Curvature tensor Based Triangle Mesh Segmentation with Boundary Rectification

Lavoué Guillaume	Dupont Florent	Baskurt Atilla
LIRIS FRE 2672 CNRS	*LIRIS FRE 2672 CNRS*	*LIRIS FRE 2672 CNRS*
43, Bd du 11 novembre	*43, Bd du 11 novembre*	*43, Bd du 11 novembre*
69622 Villeurbanne Cedex, France	*69622 Villeurbanne Cedex, France*	*69622 Villeurbanne Cedex, France*
glavoue@liris.cnrs.fr	*fdupont@liris.cnrs.fr*	*abaskurt@liris.cnrs.fr*

Abstract

This paper presents a new and efficient algorithm for decomposition of 3D arbitrary triangle mesh into surface patches. Our method is based on the curvature tensor field analysis and presents two distinct complementary steps: a region based segmentation 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 artefacts 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 task like parametric or subdivision surface fitting in an adaptive compression objective.

Keywords: Segmentation, 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.

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 connected known and near constant curvature regions with clean and regular 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

There has been a considerable research work relevant to the problem of 3d-object segmentation. However the majority of these methods concern range images [1][2][3][4] or 3d point clouds [5][6]. Only few studies concern triangle meshes which is nevertheless the most widespread representation for 3d-objects. Wu and Levine [7] present a physics-based original method which uses the idea of electrical charge but this approach is computationally expensive. The other approaches generally use discrete curvature analysis combined with the Watershed algorithm described by Serra [8] in the 2D image segmentation field. Mangan and Whitaker [9] generalize the Watershed method to arbitrary meshes, using the Gaussian curvature or the norm of covariance of adjacent triangle normals at each mesh vertex as the height field. Sun and al. [10] 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 [11] proposed an hybrid method which combines Watershed algorithm with the extraction of feature boundaries by the analysis of dihedral angle between polygon faces. Zhang et al. [12] use the sign of the Gaussian curvature to mark boundaries, and process a part decomposition.
We have distinguished two major shortcomings in these existing methods. They are described below.
Firstly, many approaches are only vertex based [10] [12], each vertex has its region information, therefore triangles on boundaries have multi-regions information, it results that boundaries are fuzzy; they are not clearly identified in term of edges. Our method is an hybrid approach vertex-triangle, which combines a vertex classification with a triangle region growing and merging. Boundaries between regions are thus clearly distinguishable edges. Moreover our boundary rectification method allows extracting “real” regular boundaries of the object, by suppressing artefacts or discontinuities.
Secondly, most of the approaches discussed above, particularly those based on the Watershed algorithm, extract regions surrounded by high curvature boundaries [10] [11] [12] (see Fig.9.b) but fails to distinguish simple curvature transition between vertices (see Fig.9.a) without curvature pick. The K-Means vertex classification that we use allows to detect these transitions.

3. Method overview

We present a decomposition algorithm of arbitrary triangle meshes into known and almost constant curvature surface patches with clean and regular boundaries. We address particularly the problem of CAD parts, but natural objects are also considered. Our approach is based on two steps (see Fig.1):
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. [13]. 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.

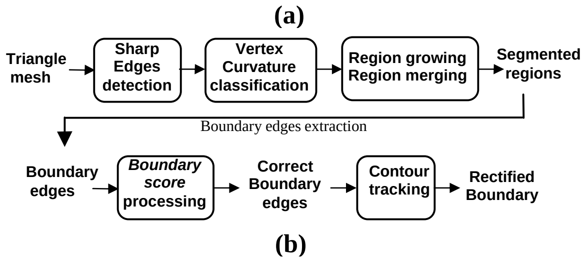


Figure 1. The two steps of the method. (a) Constant curvature region segmentation. (b) boundary rectification.

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 [11]. Therefore we subdivide these *sharp triangles* by adding a new

vertex at the center (see Fig.2). The region segmentation is thus applied on this modified mesh and added vertices are removed at the end of the algorithm.

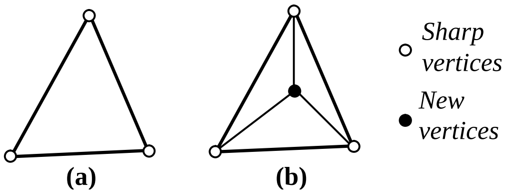


Figure 2. The mesh enrichment process. (a) Triangle with three *sharp vertices*. (b) Associated subdivided triangle.

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 these information for each vertex of the input mesh.

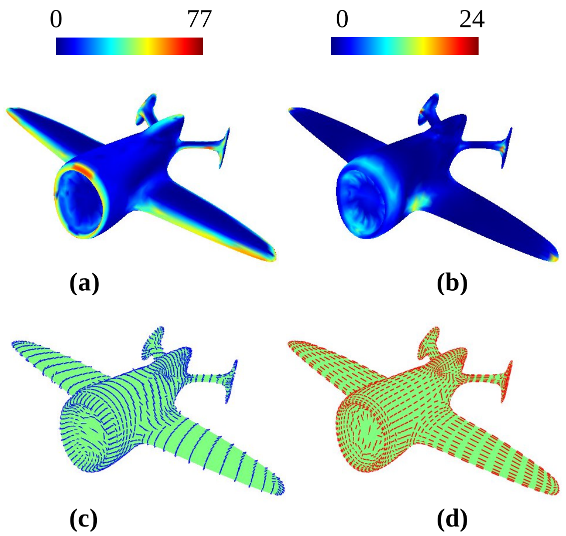


Figure 3. Curvature fields for the 3D object "Plane". (a) *Kmax*, (b) *Kmin* (absolute value), (c) *dmax*, (d) *dmin*.

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 [13][14][15].

We have implemented the work of Cohen-Steiner et al. [13], 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 [16], therefore it is independent of the sampling; thus 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.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. 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.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 [17][18] 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) [19], 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.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 (clusters colors are yellow, orange, blue, dark blue and green).

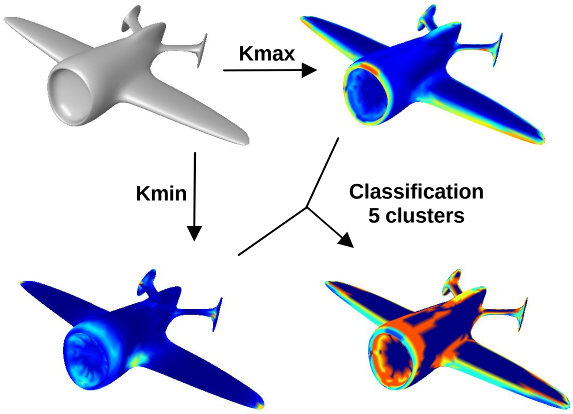


Figure 4. Vertex classification of the *Plane* mesh in 5 curvature clusters.

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 two 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.5.a).
- It contains two *sharp vertices*, thus the curvature cluster value c_i of the third vertex is assigned to the created region (see Fig.5.b).

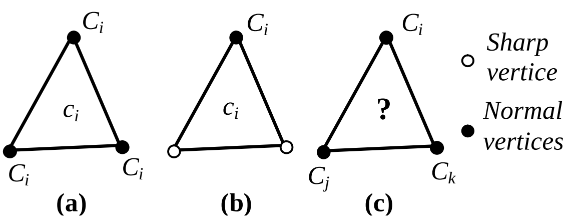


Figure 5. The two *seed triangle* situations (a) (b) and an undetermined triangle (c).

In every other cases (see Fig.5.c for example), 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.

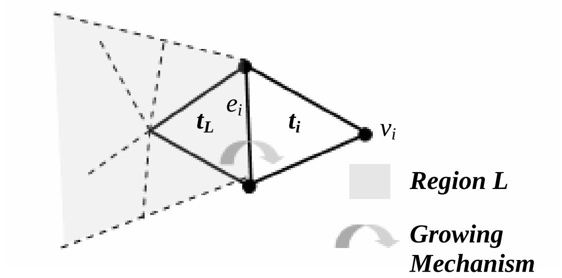


Figure 6. Considered features for the region growing process.

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 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.

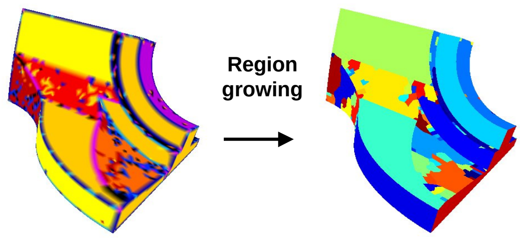


Figure 7. The region growing process for the "Fandisk" mesh (Regions colors are randomly chosen).

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 (regions colors are randomly chosen).

The growing algorithm is very dependant of 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 merging step can give different results in term of number and localization of extracted regions. Because of this uncertainty, and to suppress the dependency to the number of curvature clusters, a region merging process was developed, in order to unify results.

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.

4.4.1. The region adjacency graph. The efficiency of an algorithm depends on the data scheme used. The purpose here is to merge adjacent similar regions. Thus a good representation to operate is a region adjacency graph (RAG), a data scheme used in image segmentation [20][21]; 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 curvature 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. When two regions are merged, their curvatures are merged proportionally to their areas to give the curvature of the resulting region. 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.

4.4.3. 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.

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

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. 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.

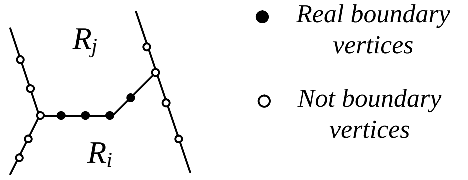


Figure 8. Representation of the vertices taken into account for the calculation of the average curvature of the boundary between R_i and R_j.

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 exist between these regions. Either regions have different curvatures and no precise boundary (see Fig.9.a), or regions have almost the same curvature and a very different boundary curvature (see Fig.9.b).

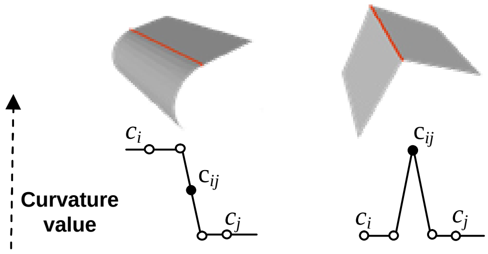


Figure 9. The two different situations between two adjacent regions. (a) no boundary but a curvature difference, (b) no curvature difference, but a significant boundary.

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_j) the perimeter of the i^th^ (resp. j^th^) region and P_{ij} the size of the common border between the i^th^ and j^th^ regions. This coefficient was introduced in image processing by Schettini [22] for color image segmentation. 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, thus their similarity distance is reduced.

The S_{ij} coefficient purpose is to accelerate the fusion of the smallest regions.

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

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

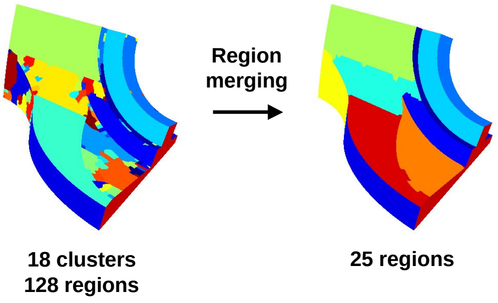


Figure 10. The region merging process for the "Fandisk" mesh.

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. It contains 128 connected spatial regions. After the merging process, the final region number is 25. The merging threshold was fixed to 5.

4.5. Experiments and results

Our segmentation method was tested on several different objects. Examples are given for three objects from different nature: a rather smooth object (Pawn), a mechanical highly tesselated object (Fandisk) and an optimized triangulated CAD object (Swivel). Results are shown on Fig.11. For the "Fandisk" object (see Fig.11.b) we obtain patches with almost constant curvature as for the "Pawn" (see Fig.11.a). Our method allows detecting curvature transition or inflexion points and not only regions separated by high curvature boundaries, or sharp boundaries, like traditional watershed method. Even for the bad tessellated "Swivel" object (see Fig.11.c), we obtain good results after the enrichment of detected *sharp triangles*.

We have studied the dependency of the algorithm on the number of curvature clusters *K* which parameters the K-means algorithm during the vertex classification step. We have conducted tests with several objects; results for Fandisk are shown in Table 1. 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 merging process. 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. Thus we have suppressed the algorithm dependency to the number of curvature clusters; it does not have to be considered as a parameter for the method.

Table 1. 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

Our purpose is to obtain clean patches with constant curvature in a subdivision or parametric surface fitting objective.

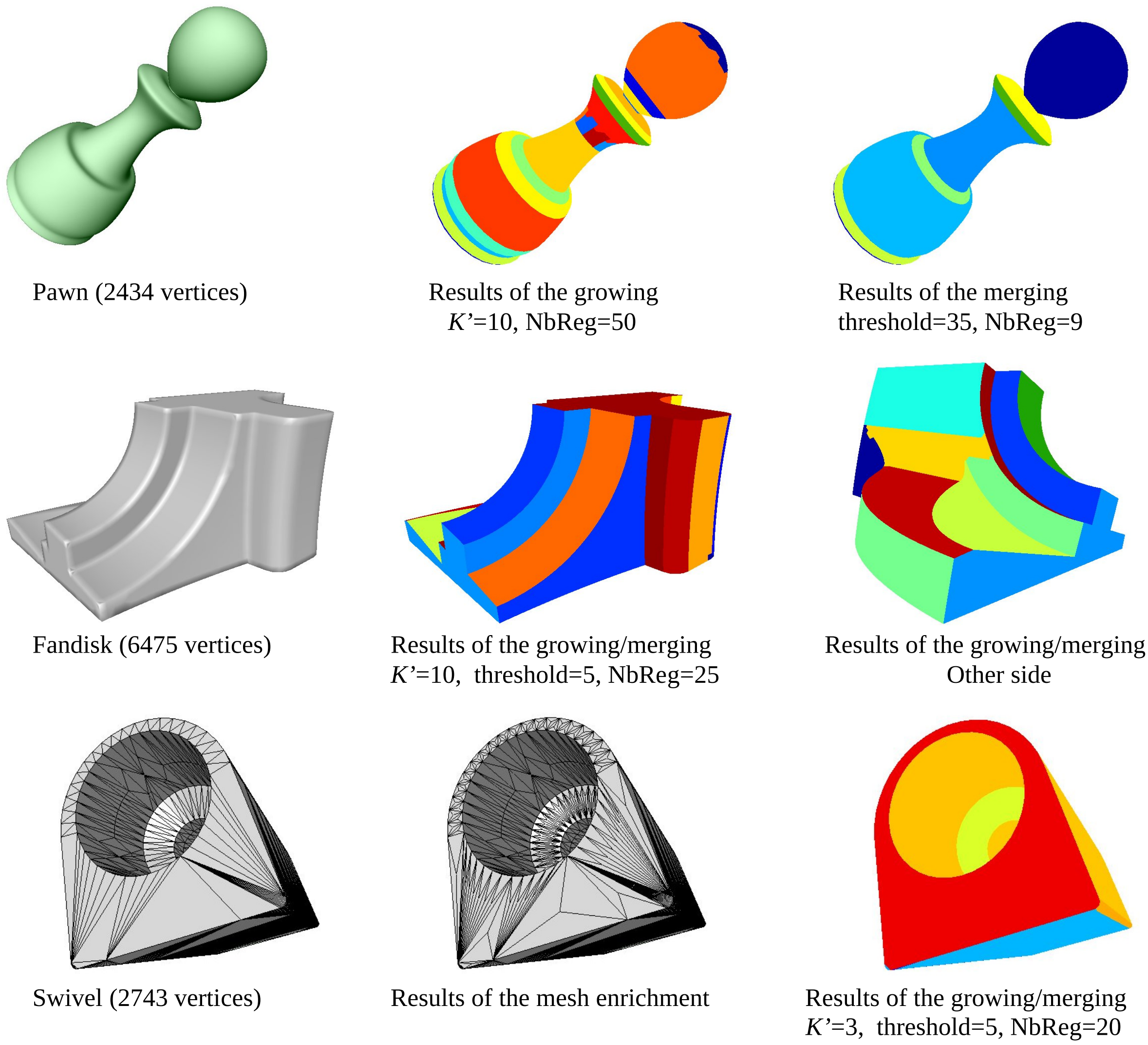


Figure 11. Segmentation of "Pawn", "Fandisk" and "Swivel" objects. The region merging threshold is 5 for "Fandisk" and "Swivel", and 35 for "Pawn".

Although our segmented regions are very correct in terms of general shape, their boundaries present some discontinuities (see Fig.12), particularly when we consider a high number of curvature clusters; for instance, in Fig.10 after the merging, we obtain good surface patches in terms of disposition, and general shape but a lot of small defects have appeared on the boundaries.

There exists a main reason for the apparition of these artefacts: the basis for our method is the curvature field of the object. This information is based on the vertices, whereas we finally consider triangles and edges. This transition Vertex-Triangle (processed during the region growing step) is very difficult to manage thus we lost a part of the initial accuracy.

5. The boundary rectification process

5.1. Objective

Our region segmentation method extracts near constant curvature, topologically simple patches from the 3D-objects, and give good qualitative results in terms of nature and disposition of the segmented regions. Nevertheless, our method, like most of the existing methods, does not extract perfect boundaries, without discontinuities; generally, regions present artefacts at the limits of their boundaries (see section 4.5). Fig.12 presents examples of this region boundaries vagueness; in Fig.12.a blue and yellow regions in the red ellipse are not correct, their boundary is not straight; in Fig.12.b, green and pink regions are not complete regarding to the original object and the green one presents a discontinuity.

In this context, the objective of the boundary rectification process is to suppress these artefacts, in order to obtain clean boundaries corresponding to real natural boundaries of the object.

The rectification method is composed of two principal steps: firstly, segmented object boundary edges are extracted and for each of them a *correctness* score is processed. Then, starting from the estimated *correct* boundary edges, the object final boundaries are completed using a contour tracking algorithm.

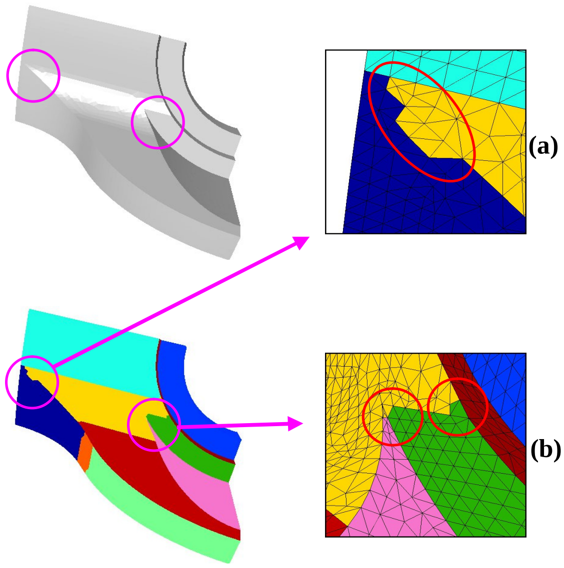
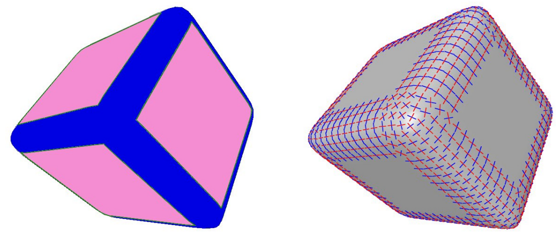


Figure 12. Zooms on artefacts (region boundary precision mistake) for the segmented Fandisk object.

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 [16]. 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.

Fig.13.a shows a natural hand made segmentation of a smooth cube object into constant curvature patches. Fig.13.b shows maximum and minimum curvature directions.



(a) **(b)**

Figure 13. Natural constant curvature patches of the Cube object (a) and its principal curvature directions (b), *dmin* in red and *dmax* in blue.

Boundaries of the patches follow the minimum directions, except around isotropic region (at the corners of the cube). Therefore the angle between a boundary edge and its vertices minimum curvature directions can represent a good evaluation of its "correctness".

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

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

w_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 $Jmin_{i1}$ and $Jmin_{i2}$ (see Fig.14) between the edge e_i and its vertices minimum directions. The score also consider the angles $Jmax_{i1}$ and $Jmax_{i2}$ 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, like the corner of the cube for instance (see Fig.13). Thus the angle score Sa is processed according to the following equation (6):

$$Sa(e_i) = \frac{\left(Jmin_{i1} \times Kmax_{i1} + Jmax_{i1} \times Kmin_{i1}\right)}{Kmax_{i1} + Kmin_{i1}} + \frac{\left(Jmin_{i2} \times Kmax_{i2} + Jmax_{i2} \times Kmin_{i2}\right)}{Kmax_{i2} + Kmin_{i2}}$$

with $Jmin_{i1}, Jmin_{i2}$ and $Jmax_{i1}$, $Jmax_{i2}$ 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 .

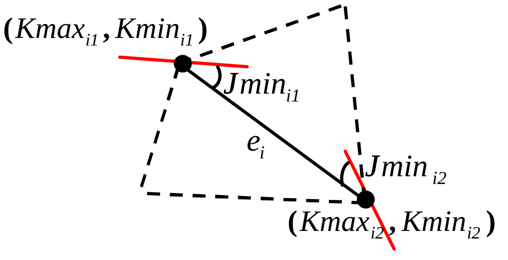


Figure 14. Elements taken into account for the calculation of the *Boundary Score* of the edge e_i .

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):

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

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.16.c and Fig.17.c show this marking process, starting from the region segmentation (see Fig.16.a, Fig.17.b), *CBEs* are represented in green, and others in red.

4.3.2. 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 final object patches 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.15.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 $J(e_i, e_{CBE})$ with its neighboring *CBE*, because, we try to limit the deviation of the boundary.

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

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

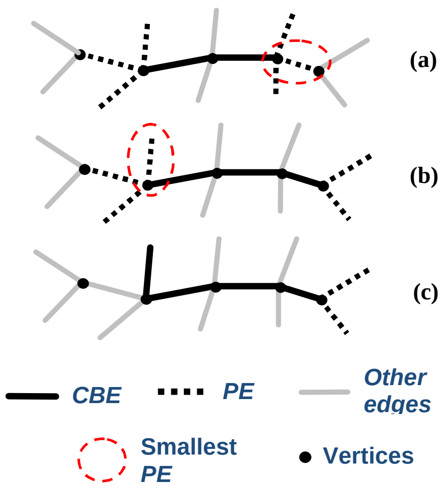


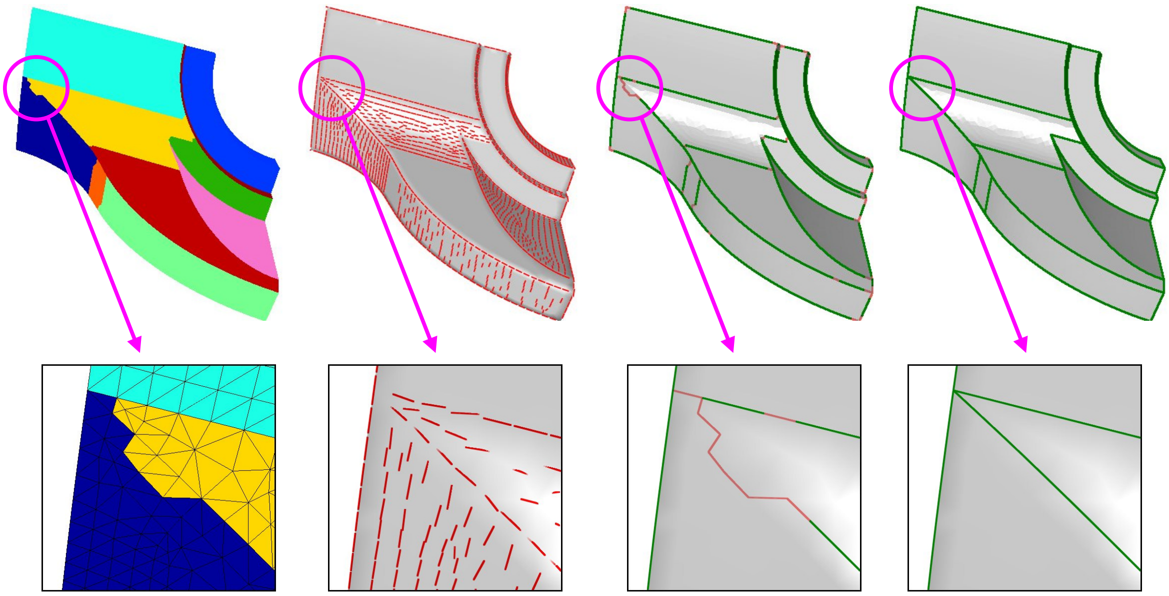
Figure 15. Three steps (a,b,c) of the boundary tracking algorithm, with associated position of correct boundary edges (*CBE*), potential edges (*PE*) and smallest potential edge, at each iteration.

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 boundary 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.15 presents three iterations of the contour tracking algorithm. In Fig.15.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.15.b). The position and number of the *PEs* is thus updated. The process continues in Fig.15.c, with another *PE* integrated to the contour.

5.4. Experiments and results

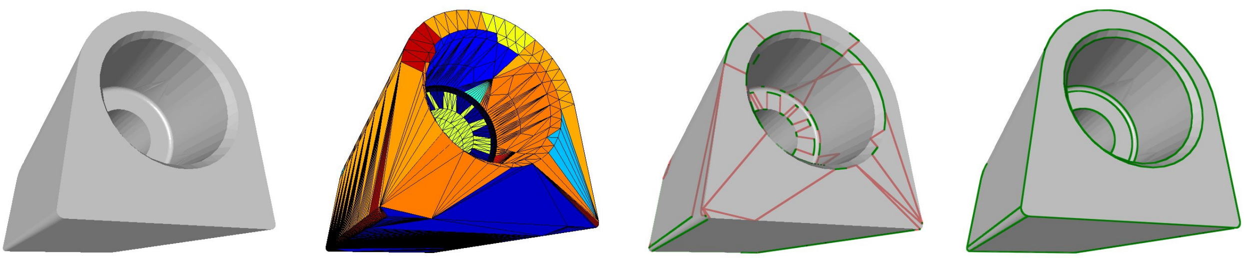
The rectification method is especially adapted to CAD or mechanical objects, where there exist real defined regular boundaries. On natural or organic objects the fact of rectifying boundaries does not have a real signification since even a human hand could not trace precise and clear boundaries.

We have tested our rectification method on various models issued from our region segmentation method. Fig.16 presents results for “Fandisk”. Artefacts coming from the region segmentation are all suppressed; we obtain surface patches with very clean and regular 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.17 shows results on a bad segmented object.



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

Figure 16. 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.



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

Figure 17. 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 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. This rectification process is independent of the region segmentation method presented in section 4; we can imagine using it as a contour tracking post process to hard edges detection methods for example.

6. Conclusion

This paper presents an original segmentation method to decompose a 3D-mesh into near curvature constant 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 transmits regions information from vertices to triangles, and thus allows to obtain clearly distinguishable edges as boundaries contrary to the traditional Watershed based methods which consider only vertices.
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 critical parameter of our method. This work is part of a larger compression process. The objective is to fit the segmented sub-surfaces with subdivision or parametric surfaces, in order to obtain the object in the form of a set of ”light” patches, which will allows 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] R. Hoffman and Jain, AK., "Segmentation and classification of range images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1987, vol. 9, no. 5, pp. 608-620.
[2] P. Besl, *Surfaces in Range Image Understanding*, New York, USA, Springer-Verlag, 1988.
[3] H. Rom and G. Medioni, "Part decomposition and description of 3d shapes.", *International Conference on Pattern Recognition*, Jerusalem, Israel, 1994, vol. 1, pp. 629-632.
[4] A. Leonardis, A. Jaklic and F. Solina, "Superquadrics for Segmenting and Modeling Range Data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997, vol. 19, no. 11, pp. 1289-1295.
[5] L. Chevalier, F. Jaillet and A. Baskurt, “A. Segmentation and superquadric modelling of 3D objects”, *WSCG*, Plzen - Bory, Czech Republic, 2003, vol .11, no. 2, pp. 232-40.
[6] R. Chaine and S. Bouakaz, "Segmentation of 3-D Surface Trace Points, Using a Hierarchical Tree-Based Diffusion Scheme.", *Fourth Asian Conference on Computer Vision ACCV2000*, Taiwan, 2000, vol. 2, pp. 995-1002.
[7] K. Wu and M. Levine, "3d part segmentation using simulated electrical charge distributions.", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1997, vol. 19, pp. 1223-1235.
[8] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
[9] 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.
[10] 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, 825-28.
[11] 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.
[12] 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. 273-76.
[13] D. Cohen-Steiner and J. Morvan, "Restricted delaunay triangulations and normal cycle", *19th Annu. ACM Sympos. Comput. Geom.*, 2003, pp. 237-246.
[14] 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.
[15] G. Taubin, "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation", *Fifth International Conference on Computer Vision*, 1995, pp. 902-907.
[16] 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.
[17] D. Alexander and J. Gee, "Elastic Matching of Diffusion Tensor Images", *Computer Vision and Image Understanding*, 2000, vol. 77, pp. 233-250.
[18] 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.

[19] A. Gersho and R. Gray , *Vector Quantization and Signal Compression*, Boston, USA, Kluwer Academic Publishers, 1992.
[20] K. Saarinen, "Color image segmentation by a watershed algorithm and region adjacency graph processing.", *IEEE International Conference on Image Processing*, Austin, TX, USA, 1994, pp. 1021-1024.
[21] K. Idrissi, G. Lavoué, J. Ricard and A. Baskurt, "Object of Interest based visual navigation, retrieval and semantic content identification system", *Computer Vision and Image Understanding, Special issue on Color for Image Indexing and Retrieval*, 2003.
[22] R. Schettini, "A Segmentation Algorithm For Color Images", *Pattern Recognition Letters*, 1993, vol. 14, pp. 499-506.