dark blue and green).

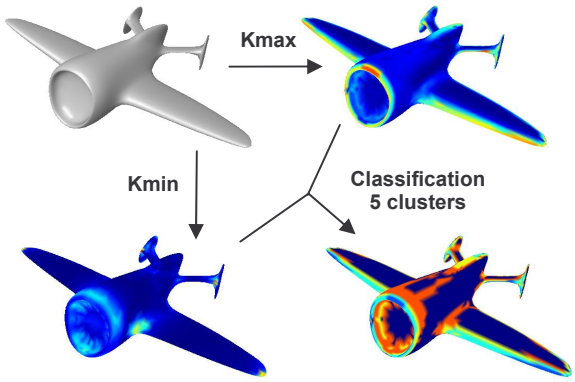

Figure 2. Vertex classification of the "Plane" mesh in 5 curvature clusters.

4.3. The region growing process

Once vertices have been classified, we want to recover triangle regions with similar curvature. This transmission of the curvature information from vertices to triangles is not a trivial operation. A triangle growing and labeling operation is performed as follows: for each triangle of which curvature is completely defined (*seed triangle*), a new region is created, labeled and extended. This process is repeated for every other *seed triangle* not yet labeled.

4.3.1. The *seed triangle* determination. There exist three situations where a triangle is considered as a *seed*:

- Its three vertices belong to the same cluster C_i, thus the curvature value c_i of this cluster is assigned to the corresponding created region.
- It is composed with two vertices from the same cluster C_i and a *sharp* one. Thus c_i is assigned to the created region.
- It contains two *sharp vertices*, thus the curvature value c_i of the third vertex is assigned to the created region.

In every other cases, we cannot assign a curvature value to the triangle, thus we cannot consider it as a *seed* to grow a region.

4.3.2. The growing mechanism. When a *seed triangle* is encountered, a new region is created, containing this triangle, associated to a new label L and a curvature value c_L.

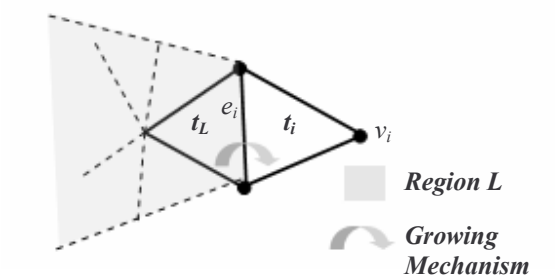

Figure 3. Considered features for the region growing process.

Then a recursive process extends this region (see Fig.3): for each triangle t_L belonging to the region, for each non sharp edge e_i of this triangle, we consider the associated neighboring triangle t_i and its opposite vertex v_i. If v_i is a *sharp vertex* or if it has the c_L curvature value, thus the considered triangle is integrated to the region. This process is repeated for every other triangle marked as *seed* and not yet labeled. With this process, it remains, sometimes, not labeled triangles at the end of the algorithm. A simple crack filling process fit these holes by integrate these triangles to the most represented region of their neighborhoods. Fig.4 shows the region growing process for the Fandisk object, starting from a 18 clusters vertex classification. The region growing extracts 128 connected regions (regions colors are randomly chosen).

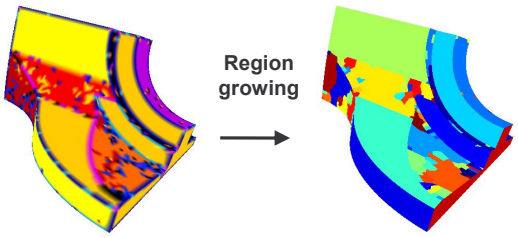

Figure 4. The region growing process for the "Fandisk" mesh (Regions colors are randomly chosen).

4.4. The region merging process

The region merging process aims to:

- Reduce the over-segmentation resulting from the growing step.
- Suppress the algorithm dependency to the number of curvature clusters issued from the K-Means vertex classification.

The reader can refer to [1] for a more detailed presentation of this algorithm which is just summarized here.

4.4.1. General algorithm. Once connected regions have been extracted by the region growing algorithm, a region adjacency graph is processed. Each node represents a connected region (i.e. a connected subset of the mesh), and each edge represents an adjacency between two regions. Edges are evaluated by a similarity distance D_{ij} between the two corresponding regions. The reduction of the graph is then processed: at each iteration the smallest edge of the graph is eliminated, thus the corresponding regions are merged; then the graph is updated. This graph reduction stops when the number of regions reaches a queried number chosen by the user, or when the weight of the smallest edge is larger than a given threshold.

4.4.2. Region distance measurement. The distance D_{ij} used in our method is equal to the curvature distance DC_{ij}, between the two corresponding regions R_i and R_j weighted by two coefficients: N_{ij}, which measures the nesting between the two corresponding regions and S_{ij} of which aim is to eliminate the smallest regions.

$$D_{ij} = DC_{ij} \times N_{ij} \times S_{ij} \qquad (1)$$

Each coefficient is detailed in the following paragraphs.

The curvature distance DC_{ij} is processed using the curvature values c_i and c_j of the two corresponding regions and the curvature value c_{ij} of their boundary.

$$DC_{ij} = \|c_i - c_{ij}\| + \|c_j - c_{ij}\| \qquad (2)$$

c_i and c_j come from the region growing step. c_{ij} is the average of the vertices curvatures on the boundary between the two regions.

The N_{ij} coefficient measures the nesting between the two corresponding regions by considering their respective perimeters P_i and P_j, and their common perimeter P_{ij}.

$$N_{ij} = \frac{\min(P_i, P_j)}{P_{ij}} \qquad (3)$$

Regions with a large common border are more likely to belong to the same "meaningful" part of the object, thus their similarity distance is reduced.

The S_{ij} coefficient purpose is to accelerate the fusion of the smallest regions according to the values of their areas A_i.

$$S_{ij} = \begin{cases} \varepsilon & if\,(A_i < A_{\min}\ or\ A_j < A_{\min}) \\ 1 & else \end{cases} \qquad (4)$$

If the area A_i of a region is smaller than a threshold A_{min}, thus its distance with its neighbouring regions is reduced by the ε coefficient, fixed near 0.

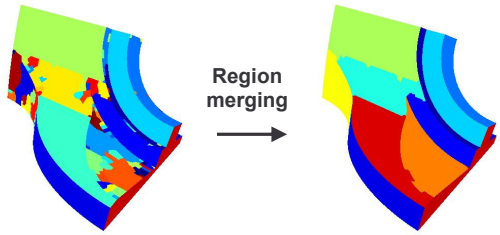

Figure 5. The region merging process for the "Fandisk" mesh.

Fig.5 shows the merging process. The initial pre-segmented object was obtained after the classification step in the curvature space (18 curvature clusters), and after the region growing step (see Fig.4). It contains 128 connected spatial regions. After the merging process, the final region number is 25. The merging threshold was fixed to 5.

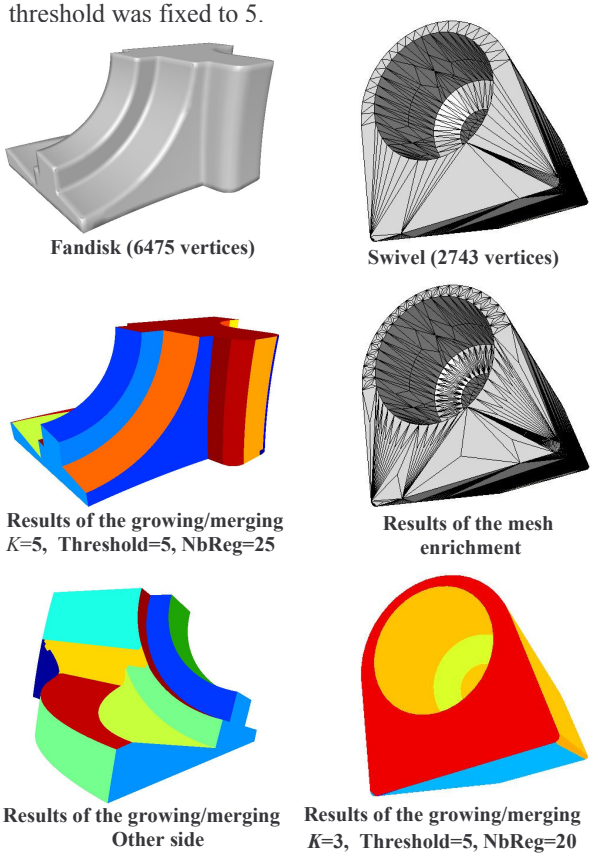

Figure 6. Segmentation of "Fandisk" and "Swivel" objects.

4.5. Experiments and results

Our segmentation method was tested on several different objects. Examples are given for two objects from different nature: a mechanical highly tesselated object (Fandisk) and an optimized triangulated CAD object (Swivel). Results are shown on Fig.6. For the "Fandisk" object we obtain patches with almost constant curvature. Our method allows detecting curvature transition or inflexion points and not only regions separated by high curvature boundaries, or sharp boundaries, like traditional watershed methods. Even for the bad tessellated "Swivel" object, we obtain good results after the enrichment of detected *sharp triangles.*

5. The boundary rectification process

5.1. Objective

Our purpose is to obtain clean patches with constant curvature in a subdivision or parametric surface fitting objective. Our region segmentation method extracts near constant curvature, topologically simple patches from the 3D-objects, and gives good qualitative results in terms of general shape and disposition of the segmented regions. Nevertheless, boundaries of the extracted patches are often jagged, like for most of the existing segmentation methods and present artefacts particularly when we consider a high number of curvature clusters (like in Fig.5). Fig.7 presents an example of discontinuity, blue and yellow regions in the red ellipse are not correct, their boundary is not straight. In this context, the objective of the boundary rectification process is to suppress these artefacts, in order to obtain clean and smooth boundaries corresponding to real natural boundaries of the object.
The rectification method is composed of two principal steps: firstly, segmented regions boundary edges are extracted and for each of them a *correctness* score is processed. Then, starting from the estimated *correct* boundary edges, the final boundaries of the patches are completed using a contour tracking algorithm.

5.2. The *Boundary Score* definition

The goal of this score is to define a notion of correctness for each boundary edge extracted from the previous region segmentation. For this purpose, we consider the principal curvature directions *dmin* and *dmax* (see section 4.2.1) which define the lines of curvature of the object. Indeed, they represent pivotal information in the geometry description [12]. The curvature tensors at the natural boundaries of an object tend to be very anisotropic with a maximum direction following the curvature transition and therefore orthogonal to the boundaries. Thus the boundaries will tend to be parallel to the lines of minimum curvature

(see Fig.10.b). Therefore the angle between a boundary edge and its vertices minimum curvature directions can represent a good evaluation of its "correctness".

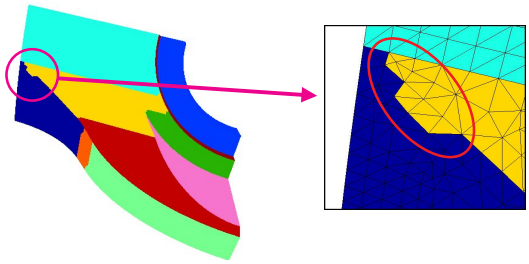

Figure 7. Zoom on an artefact for the segmented "Fandisk" object.

The boundary score *S*, calculated for an edge e_i, is:

$$S(e_i) = Sa(e_i) + \omega_c \times Sc(e_i) \qquad (5)$$

ω_c is a weighting coefficient, which is fixed to 1 in our examples (S_a and *Sc* are normalized).

The *angle score* S_a considers the angles ϑmin_{i1} and ϑmin_{i2} (see Fig.8) between the edge e_i and its vertices minimum directions.

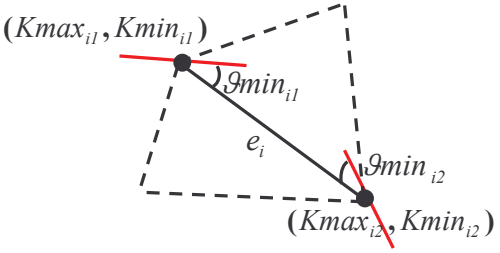

Figure 8. Elements taken into account for the calculation of the *Boundary Score* of the edge e_i.

The score also consider the angles ϑmax_{i1} and ϑmax_{i2} between the edge e_i and its vertices maximum directions, weighted by the values of the principal curvatures *Kmin* and *Kmax* in order to take into account isotropic region. Thus the angle score *Sa* is processed according to the following equation (6):

$$Sa(e_i) = + \frac{\left(\vartheta min_{i1} \times Kmax_{i1} + \vartheta max_{i1} \times Kmin_{i1}\right)}{Kmax_{i1} + Kmin_{i1}} + \frac{\left(\vartheta min_{i2} \times Kmax_{i2} + \vartheta max_{i2} \times Kmin_{i2}\right)}{Kmax_{i2} + Kmin_{i2}} \qquad (6)$$

with $\vartheta min_{i1}, \vartheta min_{i2}$ and $\vartheta max_{i1}, \vartheta max_{i2}$ the respective angles of the considered edge e_i with the minimum curvature directions of its vertices and their maximum curvature directions.

$Kmin_{i1}, Kmin_{i2}$ and $Kmax_{i1}, Kmax_{i2}$ are the respective values of minimum curvatures and maximum curvatures of the vertices of the edge e_i.

The *curvature score Sc* corresponds to a normalized curvature difference between curvature values of the two vertices of the edge. If curvatures of the edge's vertices are very different, thus the edge must not be considered as a correct boundary. *Sc* is defined by the following equation (7):

$$Sc(e_i) = \frac{\left|Kmin_{i2} - Kmin_{i1}\right| + \left|Kmax_{i2} - Kmax_{i1}\right|}{\mathbf{max}(Kmin_{i2}, Kmin_{i1}) + \mathbf{max}(Kmax_{i2}, Kmax_{i1})} \qquad (7)$$

5.3. Algorithm

The rectification algorithm is composed of two steps: the marking of the correct boundary edges coming from the region segmentation and the contour tracking to complete final boundaries.

5.3.1. Correct boundary marking. For every boundary edges coming from the region segmentation step, the *Boundary Score* previously defined is processed. Then, a threshold *ST* is fixed; for each edge, if its *Boundary Score* is below *ST*, the edge is considered as a *correct boundary edge* (*CBE*), else the edge is no more considerate. Fig.10.c and Fig.11.c show this marking process, starting from the region segmentation (see Fig.10.a, Fig.11.b), *CBEs* are represented in green, and others in red.

5.3.2. Contour tracking. The second step of the rectification algorithm is the contour tracking. Once *CBEs* have been extracted, they form pieces of boundary contours; our purpose is to complete these contours to obtain a set of closed contours corresponding to the final regions boundaries. For each not closed boundary contour, we extract the edges potentially being able to complete it (we call them *potential edges*). They are edges adjacent to one *CBE* at the extremity of an open contour. Fig.9.a shows a piece of contour formed by two *CBEs* (in black), with associated *potential edges* (*PE*) (in dotted black) which are candidates to complete the open contour. Then, each potential edge is associated with a weight *P* which will determine its possibilities to be integrated to the contour; the smallest is this weight, the more the edge has possibilities to be considered as a *CBE*.

The weight *P* of a *potential edge* depends of its score $S(e_i)$ but also of its angle $\vartheta(e_i, e_{CBE})$ with its

neighboring *CBE*, because, we try to limit the deviation of the boundary.

$$P(e_i) = S(e_i) + \omega_\vartheta \times \vartheta(e_i, e_{CBE}) \qquad (8)$$

ω_ϑ is a weighting coefficient, it is fixed to 1 in our examples.

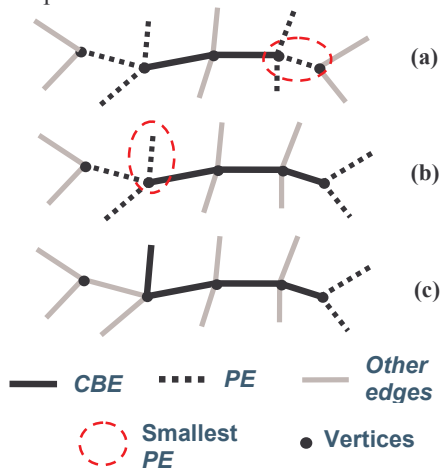

Figure 9. Three steps (a,b,c) of the boundary tracking algorithm, with associated positions of correct boundary edges (*CBE*), potential edges (*PE*) and smallest potential edges, at each iteration.

Once each potential edge has been valuated, we organize them into a sorted list. Then the contour tracking algorithm starts; its mechanism is the following: once we have the *potential edges* (*PE*) sorted list, the *PE* associated with the lowest weight *P* is extracted and integrated to the considered boundary contour, and therefore this *PE* becomes a *CBE*. Then the list is updated (the *PEs* are redistributed) and the list reduction continues until every boundary contour is closed. Fig.9 presents three iterations of the contour tracking algorithm. In Fig.9.a, there are two *CBEs* which form an open contour (in black), thus there are six *PEs* candidates to complete the contour (in dotted black). The *PE* inside the red ellipse is considered as the one with the smallest weight *P*, thus at the next iteration it is extracted and integrated to the contour (see Fig.9.b). The position and number of the *PEs* is then updated. The process continues in Fig.9.c, with another *PE* integrated to the contour.

5.4. Experiments and results

The rectification method is especially adapted to CAD or mechanical objects, where there exist real defined smooth boundaries. On natural or organic objects the fact of rectifying boundaries does not have a real signification since even a human hand could not trace precise and clear boundaries.

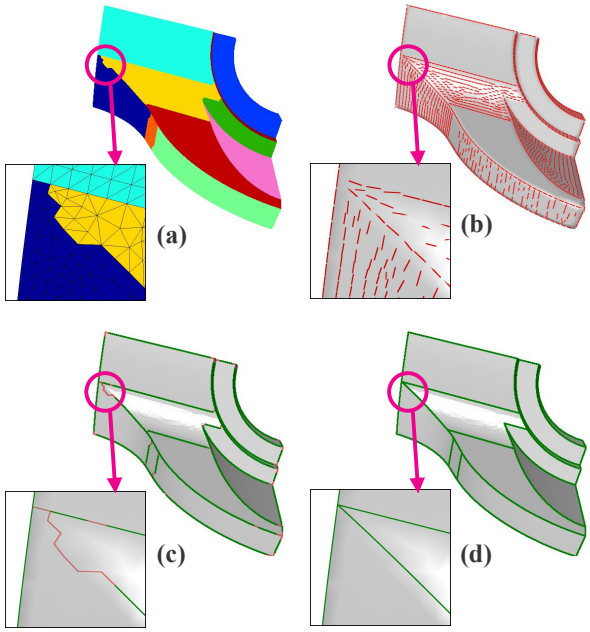

Figure 10. The different steps of the Boundary Rectification for the "Fandisk" object with a zoom on an artifact correction. (a) Segmented object. (b) Minimum curvature directions. (c) Correct Boundary Edges extraction and marking. (d) Corrected boundaries after the contour tracking.

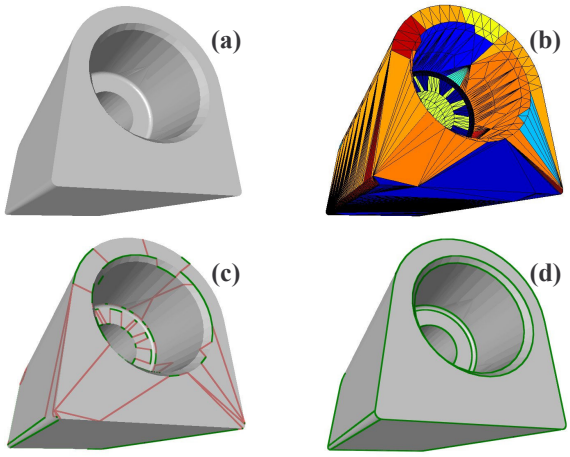

Figure 11. The different steps of the Boundary Rectification for an artificially bad segmented CAD object. (a) Original object. (b) Bad segmented object. (c) Correct Boundary Edges extraction and marking. (d) Corrected boundaries after the contour tracking.

We have tested our rectification method on various models issued from our region segmentation method. Fig.10 presents results for "Fandisk". Artefacts coming

from the region segmentation are all suppressed; we obtain surface patches with very clean and smooth boundaries, adapted for tasks like parametric or subdivision surface fitting. We have also conducted tests on artificially bad segmented objects, in order to see if the rectification method could repair a bad segmentation and not only suppress some small imperfections. Fig.11 shows results on a bad segmented object. We can observe that bad boundary edges are eliminated whereas correct ones are correctly extracted and completed to give a very correct set of surface patches. Even with very few correct boundary edges, final boundaries of the object are well extracted.

6. Conclusion

This paper presents an original segmentation method to decompose a 3D-mesh into near constant curvature surface patches with clean boundaries.
The simple and efficient curvature classification detects any curvature transition and thus allows segmenting the object into known and near constant curvature regions and not just cutting the object along its hard edges. The triangle growing process well transmits regions information from vertices to triangles even for bad tessellated CAD objects.
Our original boundary rectification method based on curvature tensor orientation, allows suppressing boundaries defects commonly produced by most of the segmentation algorithms, even if they are important. We obtain, in the case of CAD or mechanical objects, the real natural boundaries corresponding to an intuitive hand made segmentation of the object. This method is independent of the previous region segmentation and can be used as a post process of a hard edges detection, for example to complete hard edges contours of an object.
About perspectives, we plan to consider variance and histogram distribution of curvature, in order to improve the curvature classification method, and also to be able to automatically process the region merging threshold which remains a user defined parameter of our method. This work is part of a larger compression process. The objective is to fit the segmented regions with subdivision or parametric surfaces, in order to obtain the object in the form of a set of ”light” patches, which will allow adaptive and scalable compression and transmission.

7. Acknowledgments

This work is supported by the French Research Ministry and the RNRT (Réseau National de Recherche en Télécommunications) within the framework of the Semantic-3D national project (http://www.semantic-3d.net).

8. References

[1] G. Lavoue, F. Dupont and A. Baskurt, "Constant Curvature Region Decomposition of 3D-Meshes by a Mixed Approach Vertex-Triangle.", *Journal of WSCG*, 2004, vol. 12, no. 2, pp. 245-252.
[2] M. Garland, A. Willmott and P. Heckbert, "Hierarchical face clustering on polygonal surfaces.", *ACM Symposium on Interactive 3D Graphics*, 2001, pp. 49-58.
[3] A. Mangan and R. Whitaker, "Partitioning 3D Surface Meshes Using Watershed Segmentation", *IEEE Visualization and Computer Graphics*, 1999, vol. 5, no. 4, pp. 308-321.
[4] Y. Sun, D. Page, J. PAIK, A. Koschan and M. Abidi, "Triangle Mesh-Based Edge Detection And Its Application To Surface Segmentation And Adaptive Surface Smoothing", *IEEE International Conference on Image Processing*, NY, USA, 2002, Vol. 3, pp. 825-828.
[5] A. Razdan and M. Bae, "A hybrid approach to feature segmentation of triangle meshes", *Computer-Aided Design*, 2003, vol. 35, no. 9, pp. 783-789.
[6] Y. Zhang, J. PAIK, A. Koschan, M. Abidi and D. Gorsich, "A simple and efficient algorithm for part decomposition of 3D triangulated models based on curvature analysis", *IEEE International Conference on Image Processing*, Rochester, NY, USA, 2002, vol. 3, pp. 273-276.
[7] I. Li, T. Toon, T. Tan and Z. Huang, "Decomposing polygon meshes for interactive applications.", *symposium on Interactive 3D graphics*, 2001, pp. 35-42.
[8] S. Katz and A. Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts", *ACM Transactions on Graphics*, 2003, vol. 22, no. 3, pp. 954-961.
[9] D. Cohen-Steiner and J. Morvan, "Restricted delaunay triangulations and normal cycle", *19th Annu. ACM Sympos. Comput. Geom.*, 2003, pp. 237-246.
[10] M. Meyer, M. Desbrun, P. Schröder and H. Barr, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds.", *International Workshop on Visualization and Mathematics*, Berlin, Germany, 2002.
[11] G. Taubin, "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation", *Fifth International Conference on Computer Vision*, 1995, pp. 902-907.
[12] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy and M. Desbrun, "Anisotropic Polygonal Remeshing", *ACM Transactions on Graphics, SIGGRAPH '2003 Conference Proceedings*, 2003, vol. 22, no. 3, pp. 485-493.
[13] D. Alexander and J. Gee, "Elastic Matching of Diffusion Tensor Images", *Computer Vision and Image Understanding*, 2000, vol. 77, pp. 233-250.
[14] P. Basser and C. Pierpaoli, "Microstructural and Physiological Features of Tissues Elucidated by Quantitative Diffusion Tensor MRI", *Journal of Magnetic Resonance*, 1996, vol. 111, pp. 209-219.
[15] A. Gersho and R. Gray , *Vector Quantization and Signal Compression*, Boston, USA, Kluwer Academic Publishers, 1992.