ELSEVIER

Computer-Aided Design 37 (2005) 975–987

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

A new CAD mesh segmentation method, based on curvature tensor analysis

Guillaume Lavoué*, Florent Dupont, Atilla Baskurt

LIRIS FRE 2672 CNRS, 43, Bd du 11 novembre, 69622 Villeurbanne Cedex, France

Received 17 April 2004; received in revised form 7 September 2004; accepted 20 September 2004

Abstract

This paper presents a new and efficient algorithm for the decomposition of 3D arbitrary triangle meshes and particularly optimized triangulated CAD meshes. 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 by Lavoue et al. [Lavoue G, Dupont F, Baskurt A. Constant curvature region decomposition of 3D-meshes by a mixed approach vertex-triangle, J WSCG 2004;12(2):245–52] and which decomposes the object into near constant curvature patches, and a boundary rectification based on curvature tensor directions, which corrects boundaries by suppressing their artefacts or discontinuities. Experiments conducted on various models including both CAD and natural objects, show satisfactory results. Resulting segmented patches, by virtue of their properties (homogeneous 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

The context of this work is the Semantic-3D project (http://www.semantic-3d.net), supported by the French Research Ministry and the RNRT (Réseau National de Recherche en Télécommunications). The objective is the low bandwidth transmission of CAD triangulated models, coming from the car manufacturer Renault, with multi-resolution and adaptivity properties. The original NURBS information is not always available thus we cannot use it; the main reason is that many models are designed by subcontractors which provide only optimized triangulated meshes. In this context, a 3D compression algorithm is necessary but the optimized tessellation and the need of a low bandwidth transmission make this issue quite complex. The chosen approach is to convert the original triangulated object into a set of patches represented by subdivision surfaces and therefore associated with a smaller amount of data. This representation will bring a high compression rate adapted to a low bandwidth and to a multi-resolution displaying because of subdivision properties. This piece-wise subdivision surface approximation requires a prior decomposition of the meshes into adapted surface patches. 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 regions with homogeneous curvature and clean and smooth boundaries. Resulting patches are thus particularly adapted to subdivision surface fitting, on top of dealing with the reverse engineering problem. Section 2 details the related work about mesh segmentation, while 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

There has been a considerable research work relevant to the problem of 3D-object segmentation. However,

* Corresponding author. Tel.: +33 4724 48395; fax: +33 4724 31536.
E-mail addresses: glavoue@liris.cnrs.fr (G. Lavoué), fdupont@liris.cnrs.fr (F. Dupont), abaskurt@liris.cnrs.fr (A. Baskurt).

0010-4485//$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2004.09.001

the majority of these methods concern range images [2–4] or 3D point clouds [5,6]. Only few studies concern triangle meshes which is nevertheless the most widespread representation for 3D-objects. Garland et al. [7] present a face clustering of which aim is to approximate an object with planar elements; this algorithm is especially adapted for radiosity or simplification. Benko and Varady [8] consider a mesh decomposition approach specifically adapted for reverse engineering by applying a hierarchy of tests to recognize conventional engineering objects (extrusions, surfaces of revolution,…). Several approaches use discrete curvature analysis combined with the Watershed algorithm described by Serra [9] in the 2D image segmentation field. Mangan and Whitaker [10] generalize the Watershed method to arbitrary meshes, using the Gaussian curvature or the norm of covariance of adjacent triangle normal vectors at each mesh vertex as the height field. Sun et al. [11] use the Watershed with a new curvature measure based on the eigen analysis of the surface normal vector field in a geodesic window. More recently, Razdan and Bae [12] propose a hybrid method which combines the Watershed algorithm with the extraction of feature edges by the analysis of dihedral angles between faces. Zhang et al. [13] 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. [14] use skeletonization to obtain nice segmentation results, however 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) [11,13], or they are jagged and present artefacts [1,10,12], or they are too straight and do not fit to the model [14]. Only Katz et al. [15] specifically handle the boundaries by using a fuzzy decomposition. Their method, based on geodesic distances and convexity aims to extract high level sub-objects from a given model (for example, the arms of a body). Therefore, their method is not adapted to our task which is rather to extract quite low level sub-surfaces from a CAD object in our surface fitting objective.

3. Method overview

We present a decomposition algorithm of arbitrary triangle meshes into near constant curvature surface patches with clean and smooth boundaries. We address particularly the problem of CAD models, but natural objects are also considered. Our approach is based on two steps (see Fig. 1).

A curvature based region segmentation: firstly, a preprocessing 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. Secondly, the curvature tensor is calculated for each vertex according to the work of Cohen-Steiner et al. [16]. Then vertices are classified into clusters (see Section 4.2), according to their principal curvature 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 calculated (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.

The contributions introduced in this paper include:

- The use of a K-Means classification applied to vertex principal curvatures which allows a very fine region segmentation by detecting smooth curvature transitions and inflexion points and not only regions surrounded by high curvatures like other curvature based methods.
- The growing method extracts triangle regions from vertex curvature information, even for bad tessellated CAD objects, by taking into account sharp edges.

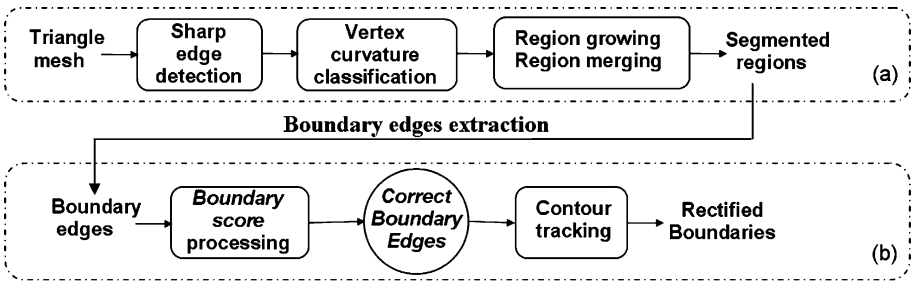

Fig. 1. The two steps of the method. (a) Constant curvature region segmentation. (b) Boundary rectification.

- The merging process is a non-trivial adaptation of an Image Processing method, taking into account common perimeters of the regions.
- The boundary rectification method is an original algorithm, using principal directions of curvature, which gives very good results by extracting quite smooth and clean boundaries.

4. The region segmentation process

4.1. Sharp feature detection

Our segmentation algorithm is based on the analysis of the curvature of each vertex. As a preliminary step, we must detect and take into account sharp edges, especially for CAD objects. 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 feature detection. A *sharp edge* is defined as follows: an edge shared by two triangles whose normal vectors make an angle higher than a given threshold. Vertices that belong to a sharp edge are considered as *sharp vertices* (nevertheless an edge shared by two sharp vertices is not necessarily a sharp edge).

This *sharp feature* 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. For each triangle associated with three sharp vertices, we cannot evaluate its curvature or associate it with a region; it ties up with the 'no hard boundary' problem risen by Razdan and Bae [12]. Therefore, we subdivide these *sharp triangles* by adding a new vertex at the centre (see Fig. 2). 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, as the boundary rectification process (see Section 5) needs principal curvature directions *dmin* and *dmax*, we have to calculate the curvature tensor for each vertex of the input mesh.

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 [16–18].

We have implemented the work of Cohen-Steiner et al. [16], based on the Normal Cycle. This estimation procedure has proven to be the most efficient and stable among the others and gives very satisfying results even for bad tessellated objects. It relies on solid theoretical foundations and convergence properties. Moreover, the tensor can be averaged over an arbitrary geodesic region, like in [19]; therefore it is independent of the sampling and it offers the possibility to filter noisy objects or to consider only a queried size of details extraction for the segmentation method.

For each vertex, the curvature tensor is calculated and the principal curvature values *Kmin*, *Kmax* and directions *dmin*, *dmax* are extracted. They correspond, respectively, to the eigenvalues and the eigenvectors of the curvature tensor, with switched order (the eigenvector associated with *Kmin* is *dmax* and vice versa).

Fig. 3 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. The figure displays the absolute value of *Kmin* on the figure (its sign is not taken into account in our algorithm). The principal curvature directions have significance only on anisotropic regions (elliptic, parabolic and hyperbolic) where they

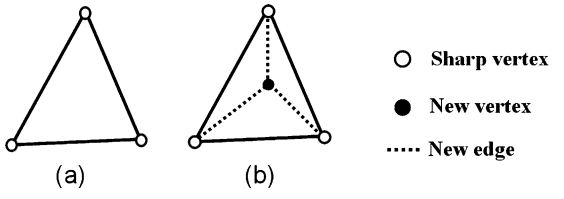

Fig. 2. The mesh enrichment process. (a) Triangle with three sharp vertices. (b) Associated subdivided triangle.

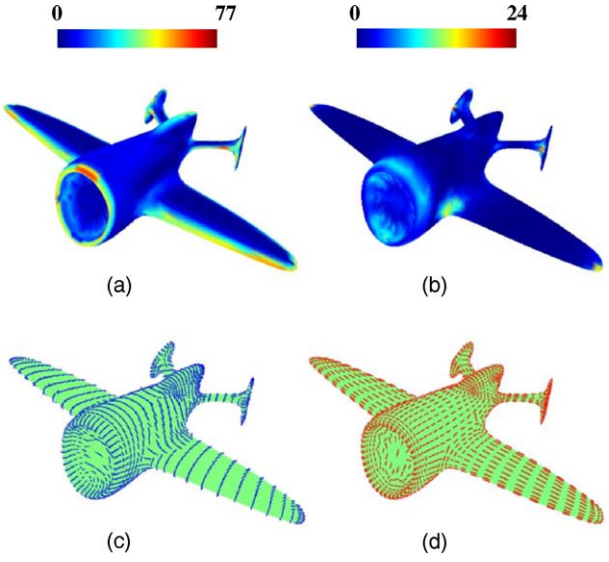

Fig. 3. Curvature fields for the 3D object Plane. (a) *Kmax*, (b) *Kmin* (absolute value), (c) *dmax*, (d) *dmin*.

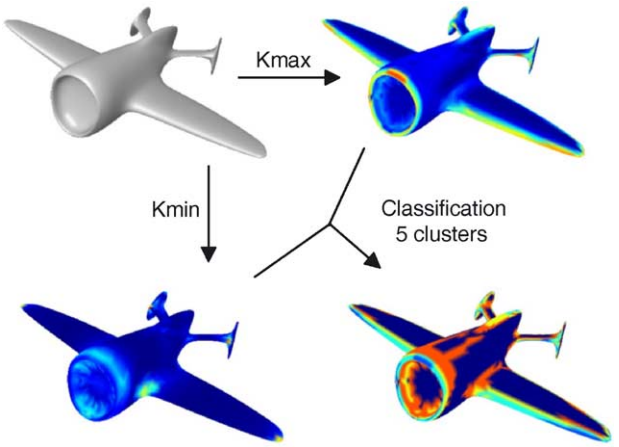

Fig. 4. Vertex classification of the Plane mesh in five curvature clusters.

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. 4), 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 [20,21], but for our purpose we just need to consider a basic curvature information and not complex tensor features like shape or orientation. Besides, *Kmin* and *Kmax* carry complementary information. *Kmin* can be negative, but we consider only its absolute value, as it is not necessary for us to differentiate elliptic (positive values) from hyperbolic (negative values) regions, which have the same visible curved aspect.

The clustering is done via a K-Means algorithm (a fast least-squares partitioning method) [22] allowing to divide vertices into K groups. At the end of the algorithm each vertex is associated to a Cluster C_i (among K) 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*) which is the centroid of the associated cluster. Starting from K initial seeds, randomly chosen, the algorithm iterates between two simple steps:

- Assign each vertex to the nearest seed.
- Compute cluster centroids and use them as new cluster seeds.

The number of clusters K, in the curvature space, is set by the user but is not critical for the final segmentation result because of the region growing and merging steps (see Section 4.3.2 and Table 2). The K-means algorithm is followed by a cluster regularization (merging of small or similar clusters) which gives K' final clusters. Fig. 4 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 (cluster colors are yellow, orange, blue, dark blue and green).

4.3. The region growing process

Once vertices have been classified, we aim at recovering triangle regions with similar curvature. We have a set of curvature clusters (groups of similar vertices in the curvature space), and we want to recover spatial regions (connected groups of triangles). This transmission of the curvature information from vertices to triangles is not a trivial operation. A triangle growing and labelling operation is performed as follows: for each triangle of which curvature is completely defined (*seed triangle*), a new region is created, labelled and extended. This process is repeated for every other seed triangle still unlabelled.

4.3.1. The seed triangle determination

There exist three situations where a triangle is considered as a *seed* (see Fig. 5):

- 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 (see Fig. 5a).
- It contains two sharp vertices, thus the curvature value c_i of the third vertex is assigned to the created region (see Fig. 5b).
- It is composed of two vertices from the same cluster C_i and a *sharp* one. Hence, c_i is assigned to the created region (see Fig. 5c).

In every other case (see Fig. 5d for example), we cannot assign a curvature value to the triangle, thus we cannot consider it as a seed to grow a region.

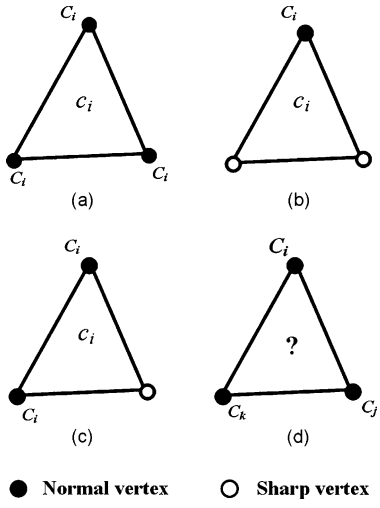

Fig. 5. The three seed triangle situations (a)–(c) and an undetermined triangle (d).

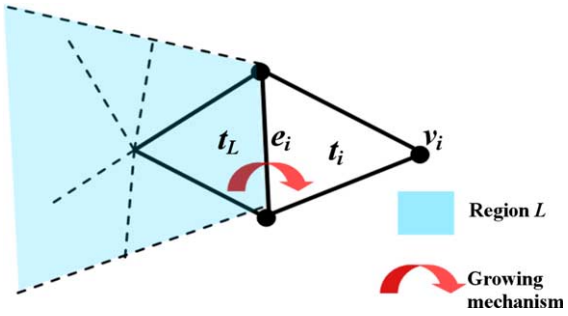

Fig. 6. Considered features for the region growing process.

4.3.2. The growing mechanism

When a seed triangle is encountered, a new region is created, containing this triangle, associated with a new label L and a curvature value c_L.

Then a recursive process extends this region (see Fig. 6): for each triangle t_L belonging to the region, for each non-sharp edge e_i of this triangle, we consider the associated neighbouring triangles t_i and their opposite vertex v_i. If v_i is a *sharp vertex* or if it has the c_L curvature value, then the considered triangle is integrated into the region (the curvature associated with the region remains the same).

This growing algorithm is repeated for every other triangle marked as seed and still unlabelled. With this process, it remains, sometimes, not labelled triangles at the end of the algorithm (for example, triangles with three vertices from different curvature clusters). In practice these holes appear only for a few triangles and often near of the boundaries between regions. A simple crack filling process fits them by integrating these triangles to the most represented regions of their neighbourhood. Fig. 7 shows the region growing process for the 'Fandisk' object, starting from a 18 clusters vertex classification. The region growing extracts 128 connected regions (region colors are randomly chosen).

The growing algorithm depends on the number of curvature clusters. Moreover, a fixed value of K for the K-Means classification algorithm can generate different sets of clusters because of the random choice of the K initial seeds. Thus for a given K, the growing step can give different results in term of number and localization of extracted regions. In order to reduce this dependence on the number of curvature clusters and on the choice of the K initial seeds, a region merging process was developed in order to unify results.

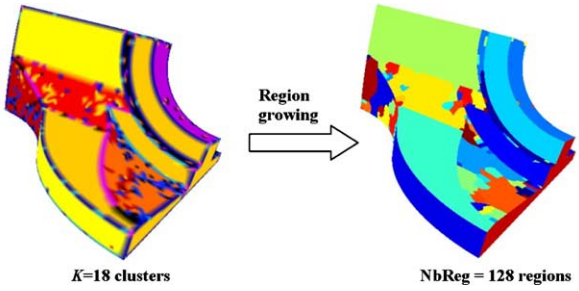

Fig. 7. The region growing process for the Fandisk mesh (region colors are randomly chosen).

4.4. The region merging process

The region merging process aims at:

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

4.4.1. The region adjacency graph

The data scheme strongly contributes to the efficiency of an algorithm. The purpose here is to merge adjacent similar regions. Consequently, a good representation is a region adjacency graph (RAG), a data scheme used in image segmentation [23,24] and by Garland et al. [7] for their face clustering algorithm. This algebraic structure contains a set of nodes and a set of edges. 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 between the two corresponding regions.

4.4.2. General algorithm

Once connected regions have been extracted by the region growing algorithm, the RAG is processed, and distances between adjacent regions are calculated. Then the reduction of the graph is processed: at each iteration the smallest edge of the graph is eliminated, thus the corresponding regions are merged; then the graph is updated (recalculation of the adjacency relations and of the similarity distances of the resulting region with its neighbours). When two regions are merged, their curvatures are merged proportionally to their areas to give the curvature of the resulting region. This graph reduction ends when the number of regions reaches a queried value chosen by the user, or when the weight of the smallest edge is larger than a given threshold.

4.4.3. Region distance measurement

The region 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} the aim of which is to eliminate the smallest regions

$$D_{ij} = \mathrm{DC}_{ij} \times N_{ij} \times S_{ij} \quad (1)$$

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

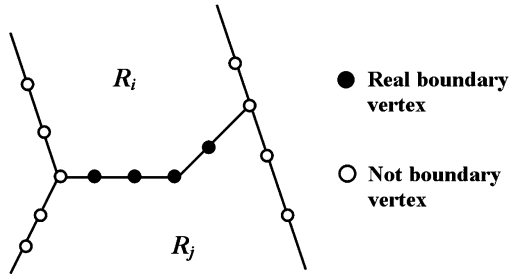

Fig. 8. Representation of the vertices taken into account for the calculation of the mean curvature c_{ij} of the boundary between R_i and R_j.

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

c_i and c_j come from the region growing step. c_{ij} is the average of the curvatures of the vertices belonging to the boundary between the two regions. Only vertices with two incident edges separating these regions (*real boundary vertices*) are taken into account (see Fig. 8), in order to consider only the real boundary between them.

It is important for the calculation of the curvature distance between R_i and R_j to consider not only their respective curvatures c_i and c_j but also their boundary one c_{ij}, because two situations may happen between these regions. Either regions have different curvatures and no precise boundary (see Fig. 9a), or regions have almost the same curvature but a very significant boundary (see Fig. 9b).

The N_{ij} coefficient measures the nesting between the two corresponding regions

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

with P_i (resp. P_i) the perimeter of the ith (resp. jth) region and P_{ij} the size of the common border between the ith and jth regions. This coefficient was introduced in image processing by Schettini [25] for color image segmentation.

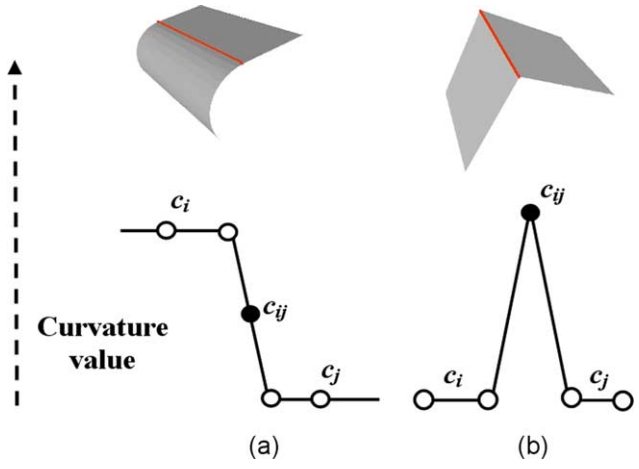

Fig. 9. The two different situations between two adjacent regions. (a) No boundary but a curvature difference, (b) no curvature difference, but a significant boundary.

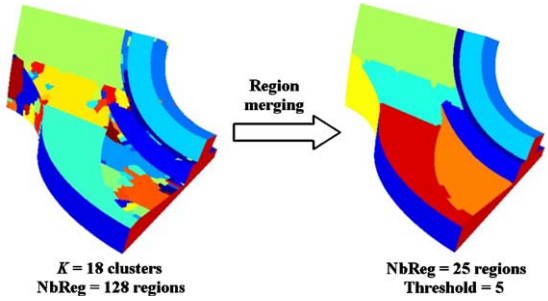

Fig. 10. The region merging process for the Fandisk mesh.

The aim of the N_{ij} factor is to consider the spatial disposition of the regions in the merging decision. 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 tends to better group smallest regions

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

where A_i (resp. A_j) is the area of the ith (resp. jth) region, $A_{\min}$ is a minimum area set by the user and ε is a near 0 positive value. The S_{ij} factor can be considerate as a filtering factor. When a region area is smaller than $A_{\min}$, it is considered as too small, hence its distance with its adjacent regions is reduced by the S_{ij} coefficient, equal to ε; the considered region will be more easily merged with another. This method aims at eliminating the smallest regions. The value of $A_{\min}$ depends on the queried size (or number) of final regions. The value of ε is fixed to 1×10^{-5}. This value accelerates the fusion of the smallest regions, while keeping the merging order.

Fig. 10 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. 7). It contains 128 connected spatial regions. After the merging process, the final region number is 25. The merging threshold was fixed to 5 (every object is scaled to a bounding box of length equal to 1).

4.5. Experiments and results

Our segmentation method was tested on several different objects. Examples are given for several basic common 3D CAD elements (see Fig. 11) and for three objects from different natures (see Fig. 12): a rather smooth object (Pawn), a mechanical highly tessellated object (Fandisk) and an optimized triangulated CAD object (Swivel).

Fig. 11 presents segmentation results for the 3D basic elements. Their decompositions are intuitively correct and adapted to our constant curvature region extraction and surface fitting objectives.

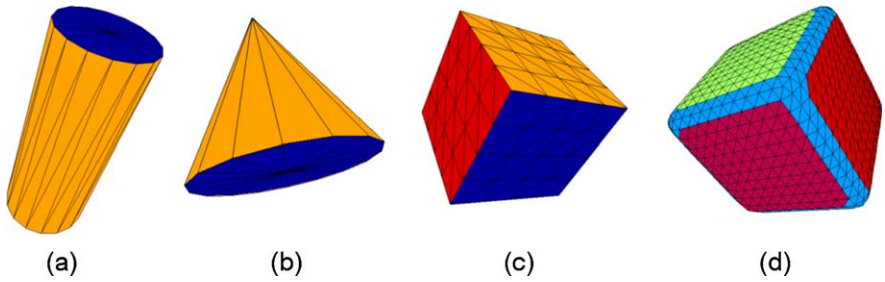

Fig. 11. Segmentation results for several basic 3D elements. (a) Cylinder (three regions), (b) cone (two regions), (c) cube (six regions), (d) smooth cube (seven regions).

For the Fandisk object (see Fig. 12b), we obtain patches with almost constant curvature as for the 'Pawn' (see Fig. 12a). Our method allows detecting curvature transitions 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 (see Fig. 12c), we obtain good results after the enrichment of detected sharp triangles.

We have studied the computational cost of the algorithm; examples are presented in Table 1. All experiments were conducted on a PC, with a 2 GHz XEON bi-processor. Let n be the number of triangles, K the number of curvature clusters and R the number of curvature classification algorithm iterations. Thus complexities are the followings: $O(n)$ for the curvature computation and $O(nRK)$ for the vertex classification. Considering the region growing and merging steps, the processing time depends on the grown region number and on the region removal number. On the whole, the entire process is rather fast and an online utilisation is realistic.

We have also studied the dependence of the algorithm on the number of curvature clusters K which parameters the K-means algorithm within the vertex classification procedure. We have conducted tests with several objects; results for Fandisk are shown in Table 2. The vertex classification was processed with different values for K (K' is the cluster number after regularization) and a unique threshold fixed to 50 was chosen for the region

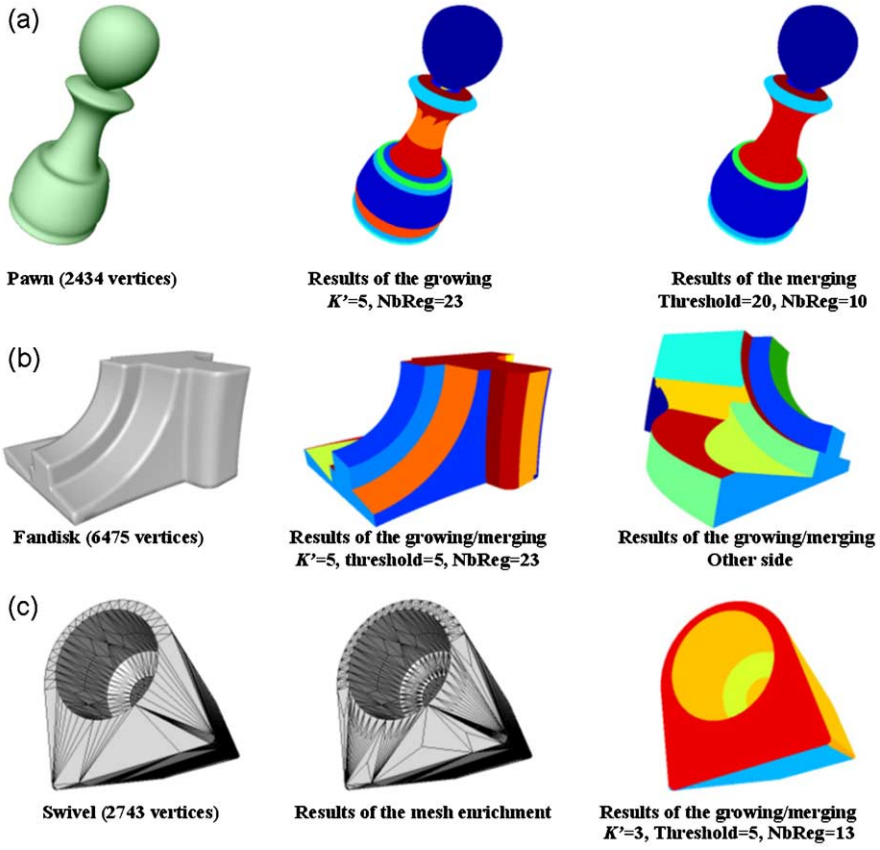

Fig. 12. Segmentation of Pawn (a), Fandisk (b) and Swivel (c) objects. The region merging threshold is 5 for Fandisk and Swivel, and 20 for Pawn.

Table 1
Computing times of the different steps of our segmentation algorithm for the objects presented in Fig. 12

Object	Curvature processing time (s)	Vertex classification time (s)	Region growing time (s)	Region number after growing	Region merging time (s)	Region number after merging	Total (s)
Pawn	31×10^{-3}	78×10^{-3}	15×10^{-3}	23	16×10^{-3}	10	156×10^{-3}
Fandisk	78×10^{-3}	281×10^{-3}	62×10^{-3}	46	203×10^{-3}	23	624×10^{-3}
Swivel	31×10^{-3}	47×10^{-3}	16×10^{-3}	18	15×10^{-3}	13	109×10^{-3}

merging step. Results show that of course K influences the number of regions created after the growing step (besides, this number can vary for a same K, because of the random choice of the K initial seeds for the K-Means algorithm) but the final region number is regularized by the merging algorithm and the resulting segmented regions are almost identical. It happens because the merging process is strongly linked to the curvature values of the regions. Indeed, the different sets of regions associated to different values of K for a same object, keep practically the same curvature distribution.

Thus we have strongly reduced the algorithm dependence to the number of curvature clusters; it does not have to be considered as a critical parameter for the method.

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. 10). Fig. 13 presents examples of artefacts, blue and pink regions in the red ellipse are not correct (see Fig. 13a), their boundary is not straight. In Fig. 13b, green and blue regions are not complete regarding to the original object and the green one presents a discontinuity.

Table 2
Influence of the cluster number K of the classification algorithm, on the number of final regions for a given threshold

K	K' (regularized)	NbReg after growing	NbReg after merging
5	5	46	15
10	7	62	15
10	9	99	15
15	9	76	15
20	11	84	15
20	17	116	15

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 region boundary edges are extracted and for each of them a correctness score is processed (the *Boundary Score*). 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 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 [19]. The curvature tensors at the natural boundaries of an object tend to be 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.

Fig. 14a shows a natural hand made segmentation of a smooth cube object into homogeneous curvature patches.

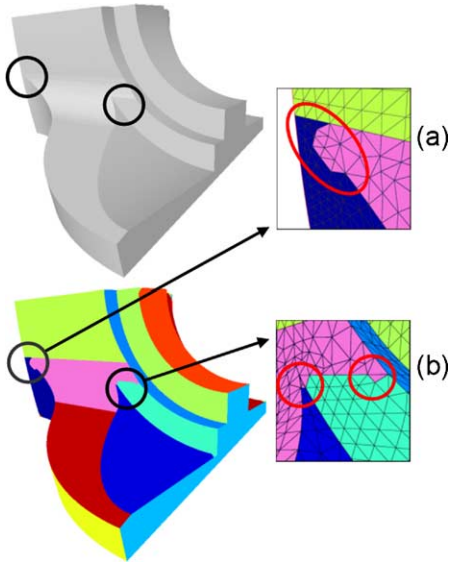

Fig. 13. Zoom on artefacts for the segmented Fandisk object.

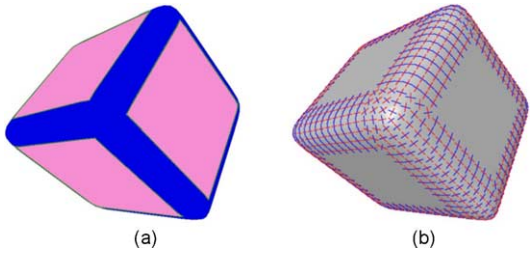

Fig. 14. Natural constant curvature patches of the 'Smooth cube' object (a) and its principal curvature directions (b), *dmin* in red and *dmax* in blue.

Fig. 14b shows maximum and minimum curvature directions.

The boundaries of the patches follow the minimum directions, except around isotropic regions (at the corners of the cube). Therefore, the angles between a boundary edge and the minimum curvature directions of its vertices can represent a good evaluation of its 'correctness'.

The boundary score $S \in [0,90]$, calculated for an edge e_i, is:

$$S(e_i) = \frac{S_a(e_i) + \omega_c \times S_c(e_i)}{1 + \omega_c} \tag{5}$$

$S_a \in [0,90]$ depends on the angles between the edge and its vertices curvature directions. This value is homogeneous to an angle value in degrees. S_c considers a curvature difference; it is also normalized in [0,90]. ω_c is a weighting coefficient which is fixed to 0.5 in our examples. S_a and S_c are detailed in the following paragraphs.

The *angle score* S_a considers the angles θmin_{i1} and θmin_{i2} (see Fig. 15) between the edge e_i and the minimum curvature directions of its vertices V_{i1} and V_{i2}. The score also considers 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 regions, like the corners of the smooth cube, for instance (see Fig. 14). Thus the angle score $S_a \in [0,90]$ is processed according to

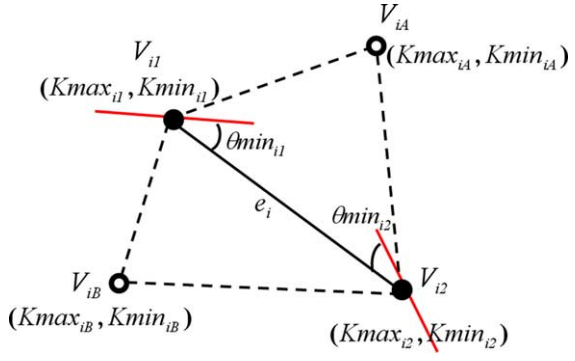

Fig. 15. Elements taken into account for the calculation of the *Boundary Score* of the edge e_i.

the following equation

$$S_a(e_i) = \frac{1}{2}\left(\frac{(\theta\text{min}_{i1} \times K\text{max}_{i1} + \theta\text{max}_{i1} \times K\text{min}_{i1})}{K\text{max}_{i1} + K\text{min}_{i1}} + \frac{(\theta\text{min}_{i2} \times K\text{max}_{i2} + \theta\text{max}_{i2} \times K\text{min}_{i2})}{K\text{max}_{i2} + K\text{min}_{i2}}\right) \tag{6}$$

with θmin_{i1}, θmin_{i2} and θmax_{i1}, θmax_{i2} are the respective angles in degrees ($\in [0,90]$) of the considered edge e_i with the minimum and maximum curvature directions of its vertices. $K\text{min}_{i1}$, $K\text{min}_{i2}$ and $K\text{max}_{i1}$, $K\text{max}_{i1}$ are the respective values of minimum and maximum curvatures of the vertices of the edge e_i.

The curvature score S_c considers the curvature variation between the edge e_i and its opposite vertices V_{iA} and V_{iB} (see Fig. 15). We consider two cases for which the edge corresponds to a correct boundary:

- Curvature of V_{iA} is different from curvature of V_{iB} (see Fig. 9a), the associated distance is D_{iAB}.
- Curvature of V_{iA} (resp. V_{iB}) is different from curvature of e_i (see Fig. 9b), the associated distance is D_{iAe} (resp. D_{iBe}).

Distances D_{iAB}, D_{iAe} and D_{iBe} are normalized in [0,1] and processed as follows:

$$D_{iAB} = \frac{\|K\text{min}_{iB} - K\text{min}_{iA}\| + \|(K\text{max}_{iB} - K\text{max}_{iA})\|}{\max(K\text{min}_{iB}, K\text{min}_{iA}) + \max(K\text{max}_{iB}, K\text{max}_{iA})}$$

For the calculation of D_{iAe} (resp. D_{iBe}), we consider the distance between V_{iA} (resp. V_{iB}) and e_i, as the minimum of the distance from V_{iA} (resp. V_{iB}) to V_{i1} and from V_{iA} (resp. V_{iB}) to V_{i2}

$$D_{iAe} = \min\left(\frac{||K\text{min}_{iA} - K\text{min}_{i1}|| + ||(K\text{max}_{iA} - K\text{max}_{i1})||}{\max(K\text{min}_{iA}, K\text{min}_{i1}) + \max(K\text{max}_{iA}, K\text{max}_{i1})}, \frac{||K\text{min}_{iA} - K\text{min}_{i2}|| + ||(K\text{max}_{iA} - K\text{max}_{i2})||}{\max(K\text{min}_{iA}, K\text{min}_{i2}) + \max(K\text{max}_{iA}, K\text{max}_{i2})}\right)$$

The formula for D_{iBe} is the same with $K\text{min}_{iB}$ and $K\text{max}_{iB}$ instead of $K\text{min}_{iA}$ and $K\text{max}_{iA}$.

Knowing that the curvature score S_c is a correctness indicator and that it must take into account the two cases described above, thus S_c is defined as follows:

$$S_c(e_i) = 1 - \max(D_{iAB}, D_{iAe}, D_{iBe}) \tag{7}$$

S_c is then normalized in [0,90] to keep the coherency with the angle score S_a.

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 which completes final boundaries.

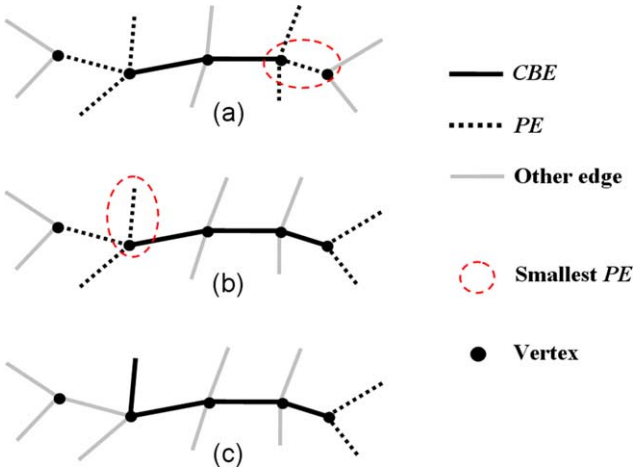

Fig. 16. Three steps (a–c) of the boundary tracking algorithm, with associated positions of correct boundary edges (CBE), potential edges (PE) and smallest potential edges, at each iteration.

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 chosen (ST is fixed to 5 in our examples). 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. Figs. 17c and 18c show this marking process, starting from the region segmentation (see Figs. 17a and 18b), 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. 16a 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 e_i depends of its score $S(e_i) \in [0,90]$ but also of its angle $\theta(e_i, e_{\text{CBE}}) \in [0,180]$ with its neighbouring CBE, because we try to limit the deviation of the boundary

$$P(e_i) = S(e_i) + \omega_\theta \times \theta(e_i, e_{\text{CBE}}) \tag{8}$$

ω_θ is a weighting coefficient, it is fixed to 2 in our examples.

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 the potential edge (PE) sorted list is organised, 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. 16 presents three iterations of the contour tracking algorithm. In Fig. 16a, 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. 16b). The positions and numbers of the PEs are then updated. The process continues in Fig. 16c, with another PE integrated to the contour.

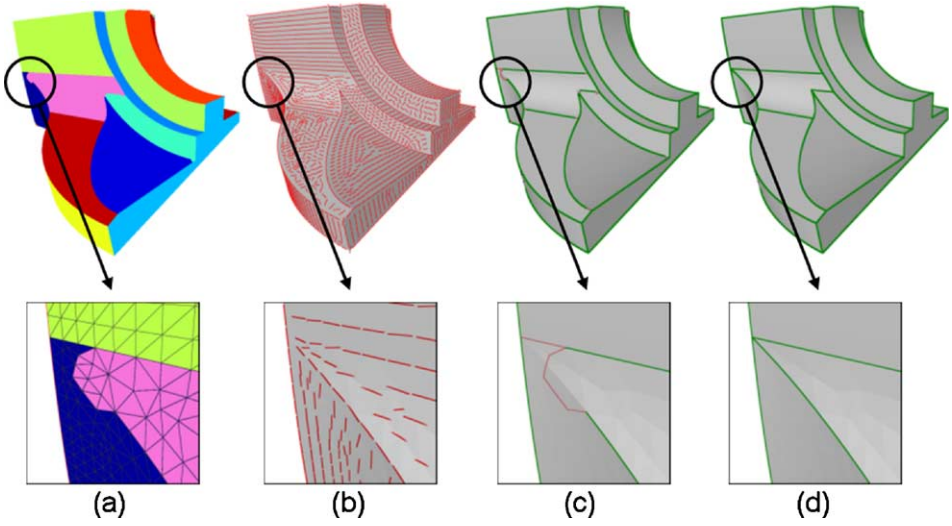

Fig. 17. The different steps of the Boundary Rectification for the Fandisk object with a zoom on an artefact correction. (a) Segmented object. (b) Minimum curvature directions. (c) Correct boundary edge extraction and marking. (d) Corrected boundaries after the contour tracking.

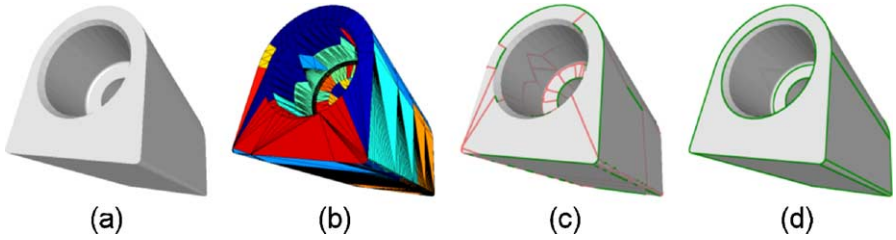

Fig. 18. The different steps of the Boundary Rectification for an artificially bad segmented CAD object. (a) Original object. (b) Bad segmented object. (c) Correct boundary edge extraction and marking. (d) Corrected boundaries after the contour tracking.

5.4. Experiments and results

The rectification method is especially adapted to CAD or mechanical objects, where there exist real defined regular boundaries. On natural or organic objects the fact of rectifying boundaries does not have a real significance since even a human hand could not trace precise and smooth boundaries. We have tested our rectification method on various models resulting from our region segmentation algorithm. Fig. 17 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. 18 shows results on an artificially bad segmented CAD object.

We can observe that bad boundary edges are eliminated whereas correct ones are correctly extracted and completed to give a very satisfying set of surface patches. Even with very few correct boundary edges, final boundaries of the object are well extracted. The rectification has also been tested on large and complex CAD models. Fig. 19 shows an example for the 'Sheet' model which contains about 56,000 triangles (40,316 vertices). Initial extracted regions (see Fig. 19b) are satisfactory regarding to the global decomposition and final boundaries (see Fig. 19c) are quite smooth and correct after the rectification.

This rectification process is very fast: 16 ms for Fandisk, 15 ms for Swivel and 94 ms for Sheet, and moreover, it is independent of the region segmentation method presented in Section 4; we can imagine using it as a contour tracking post-process to a hard edge detection algorithm for example.

Finally, Fig. 20 presents some results of the whole process (region segmentation and boundary rectification) for three quite complex CAD objects. We have represented boundaries of the final extracted patches. The decomposition results as well as the boundaries are very satisfying with regard to our further surface fitting application.

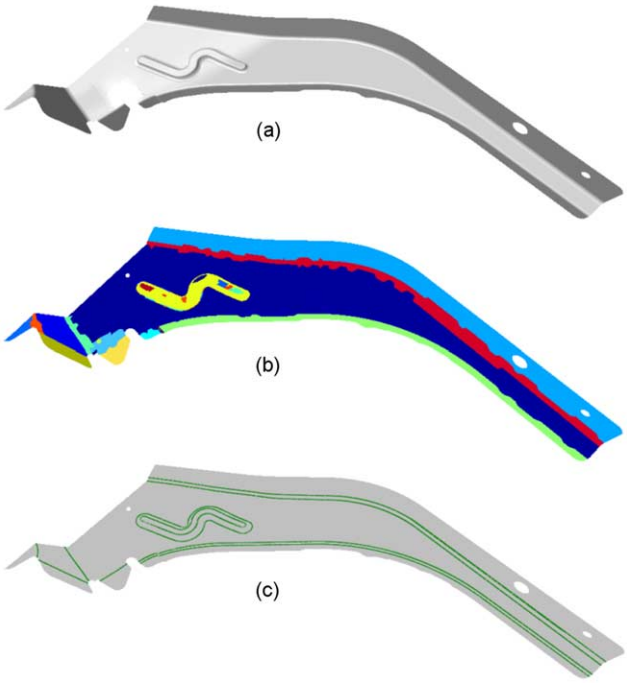

Fig. 19. The boundary rectification for a large CAD object (40,316 vertices). (a) Original object. (b) Results of the region segmentation. (c) Results of the boundary rectification.

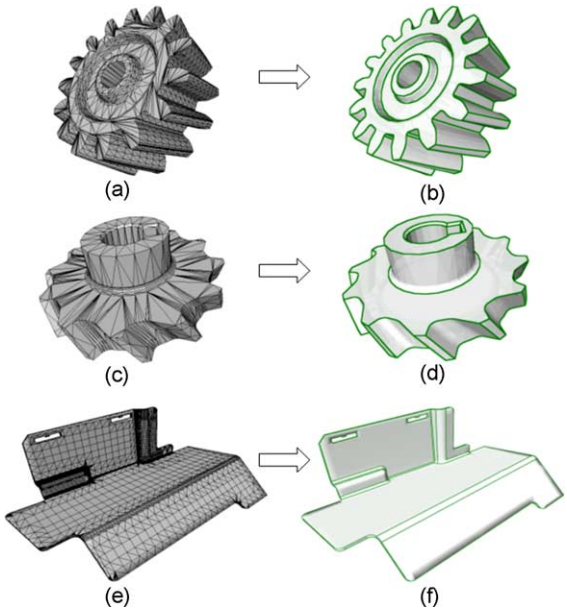

Fig. 20. Three CAD objects, 'Wheel' (3721 vertices) (a), 'Hub' (1247 vertices) (c), and 'Clip' (18,734 vertices) (e), with their resulting patches after the whole algorithm (b, d, and f).

6. Conclusion

This paper presents an original segmentation method to decompose a 3D-mesh into homogeneous curvature surface patches with clean boundaries. The simple and efficient curvature classification detects any curvature transition and thus allows segmenting the object into near constant curvature regions and not just cutting the object along its hard edges.

The triangle growing process well transmits region information from vertices to triangles even for optimized tessellated CAD objects.

Our original boundary rectification method based on curvature tensor orientations, allows suppressing boundary artefacts 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 to hard edge detection algorithms for example, to complete hard edge 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 segmentation method is involved in a larger CAD object compression scheme. 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.

References

[1] Lavoue G, Dupont F, Baskurt A. Constant curvature region decomposition of 3D-meshes by a mixed approach vertex-triangle. J WSCG 2004;12(2):245–52.
[2] Hoffman R, Jain AK. Segmentation and classification of range images. IEEE Trans Pattern Anal Mach Intell 1987;9(5):608–20.
[3] Sapidis NS, Besl PJ. Direct construction of polynomial surfaces from dense range images through region growing. ACM Trans Graph 1995; 14(2):171–200.
[4] Leonardis A, Jaklic A, Solina F. Superquadrics for segmenting and modeling range data. IEEE Trans Pattern Anal Mach Intell 1997; 19(11):1289–95.
[5] Chevalier L, Jaillet F, Baskurt A. A segmentation superquadric modelling of 3D objects. WSCG, Plzen–Bory, Czech Republic 2003; 11(2):232–40.
[6] Chaine R, Bouakaz S. Segmentation of 3-D surface trace points, using a hierarchical tree-based diffusion scheme. Fourth Asian conference on computer vision ACCV2000, Taiwan, vol. 2; 2000. p. 995–1002.
[7] Garland M, Willmott A, Heckbert P. Hierarchical face clustering on polygonal surfaces. ACM Symp Interactive 3D Graph 2001;49–58.
[8] Benko P, Varady T. Segmentation methods for smooth point regions of conventional engineering objects. Computer-Aided Des 2004; 36(6):511–23.
[9] Serra J. Image analysis and mathematical morphology. London: Academic Press; 1982.
[10] Mangan A, Whitaker R. Partitioning 3D surface meshes using watershed segmentation. IEEE Visual Computer Graph 1999;5(4): 308–21.
[11] Sun Y, Page D, Paik J, Koschan A, Abidi M. Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing. IEEE Int Conf Image Process, NY, USA 2002;3:825–8.
[12] Razdan A, Bae M. A hybrid approach to feature segmentation of triangle meshes. Computer-Aided Des 2003;35(9):783–9.
[13] Zhang Y, Paik J, Koschan A, Abidi M, Gorsich D. A simple and efficient algorithm for part decomposition of 3D triangulated models based on curvature analysis. IEEE Int Conf Image Process, Rochester, NY, USA 2002;3:273–6.
[14] Li I, Toon T, Tan T, Huang Z. Decomposing polygon meshes for interactive applications. Symp Interactive 3D 2001;35–42.
[15] Katz S, Tal A. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans Graph 2003;22(3):954–61.
[16] Cohen-Steiner D, Morvan J. Restricted delaunay triangulations and normal cycle. 19th Annual ACM Symposium on Computational Geometry; 2003. p. 237–46.
[17] Meyer M, Desbrun M, Schröder P, Barr H. Discrete differential-geometry operators for triangulated 2-manifolds. International workshop on visualization and mathematics, Berlin, Germany; 2002.
[18] Taubin G. Estimating the tensor of curvature of a surface from a polyhedral approximation. Fifth international conference on computer vision; 1995. p. 902–7.
[19] Alliez P, Cohen-Steiner D, Devillers O, Levy B, Desbrun M. Anisotropic polygonal remeshing. ACM Trans Graph, SIGGRAPH'2003 Conf Proc 2003;22(3):485–93.
[20] Alexander D, Gee J. Elastic matching of diffusion tensor images. Computer Vision Image Understanding 2000;77:233–50.
[21] Basser P, Pierpaoli C. Microstructural and physiological features of tissues elucidated by quantitative diffusion tensor MRI. J Magn Reson 1996;111:209–19.
[22] Gersho A, Gray R. Vector quantization and signal compression. Boston: Kluwer; 1992.
[23] Saarinen K. Color image segmentation by a watershed algorithm and region adjacency graph processing. IEEE Int Conf Image Process, Austin, TX, USA; 1994. p. 1021–4.
[24] Idrissi K, Lavoué G, Ricard J, Baskurt A. Object of interest based visual navigation, retrieval and semantic content identification system. Computer Vision Image Understanding 2004;94(1–3): 271–94.
[25] Schettini R. A segmentation algorithm for color images. Pattern Recogn Lett 1993;14:499–506.

Guillaume Lavoué received in 2002 his Engineering Degree in Electronic, Telecommunication and Computer Science from CPE-Lyon (France) and his MS degree in Image Processing from the university Jean Monnet of St Etienne (France). He is actually PhD student at LIRIS Laboratory in the University Claude Bernard of Lyon (France). His research interests include 3D digital image processing, geometric modeling and more precisely 3D compression and subdivision surfaces.

Florent Dupont received his BS and MS degree in 1990, and his PhD in1994 from INSA of Lyon, France. Since 1998, he is Associate Professor. He now works at LIRIS Laboratory in the University Claude Bernard of Lyon, France. His technical research concerns 3D digital image processing, 3D compression and discrete geometry.

Atilla Baskurt received his BS degree in 1984, his MS in 1985, and his PhD in 1989, all in electrical engineering from INSA of Lyon, France. From 1989 to 1998, he was Associate Professor at INSA of Lyon. Since 1998, he has been with the University Claude Bernard of Lyon, France, where he is a professor in electrical and computer engineering. He leads the images & videos group of LIRIS research laboratory at this University. This group performs image and video analysis and segmentation for image compression, image retrieval, shape detection, and identification. His technical research and experience includes digital image processing, image compression and segmentation, image indexing and retrieval, especially for multimedia applications.