Courbure discrète : théorie et applications

Rencontre organisée par :
Laurent Najman and Pascal Romon

18-22 novembre 2013

Alexandra Bac, Jean-Luc Mari, Dimitri Kudelski, Nam-Van Tran, Sophie Viseur, and Marc Daniel

Application of discrete curvatures to surface mesh simplification and feature line extraction

Vol. 3, n° 1 (2013), p. 31-49.

<http://acirm.cedram.org/item?id=ACIRM_2013__3_1_31_0>

Centre international de rencontres mathématiques
U.M.S. 822 C.N.R.S./S.M.F.
Luminy (Marseille) France

Actes des rencontres du C.I.R.M.
Vol. **3** n° 1 (2013) 31-49

Application of discrete curvatures to surface mesh simplification and feature line extraction

Alexandra Bac, Jean-Luc Mari, Dimitri Kudelski, Nam-Van Tran, Sophie Viseur, and Marc Daniel

Abstract

We present two applications of discrete curvatures for surface mesh processing. The first one deals withÊsimplifying a mesh while preserving its sharp features. The second application can be considered as a dual problem, as we investigate ways to detect feature lines within a mesh. Both applications are illustrated with valuable results.

1. Introduction

Estimating shape of discrete objets known by a triangular approximating mesh or even by a point cloud is a relevant problem in the numerous software handling 3D objets. The problem has rather old origins, since one finds its first elements in the works of Gauss and Legendre. The first recent work on the subject was proposed by Alexandrov ([2]). Shape analysis are based on discrete curvature computations and different approaches exist to obtain these second order estimators (see for example [1] for the description of some estimators and results about convergence published in the literature).

We present in this paper two disconnected applications of discrete curvatures for surface mesh processing to illustrate the wide range of information which can be received from theses estimators. The first one deals withÊsimplifying a mesh while preserving its sharp features. Through the quadratic error metric introduced by Garland et al.,Êsuch a simplification can be performed by an edge collapse process guided by the metric. Such an approach leads to highÊquality simplification but remains slow and costly both in terms of space and time. We introduce a two-step method in whichÊwe perform an initial adaptive cell segmentation guided by the curvature and direction of each cell (computed by PCA).ÊThis pre-segmentation according to local curvatures preserves the quality of simplified meshes while reducing computingÊtime by a factor 3 to 4. The second application can be considered as a dual problem, as we investigate ways to detect feature lines within a mesh. Robust extraction of the feature lines of aÊ3D surface model is a challenging problem. Classical approaches generally rely on curvature derivatives, leading to the detection of a salient part as multipleÊsegments despite the fact that it visually appears as a single and fully connected element. We propose a two-step method aiming at extracting feature lines onÊ3D meshes with connectivity preservation. First, all the mesh vertices are labeled according to their curvature values in order to construct regions of interestÊon the discrete surface. The second step consists in a skeletonization directly on the mesh that corresponds to a homotopic thinning of the previouslyÊbinarized areas. Consequently, the resulting lines are highly connected due to the topological properties of the thinning operator.

Text presented during the meeting "Discrete curvature: Theory and applications" organized by Laurent Najman and Pascal Romon. 18-22 novembre 2013, C.I.R.M. (Luminy).
Key words. geometric modeling, discrete curvature, feature extraction, mesh processing.

2. Surface Mesh Segmentation

2.1. **Context.** Our work originates in the study of triangular mesh surfaces originated from geology and geologic surface modelling (as part of a collaboration with the IFP - French Institute of Petroleum). Our data, obtained by physical measures, are typically inhomogeneous, sparse, noisy and voluminous. Therefore, we are interested in the improvement of such surfaces and more particularly in the detection and filling of holes and faults. However, most improvement algorithms are both time and space consuming and thus, it is fundamental to simplify, smooth and homogenize data before any further treatment while preserving curvatures and critical areas such as faults (see [4]).

The present work was undertaken in this context: our hybrid mesh simplification method allies both vertex clustering and iterative edge collapse techniques. These approaches are actually complementary: iterative edge contraction (based on quadratic error metrics, see [10], ([11]), compared to vertex clustering approaches, leads to results of good quality but proves very costly both in terms of time and space. Vertex clustering algorithms (see for instance [20], [7], [17]) are simple, light and efficient methods but they hardly take into account the local geometry of the surface. Therefore, our idea was to combine both an adaptive segmentation step followed by an iterative edge collapse process (this last step ends when the expected simplification rate is reached).

The paper is organized as follows: in section 2.2, we introduce our two step method, while sections 2.3 and 2.4 respectively detail each step. Section 3.4 emphasizes the very interesting results we obtained.

2.2. **Method General presentation.** Our work starts from an observation: the approaches to triangular mesh simplification are various and actually each of them is relevant in its own field. On the one hand, vertex clustering approaches are particularly interesting in terms of time and space consumption and will be more efficient for low simplification rates. On the other hand, iterative edge contraction is slower and requires more memory, but produces better results (specially for high simplification rates).

The purpose of our algorithm is to conciliate the advantages of both approaches in order to efficiently handle models of any size while preserving the quality of the resulting approximations.

The underlying idea of our algorithm is to combine a first adaptive segmentation step with a second iterative edge collapse step.

2.2.1. *Vertex grouping: spatial adaptive clustering.* The first step of our algorithm consists in a vertices grouping step. As we have explained previously, in order to obtain satisfactory results, it is necessary to take into account the local geometry of the surface and hence to use an adaptive approach. However, if the original data is inhomogeneous and if some areas of the original surface are sparse, a purely adaptive approach can lose too many informations in these areas. Therefore, in order to avoid such problems, our algorithm starts from a rough regular grid. This initial grid is then refined by successive approximations: splitting planes are determined by a principal component analysis and inserted in the cells where more detail is necessary (see section 2.3.2).

In order to split cells efficiently, it is necessary to define a priority for their treatment. We chose to estimate the absolute curvature at each vertex (we use the estimation by Meyer et al. [19], see section 2.3.1). The indicator attached to a cell is the sum of the absolute curvatures of its vertices; cells are processed according to this indicator.

Last, a representative vertex is computed for each cell (by minimization of the quadratic error metric associated to the cell), and a topology is rebuilt over these vertices, inherited from the initial topology (see section 2.3.2).

2.2.2. *Iterative edge collapse.* Starting from the intermediate approximation of the mesh obtained by vertices grouping together with the quadratic error matrices previously computed, an iterative edge collapse process is applied in order to produce a smaller and smoother simplification (see section 2.4).

2.3. **Vertex clustering : adaptive segmentation.**

2.3.1. *Discrete curvatures.* A triangular mesh is a piecewise linear surface. Therefore, its curvature (in the sense of differential geometry) is null everywhere except on the edges where it is not defined. However, it can be interesting to consider such a surface as a discrete approximation of a continuous surface. In this perspective, one can define discrete curvature indicators; ideally these discrete indicators should converge to the continuous ones as the mesh density increases. Several definitions have been proposed for such discrete curvature indicators (see [8], [19], [24]). We chose to use the definition by M. Meyer and al. ([19]) as it constitutes a good trade-off between quality and complexity (convergence results have been formally obtained by G. Xu in [26]).

For any vertex v, we use Meyer's estimates to compute both mean curvature H and Gaussian curvature K at v. Let κ_1 and κ_2 be the principal curvatures at vertex v, then: $\kappa_1\kappa_2 = K$ and $\kappa_1 + \kappa_2 = 2H$. Therefore κ_1 and κ_2 are the roots of the polynomial $X^2 - 2H \cdot K + K$. The absolute curvature at v is defined by: $K_{\text{abs}} = |\kappa_1| + |\kappa_2|$. In our algorithm, this indicator is used throughout the vertex clustering process. Figure 2.1 presents absolute curvature fields for both a geological surface and the well known rocker arm model.

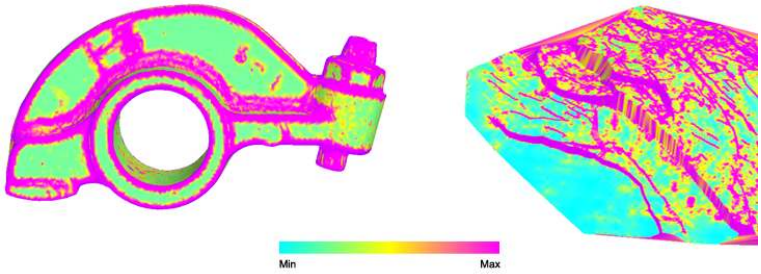

Figure 2.1: Discrete absolute curvature fields: left, a rocker arm - right, a geological surface

2.3.2. *Adaptive segmentation.* The spatial vertex partition is technically handled using a forest of BSP trees in order to control efficiently the size of the resulting mesh. Provided that each leaf of the BSP trees eventually produces a vertex, the leaves of the BSP tree are subdivided until the desired number of vertices is reached.

This process consists of three steps: initialization, adaptive segmentation, and last post-processing. Let us now detail each of them.

Initialization. After loading the mesh, the initialization step consists both in regularly segmenting the surface (subdividing the whole mesh by a 3D regular grid) and in computing for each vertex the corresponding absolute curvature indicator. The number of trees created corresponds to the number of cells of the uniform grid used for segmentation. Each root of this forest maintains a list of vertices and an absolute curvature value (defined as the sum of the absolute curvatures at the vertices of the cell).

Note that the size of the uniform grid does not directly control those of the resulting segmented mesh: this control arises from the adaptive segmentation step.

When the input data are voluminous, it is important that the size of the regular grid cells be small enough to simplify and accelerate the adaptive segmentation step. Moreover, in the sparse areas, the initial uniform clustering step prevents that too distant vertices be grouped by adaptive segmentation (which would result in distortions).

Adaptive segmentation of the mesh. Once the surface has been segmented by means of a regular grid (as described previously) we obtain an array of n BSP trees (where n is the number of cells of the initial regular grid). Moreover, these trees are sorted in a priority queue ordered by decreasing absolute curvature value.

The BSP tree is then iteratively updated as follows (let n be the number of leaves of the forest and let m be the number of vertices required for the simplified mesh):

While $n < m$:

(1) *chose the leaf of maximal absolute curvature*
(2) *create a subdivision plane by PCA analysis*
(3) *subdivide the leaf according to this plane and update the BSP tree*

In order to determine a **subdivision plane** appropriate to the repartition of vertices in the cell (see [11]), we use a principal component analysis of the normals of the cell (see [13]).

Let us recall the main results on principal component analysis. Let $\{x_1, \ldots, x_n\}$ be a set of vertices. The covariance matrix of this set is defined by:

$$Z = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^\top$$

where $\bar{v}$ denotes the average of the set $\{x_1, \ldots, x_n\}$.

The eigenvectors of this matrix give the main variation directions of the set of vectors (for a cloud of points inscribed in a rugby ball, these directions are the axes of the ball). The eigenvector associated to the largest (resp. smallest) eigenvalue corresponds to the direction in which vectors spread out[1] the most (resp. the least).

In our setting, at each step of the adaptive process, the strongly bent cells are split in order to decrease their curvature as much as possible. Ideally, the subdivision plane should be orthogonal to the direction of maximal curvature (see figure 2.2). However, contrarily to figure 2.2, we are not interested in smooth surfaces but in cells issued from a triangular mesh. Therefore, it is necessary to find a discrete approximation of principal directions.

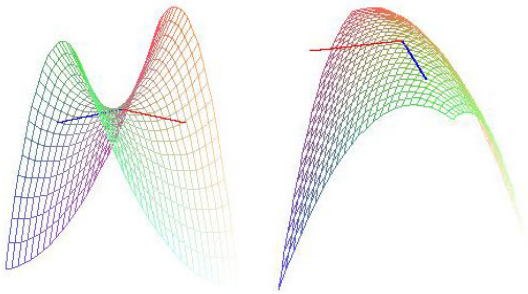

Figure 2.2: Principal curvatures on smooth surfaces: (in red, direction of maximal curvature - in blue, direction of minimal curvature)

Normal curvature in direction τ is the normal component of acceleration in this direction. Therefore, principal directions correspond to directions (in the tangent plane) of minimal and maximal variation of the normal vector.

In the discrete case, principal component analysis of the set of normals of the cell provides the main spreading directions of this set. Let e_1, e_2 and e_3 be unitary eigenvectors of the covariance matrix, associated to eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$ (eigenvalues and eigenvectors are computed with the Jacobi method [25]) . Direction e_1 is that of minimal variance, therefore it approximates the average normal vector of the cell. Direction e_3 (orthogonal to e_1) is that of maximal variance. Thus it approaches the principal direction of maximal curvature and we will take e_3 **to be the normal of the splitting plane**.

Moreover, the affine subdivision plane should be inserted around the vertex of maximal curvature; but in order to split the cell efficiently, this vertex should not be too close from the border. Therefore, we insert the splitting plane at the **barycenter of the vertices weighted by their absolute curvature**. The resulting clustering is quite satisfactory both for large and small cells (see figure 2.3).

Once the subdivision plane is determined, the leaf corresponding to the considered cell in the BSP tree is split into two new leaves. Vertices of the original cell are assigned to one of these leaves depending on their position with respect to the splitting plane. Then, we assign each triangle to the set of cells its vertices belong (thus, a triangle generally belongs to up to three cells). The discrete surfaces we are studying are topologically connected. However, nodes can contain distinct disconnected components. In such a case, replacing the vertices of the cell by a single vertex would produce a non-manifold mesh; thus we test the connectivity of nodes and eventually split the non connected leaves into their connected components.

[1]The direction in which vectors spread out the most is actually the direction of maximal variance

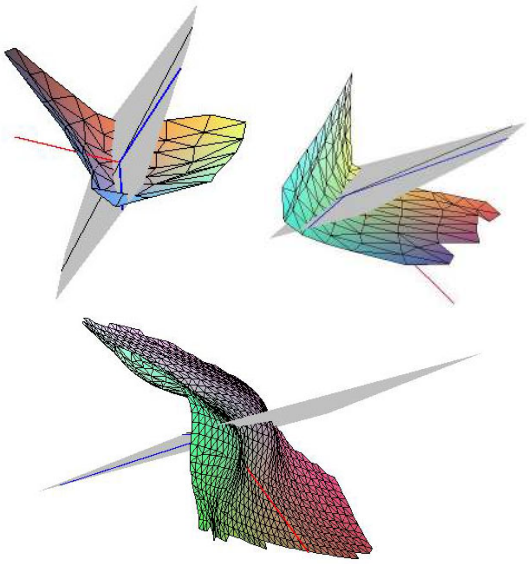

Figure 2.3: Splitting planes for small cells (top) and a 300 vertices cell

The following test is applied to each leaf of the BSP tree; the algorithm uses a list L (initially containing all the vertices of the leaf) and a queue f (initially empty).

- Get the head of L into v
- Insert v into f
- While f is not empty :
  - Get the head of f into v
  - For any v' neighbor of v:
    if v' belongs to L then
    - Insert v' into f
    - Remove v' from L

At the end of this test, if L is empty, the cell contains a single connected component and thus, the simplification process goes on normally. If L is not empty, the cell contains disconnected components. The leaf is split into two new leaves respectively containing the vertices still present in L and the others. The topological test goes on on the first set until all the connected components have been identified.

In spite of its cost, this test is necessary to guarantee the topological properties of the simplified surface. Figure 2.4 presents both the uniform cells and those obtained after the adaptive subdivision process.

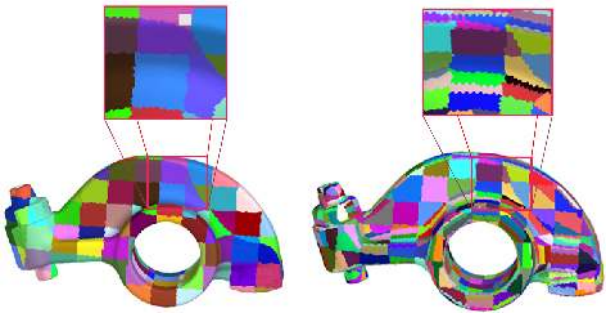

Figure 2.4: Results of the adaptive subdivision process: left, uniform clustering - right, adaptive clustering

Post-processing. Once the cells have been split and the expected decimation rate is reached, a representative vertex must be computed for each of them (together with an appropriate topology, inherited from the original mesh).

In order to approximate cells as precisely as possible, we use a method similar to [7], [17] and [22]. For each cell, we define a quadratic form (called *quadratic error metric*) estimating the distance between any point of space and the cell. The optimal position of the representative vertex is obtained by minimization of this quadratic form.

Let us now define this quadratic form. For any triangle t in the cell, let $\mathcal{P}_t$ be the plane defined by t, the quadratic form $Q_t : \mathbb{R}^3 \to \mathbb{R}$ associated to t is defined by $Q_t(v) = \mathrm{d}(v, \mathcal{P}_t)^2$. The cartesian equation of $\mathcal{P}_t$ can be written: $n^\top v + d = 0$ where n denotes the unitary normal of t and d is a constant. The distance $d(v, \mathcal{P}_t)^2$ can thus be written as $d(v, \mathcal{P}_t)^2 = v^\top (nn^\top)v + 2(dn^\top)v + d^2$. Let us define:

$$Q_t(v) = v^\top A_t v + 2B_t^\top v + C$$

with $A_t = nn^\top$, $B_t = dn^\top$ and $C_t = d^2$.

The quadratic form associated to a cell is the sum of the forms associated to each of its triangles. As a consequence, it can also be written: $\mathcal{Q}(v) = v^\top Av + 2B^\top v + C$. Figure 2.5 presents quadratic error metrics for different cells. The red axes represent the axes of Q; they originate at the point $v_{\min}$ minimizing Q (let $\varepsilon_{\min} = Q(v_{\min})$). The isosurface $Q = 1.5 \times \varepsilon_{\min}$ is represented in black.

Observe that the axes produced by the principal component analysis of the cell (represented in blue) are quite similar to the axes of the quadratic error metric[2].

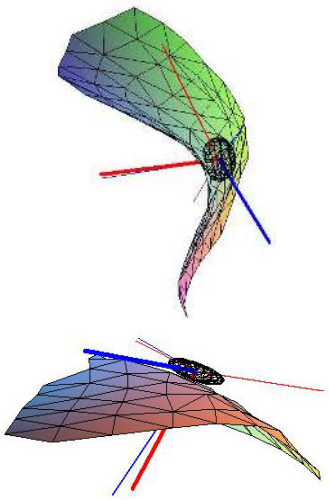

Figure 2.5: Quadratic error metric for different cells - top: a saddle cell - bottom: a convex cell

We have $\mathrm{d}Q(v).h = 0$ and as matrix A is symmetric and non negative, minimizing Q comes to solving $Av + B = 0$. This linear system is solved by singular values decomposition: $A = U\Sigma V^\top$ where Σ is a diagonal matrix and U and V are orthogonal matrices. Let us define matrix Σ^+ by:

$$(\Sigma^+)_{i,j} = \begin{cases} \frac{1}{\Sigma_{i,j}} & \text{if } \Sigma_{i,j} \neq 0 \\ 0 & \text{else} \end{cases}$$

Let $\hat{x}$ be the barycenter of the cell. The closest point to $\hat{x}$ satisfying equation $Ax + B = 0$ is given by:

$$x = \hat{x} - V\Sigma^+ U^\top (B + A\hat{x})$$

[2]Which is not so surprising as

$$A = \sum_{t \in \text{cell}} n_t n_t^\top \text{ whereas } Z = \frac{1}{k-1} \sum_{t \in \text{cell}} (n_t - \bar{n})(n_t - \bar{n})^\top$$

where n_t denotes the normal of triangle t and $\bar{n}$ the average normal of the cell.

Once this representative vertex is determined for each cell, it remains to rebuild a topology over these vertices, inherited from the initial topology of the surface. The algorithm is as follows:

For any face f in the initial mesh:

- *if f belongs to three different cells, it is kept,*
- *otherwise, it is degenerate (reduced to a segment or vertex in the new mesh) and therefore, it is removed.*

The remaining faces generate the topology over the set of representative vertices and the quadratic error metric of each cell becomes that of its representative vertex.

Let us point out that this post-processing (also used by [7], [17] and [22]) does not guarantee the manifoldness of the result (only that generally, it is manifold). The following example (figure 2.6) illustrates such a topological problem. The initial mesh (drawn in black on the left figure) is split into four cells and thus, the simplified mesh (in red) is not a manifold. Flipping edge (e, i) solves the problem (see right figure).

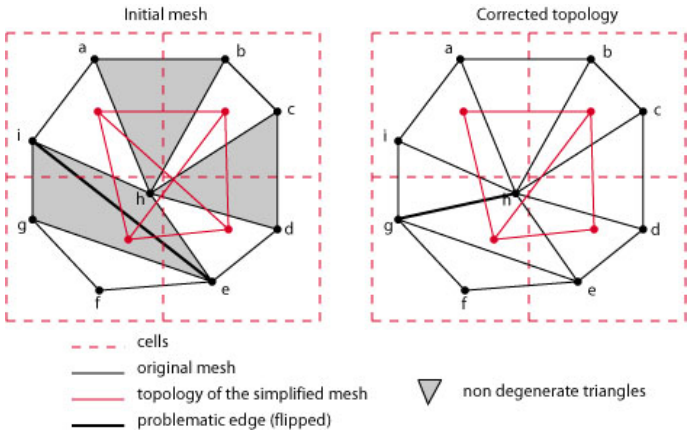

Figure 2.6: Heuristic for the well known topological problem (non-manifoldness): left, the original mesh - right, the corrected mesh (an edge has been flipped) which gives rise to a manifold simplified mesh

Our idea is to detect and avoid edges causing non-manifoldness, and actually, edges of the original mesh belonging to two triangles that will be non degenerate are one of the main cause for such problems (as they produce crossing edges). Therefore, before building the topology of the simplified mesh, we apply the following heuristic to the initial mesh:

(1) *select the edges (v_1, v_2) of the initial mesh incident to two different non degenerate triangles ((v_1, v_2, v_3) and (v_1, v_2, v_4)); these edges are responsible for non-manifoldness*

(2) *for each of these edges:*
if (v_3, v_4) belongs to a single cell
flip (v_1, v_2) ((v_1, v_2) is replaced by (v_3, v_4)):

In the previous example, only edge (e, i) is concerned and its flip makes the simplified mesh a manifold surface.

All this data (representative vertices, topology and quadratic error metric) is transmitted to the second step of our simplification algorithm.

2.4. **Iterative edge collapse.** The second step of our algorithm consists in simplifying more finely (by iterative edge collapse) the intermediate mesh previously obtained. We apply the method introduced by Garland and al. ([10]) with the quadratic error metrics previously computed.

Contracting a pair of vertices $(v_1, v_2) \to \bar{v}$ consists in replacing the vertices v_1 and v_2 by a new vertex $\bar{v}$ minimizing the resulting error (where error is measured with the quadratic error metric just described). Vertex $\bar{v}$ is then linked with the neighbors of v_1 and v_2.

Let us now come into details. The quadratic error made on the edge (v_1, v_2) is estimated by $Q_{(v_1,v_2)}(v) = Q_{v_1}(v) + Q_{v_2}(v)$. The algorithm is as follows:

- *For any edge* (v_1, v_2)*, compute* $\bar{v}$ *the vertex minimizing error* $Q_{(v_1,v_2)}(v)$*. The cost of contraction* $(v_1, v_2) \to \bar{v}$ *is defined as* $Q_{(v_1,v_2)}(\bar{v})$*.*
- *Order the pairs in a stack by increasing order.*
- *While the desired decimation rate is not reached:*
  - *remove the pair* (v_1, v_2) *of lower cost from the stack,*
  - *contract this pair; the quadratic error metric associated to the new vertex* $\bar{v}$ *is* $Q_{\bar{v}} = Q_{v_1} + Q_{v_2}$
  - *update the contractions (position of the optimal vertices) and their costs for the 1-neighbor ring of* $\bar{v}$

2.5. **Results.** The performances of the simplification process strongly depend on the following parameters: first the size of the intermediate mesh (that is the simplified mesh obtained after the first step), second, the size of the uniform grid.

The size of the uniform grid must not be too small, otherwise, the following adaptive subdivision makes no more sense and wouldn't improve uniform segmentation anymore. However, this parameter provides a control over the errors made by adaptive segmentation: at worst, after the adaptive segmentation step, the size of the cells equals those of the grid. In practice, a good choice for the size of the cells is to take them between 1.5 and 2 times the average length of the edges. As for the size of the intermediate mesh, we experimentally choose a ratio between 0.5 and 0.8 of the size of the initial mesh. Both parameters must actually be chosen in order to let enough "place" to both steps to work over the data.

As one can observe (figure 2.7 and 2.8, the simplified surfaces are visually very satisfactory; actually, they are very close to those obtained by a pure iterative edge contraction - this will illustrated when studying the Hausdorff distance between the initial surface and the simplified one. Observe that the sharp edges are well preserved. For geological surfaces, it is essential as

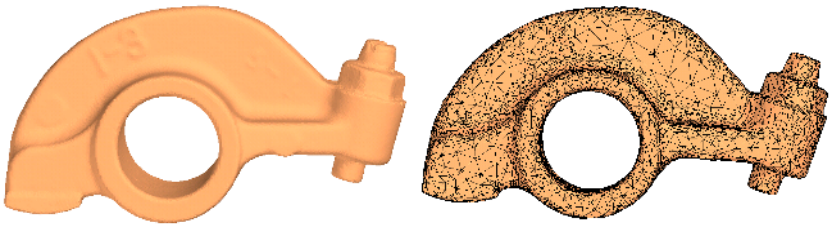

Figure 2.7: The rocker arm model simplified by our method: initial model, 40k vertices (left) - simplified model, 5k vertices, $D_{\max} = 0.00029$, $D_{\text{avg}} = 0.0000345$ (right) - size of the uniform grid: 41x24x78, size of the intermediate mesh: 20088 vertices

these characteristic lines are of particular interest for the geological interpretation of surfaces.

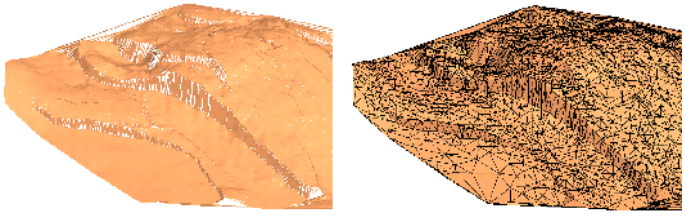

Figure 2.8: A geological surface simplified with our method: initial model, 112k vertices (left) - simplified model, 3k vertices - size of the uniform grid: 151x188x27, size of the intermediate mesh: 56136 vertices

In order to estimate the quality of our results, we have first compared them with those obtained by Shaffer and Garland with their mixed approach ([11]). The tests have been performed with two models: the "lucky lady" model (500k vertices) and the "dragon" model (437k vertices). Table 2.9 presents the numerical results obtained for this comparison.

Model	V_{in}	V_{out}	(1)/(2)	Grid size	Occupied cells	1st phase mesh size	Non-manifold edges	Time(s)	Error	Gain (2)/(1)
Venus	134k	20k	(1)	78x104x93	31464	31464	210	13	$D_{\text{avg}} = 0.17E-3$ $D_{\text{max}} = 4.75E-3$	
Fig. 9			(2)	62x84x74	20975	31507	84	14	$D_{\text{avg}} = 0.17E-3$ $D_{\text{max}} = 2.37E-3$	$G_{\text{avg}} = 2,13\%$ $G_{\text{max}} = 50,12\%$
Venus	134k	3k	(1)	62x84x74	20975	20975	190	12	$D_{\text{avg}} = 0.39E-3$ $D_{\text{max}} = 4.94E-3$	
			(2)	52x70x62	15072	20975	114	16	$D_{\text{avg}} = 0.38E-3$ $D_{\text{max}} = 2.80E-3$	$G_{\text{avg}} = 2,92\%$ $G_{\text{max}} = 43,46\%$
Lucky	500k	45k	(1)	182x311x105	119713	119713	5457	83	$D_{\text{avg}} = 0,21$ $D_{\text{max}} = 5,35$	
			(2)	145x249x84	79190	119700	1587	86	$D_{\text{avg}} = 0,19$ $D_{\text{max}} = 2,04$	$G_{\text{avg}} = 8,24\%$ $G_{\text{max}} = 61,93\%$
Lucky	500k	100k	(1)	242x414x139	197251	197251	4579	82	$D_{\text{avg}} = 0,12$ $D_{\text{max}} = 5,33$	
Fig. 10			(2)	182x311x105	119713	194725	866	96	$D_{\text{avg}} = 0,11$ $D_{\text{max}} = 1,16$	$G_{\text{avg}} = 7,20\%$ $G_{\text{max}} = 78,28\%$

(1) Garland and al. - (2) our method

- V_{in} is the size of the initial model and V_{out} is that of the simplified model
- *Grid size* is the size of the uniform grid
- *Occupied cells* is the number of leaves in the final BSP tree
- *1st phase mesh size* is the size of the intermediate mesh
- *Time (s)* is the running time of the whole simplification process
- *Error*: D_{max} is the Hausdorff distance between the original and the simplified mesh - D_{avg} is the average symmetric distance between both models (see[3] for more details)
- *Non-manifold edges* is the number of non-manifold edges resulting from vertex clustering (our heuristic aims at avoiding them)

Figure 2.9: Numerical comparison between our algorithm and Garland and al. 2002.

Figure 2.11 and 2.10 present the related graphical results. Observe that besides the numerical results, our method visually preserves well the sharp folds of the models and produces regular meshes.

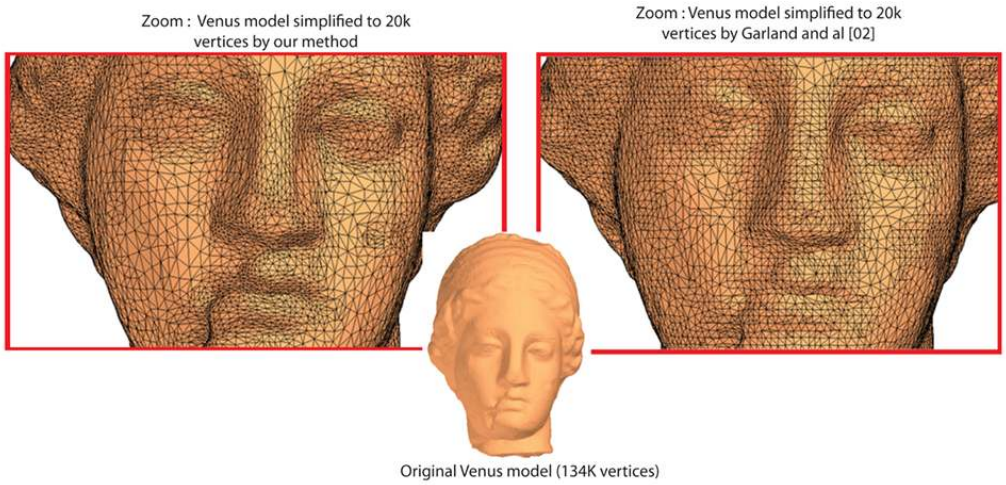

Figure 2.10: Comparison of our method and Garland and al. 2002 - the "venus" model (134k vertices) - simplified model: 20K vertices

In order to estimate the quality of our simplified meshes, we have compared them with surfaces obtained by the pure iterative edge collapse algorithm ([10]). Figure 2.12 presents running times and error maps for both of these algorithms.

Therefore, the quality of our results is similar to [10] whereas our running time is three times lower.

3. Feature Line Extraction

3.1. **Context.** The skeleton is a robust shape descriptor faithfully characterizing the topology and the geometry of an object. This notion is widely used for various applications such as video tracking [9], shape recognition [27], surface sketching [18], and in many other scientific domains. Several techniques have been proposed to extract the skeleton from binary 2D images [28], 3D

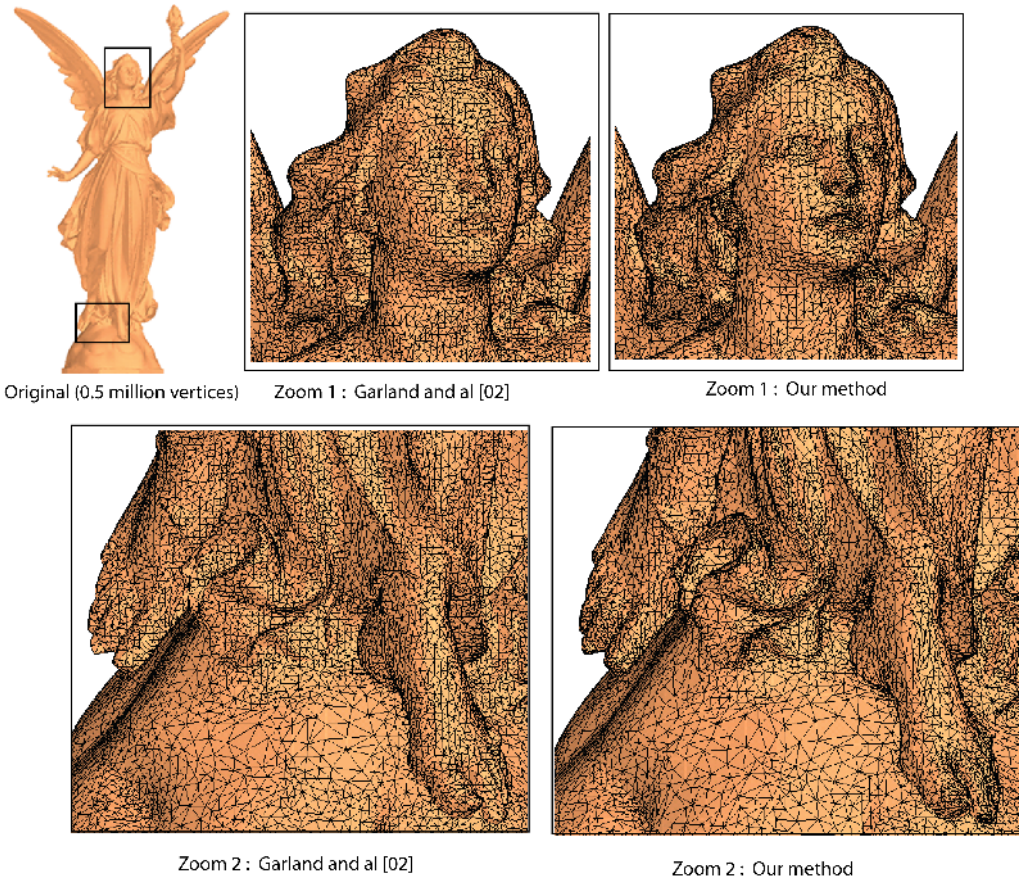

Figure 2.11: Comparison of our method and Garland and al. 2002 - the "lucky lady" model (500k vertices) - simplified model: 100K vertices

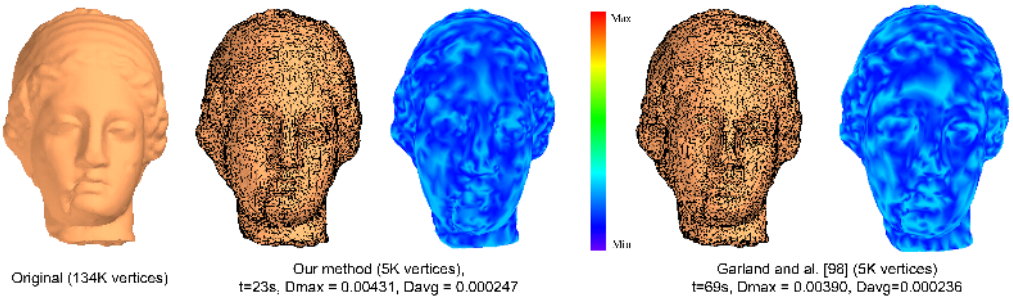

Figure 2.12: Comparison of the map of errors for our method and Garland and al. 1999 - the "venus" model (134k vertices)

closed meshes defining a volume [3], or 3D cubic grids [16]. However few have been dedicated to the extraction of skeletons from a binary information located on an arbitrary triangulated mesh. Rössl *et al.* [21] have presented a method in which some mathematical morphology operators have been ported to triangulated meshes. The main interest of this approach is to combine an efficient computation and a simple implementation. However, regarding the operator definitions and the underlying algorithm, several drawbacks have been pointed out which mainly lead to unexpectedly disconnected skeletons [15].

In this work, we propose a novel method to extract the skeleton of unstructured mesh patches

by a topological thinning process. To figure out the issues of skeletonization of heterogeneous and arbitrary triangulated meshes, we extend the concepts introduced in [21]. The presented approach herein strictly relies on the mesh connectivity to achieve the extraction of the final skeleton. Therefore, for the sake of understanding, the basic method of Rössl *et al.* is described in Section 3.2 with an assessment of its abilities and drawbacks. Section 3.3 details the proposed approach and introduces the additional definitions and the novel algorithm. The results of our method including tests on irregular meshes as well as on the performance of the algorithm are shown in Section 3.4. Finally, an application to feature line detection is presented in Section 3.5.

3.2. **Basic notions and definitions.**

3.2.1. *Position of the problem.* Let $\mathcal{S}$ be an arbitrary manifold surface represented by an unstructured mesh patch $\mathcal{M}$ such as $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$. The sets $\mathcal{V}, \mathcal{E}$, and $\mathcal{T}$ correspond, respectively, to the vertices, the edges, and the triangles composing $\mathcal{M}$, the piecewise linear approximation of $\mathcal{S}$. The vertices are denoted by p_i, with $i \in [0; n[$ and $n = |\mathcal{V}|$ being the total number of vertices of $\mathcal{M}$. The neighborhood $\mathcal{N}$ of a vertex p_i is then defined as following:

$$\mathcal{N}(p_i) = \{q_j \mid \exists \text{ a pair } (p_i, q_j) \text{ or } (q_j, p_i) \in \mathcal{E}\}. \tag{3.1}$$

In such a case, $m_i = |\mathcal{N}(p_i)|$ represents the total number of neighbors of p_i.

Let now consider a binary attribute F on each vertex of $\mathcal{V}$. The set $R \subseteq \mathcal{V}$ is then written as follows:

$$\forall p_i \in R \iff F(p_i) = 1. \tag{3.2}$$

The attribute F may be defined from beforehand process such as a manual selection, or a thresholding based on geometrical properties (triangle area, principal curvatures, *etc.*). Then, an edge $e = (p, q)$ belongs to R if and only if $p, q \in R$. Similarly, a triangle $t = (p, q, r)$ belongs to R if and only if $p, q, r \in R$.

The main objective is to finally develop a technique to extract the skeleton of the set R by using a topological thinning based on the mesh connectivity.

3.2.2. *The existing approach.* The skeletonization algorithm introduced by Rössl *et al.* consists in an iterative constraint thinning. This relies on a classification of each vertex of R. The authors proposed then three vertex types and $c(p_i)$, the *complexity* of the vertex p_i such as:

$$c(p_i) = \sum_{j=0}^{m_i - 1} |F(q_j) - F(q_k)|, \tag{3.3}$$

where $k = j + 1 \bmod m_i$ and $q_j, q_k \in \mathcal{N}(p_i)$.

Definition 1. A vertex p_i is considered as *complex* if and only if $c(p_i) \geq 4$. The set of all *complex* vertices is named C.

A *complex* vertex p_i thus potentially corresponds to a part of a skeleton branch if $c(p_i) = 4$, or a connection through several branches if $c(p_i) > 4$.

Definition 2. A vertex p_i is marked as *center* if and only if and $\mathcal{N}(p_i) \subseteq R$. The set of all *center* vertices is named E.

Definition 3. A vertex p_i is called *disk* if and only if $\exists q_j \in \mathcal{N}(p_i), q_j \in E$ that is a *center*. The set of all *disk* vertices is named D.

A *disk* vertex corresponds to a *simple* point: a point that does not modify the expected skeleton topology if it is removed [6]. We denote $\overline{X}$ the complementary of the set X in the region R.

Definition 4. The *skeletonization operator* of R is defined as a constrained thinning:

$$skeletonize(R) = R \setminus (D \cap \overline{C \cup E}). \tag{3.4}$$

After applying the skeletonization operator until idempotence on R, the set of the remaining vertices, corresponding to the final *skeleton*, is called Sk_R. During each pass, the skeletonization operator removes the boundary *disk* vertices. Figure 3.1 illustrates the execution of the algorithm. After obtaining the skeleton Sk_R of R, it is possible to remove the smallest branches. This last operation is called *pruning* and defined as follows:

$$prune(Sk_R) = Sk_R \setminus \overline{C}. \tag{3.5}$$

This pruning step is shown by Figure 3.1 (d).

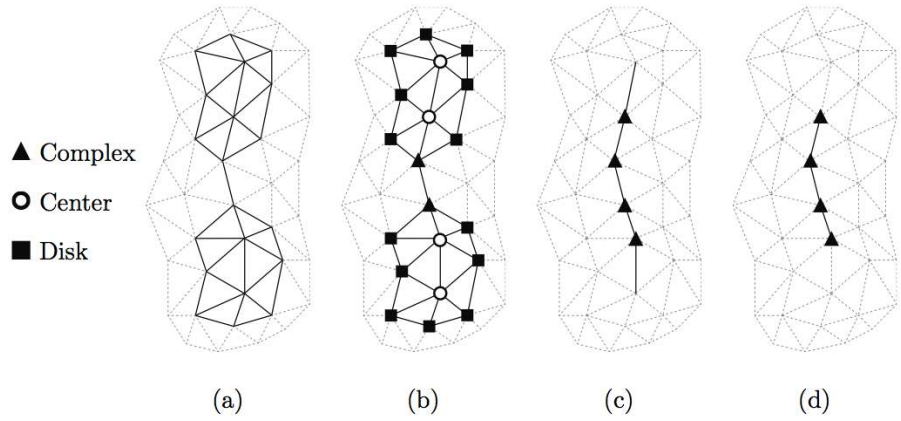

Figure 3.1: Illustration of the Rössl *et al.* algorithm. From left to right: (a) a set of vertices R, (b) classification of R, (c) thinning until idempotence, and (d) resulting skeleton after pruning.

3.2.3. *Result assessment.* Due to the simplicity of the used operators, the computational time of the Rössl *et al.* method is very low, and the skeleton extraction is thus almost instantaneous on meshes composed of 50K triangles. However, the accuracy and the continuity of the obtained skeleton deeply depends on the mesh configuration. In other words, a same set R defined on two different triangulations of S could lead to skeletons with two topologies drastically different. Moreover, the lack of continuity also occurs in the case of particular configurations that are shown in Figure 3.2 because the removal of *disk* vertices can modify the topology of the skeleton. Figure 3.3 illustrates the unexpected results and disconnections generated by the execution of the skeletonization. Once the vertices P_1 and P_2 are removed (b), the skeleton becomes disconnected at this location (c). However, some vertices would change to *complex* if a new classification step was applied. This kind of vertices represents relevant points in a topological point of view and thus, should not be deleted.

Another issue occurs since pruning is applied: the ending vertices of the skeleton are removed. As a matter of fact, when the set R contains no *center* and no *complex* vertex, the pruning operator removes all the vertices. This case is illustrated by Figure 3.4.

3.3. **A skeletonization method for any arbitrary triangulated mesh.** Both a new definition of particular vertices and a new algorithm have been elaborated to solve the disconnection issues previously raised up in Section 3.2. These two key points of the approach we propose are successively presented below.

3.3.1. *Additional definitions.* The different classes of vertices proposed by Rössl *et al.* aim at describing the topology of R. However, they are not sufficient as there are still vertices that are unmarked and that are then not considered in the skeletonization. For this reason, we introduce the *outer* class.

Definition 5. A vertex p_i is marked as *outer* if and only if $F(p_i) = 1$ and $p_i \notin (C \cup D \cup E)$. The set of *outer* vertices is named O and is defined as follows:

$$O = R \setminus (C \cup D \cup E) \tag{3.6}$$

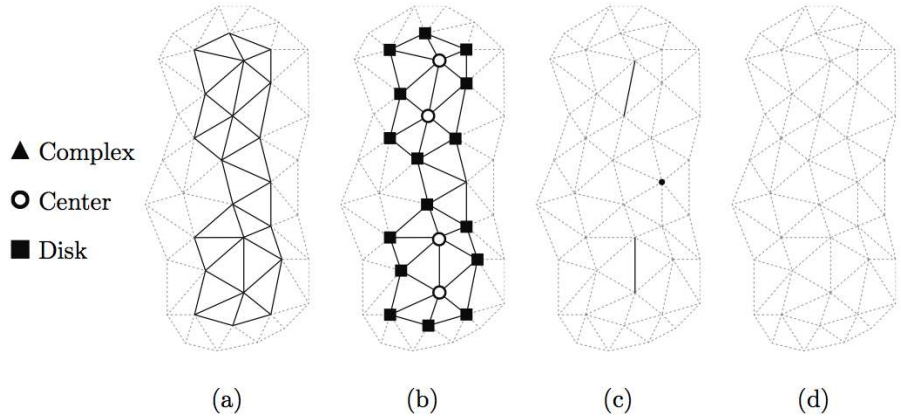

Figure 3.2: Example of unexpected results by applying the Rössl *et al.* method. From left to right: (a) the set of feature points R, (b) classification of R, (c) skeletonization of R, (d) resulting skeleton after pruning.

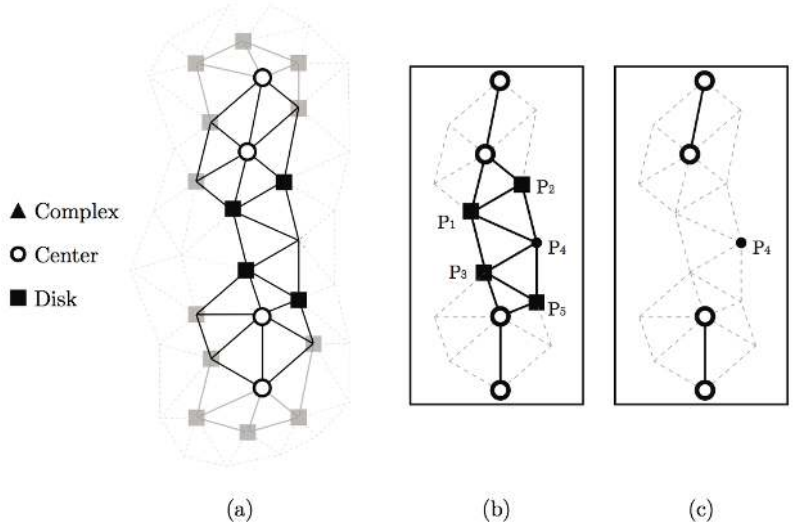

Figure 3.3: Execution of the skeletonization operator [21]: (a) vertex classification, (b) execution of the algorithm, (c) final skeleton with a broken topology.

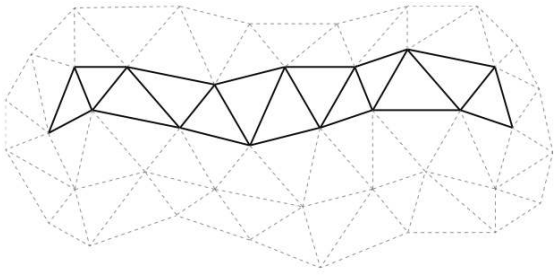

Figure 3.4: Example of a particular configuration: while the vertices of R are not classified, they will be deleted by the pruning operator of Rössl *et al.*

As it has been shown previously, a vertex may change from one class to another and, as a side-effect, this may lead to potential disconnections during the skeletonization. To counteract this issue, we propose to define a priority between the classes.

Definition 6. The *disk* class has a lower priority over the other classes.

If a vertex is already classified as *disk*, it can change to *complex*, *center* or *outer* if necessary.

3.3.2. *Algorithm.* If the skeletonization operator defined by Rössl *et al.* is directly applied to an unstructured patch, the final result may suffer from disconnections as some *disk* vertices are deleted while they characterize the topology of the object. To correct this issue, the algorithm we propose does not remove *all* the *disk* vertices but only those that will not be converted to a different priority class after the operator application. This requires to add an additional step in the algorithm: at each application of the skeletonization operator, the class of a vertex is recomputed before its deletion. For example, if a *disk* vertex becomes a *complex* vertex, the vertex is not removed.

However, the resulting skeleton may be too thick using this technique (*e.g.* if it is composed of only *outer* vertices). For this reason, a final cleaning step is added to obtain the expected skeleton. At this stage, the skeleton must be composed of *complex* vertices (*i.e.* the skeleton branches or nodes) and *outer* vertices, the ending points of the branches with only one *complex* vertex in their neighborhood. Thus, to obtain the final skeleton, a two steps process is applied:

- the *outer* vertices that have more than two neighbors belonging to R are removed;
- the *outer* vertices with at most one neighbor belonging to R are kept.

Moreover, as for the skeletonization operator, each vertex complexity change is checked before removing this vertex. Examples of resulting skeletons are shown in Figure 3.6 and the impact of the algorithm modification with the update step is presented in Figure 3.7: *disk* vertices are deleted (b) after checking their classes (c). During the deletion of P_1 and the update step, the class of P_2 changes from *disk* to *complex* and P_4 from *outer* to *complex*. Thus, these vertices are not removed and the extracted skeleton is fully connected and faithfully characterizes the topology of R (d). The complete method of skeleton extraction is summarized by the algorithm presented on Figure 3.5.

3.4. **Results.** Some results of skeleton extraction on meshes are presented in Figures 3.8, 3.9 and 3.10. The obtained skeletons describe the geometry and the topology of the original set R. The used meshes are relatively homogeneous in Figure 3.8 while, in Figures 3.9 and 3.10, the algorithm has been tested on irregular meshes to show the robustness of the proposed approach to unstructured meshes. It may be noticed that the resulting skeletons are the expected ones and reflect correctly the topology and geometry of the original set R in a proper way.

Moreover, since the definitions and the operators used to extract the skeleton are very simple, the computational time of the proposed approach is also very low, even if an additional checking step has been added. It is possible to process a mesh with 100K vertices in 1 second. The tests have been ran on an Intel Core 2 Duo 2.8 Ghz.

To complete the algorithm tests and to evaluate the robustness of the proposed approach, an application dedicated to the feature line detection is presented in the following section.

3.5. **Application to feature line detection.** The detection of features within 3D models is a crucial step in shape analysis. It is possible to extract from the surface of an object simple shape descriptors such as lines (drawn on the surface). Generally, the methods of feature line detection focus on the estimation of differential quantities and the research of curvature extrema. However, these techniques are based on *third-order* differential properties and it leads to a common issue: they produce disconnected feature lines because of flat and spherical areas and because of the noise present in data sets. Thus, it is particularly difficult to generate intersections between feature lines. To overcome these recurrent issues, we propose to apply our method to extract salient lines of a model.

In order to define sets over triangulated 3D meshes, we use the algorithm proposed by Kudelski *et al.* [14]. We compute the mean curvature H through a local polynomial fitting in the least-squares sense [12]. The binary attribute F is then defined at each vertex p_i as follows:

$$H_{p_i} > 0 \implies F(p_i) = 1. \tag{3.7}$$

repeat
  forall the *vertices* $p_i \in R$ **do**
    if p_i *is a* disk *vertex* **then**
      compute the complexity $c(p_i)$ of the vertex
      if *the priority of* p_i *does not change* **then**
        delete p_i
until *idempotence*

repeat
  forall the *vertices* $p_i \in R$ **do**
    if p_i *is an* outer *vertex* **then**
      compute the complexity $c(p_i)$ of the vertex
      if *the priority of* p_i *does not change and if* $|\mathcal{N}(p_i)| > 2$ **then**
        delete p_i
until *idempotence*

repeat
  forall the *vertices* $p_i \in R$ **do**
    if p_i *is an* outer *vertex* **then**
      compute the complexity $c(p_i)$ of the vertex
      if *the priority of* p_i *does not change and if* $|\mathcal{N}(p_i)| > 1$ **then**
        delete p_i
until *idempotence*

Figure 3.5: Extraction of the skeleton.

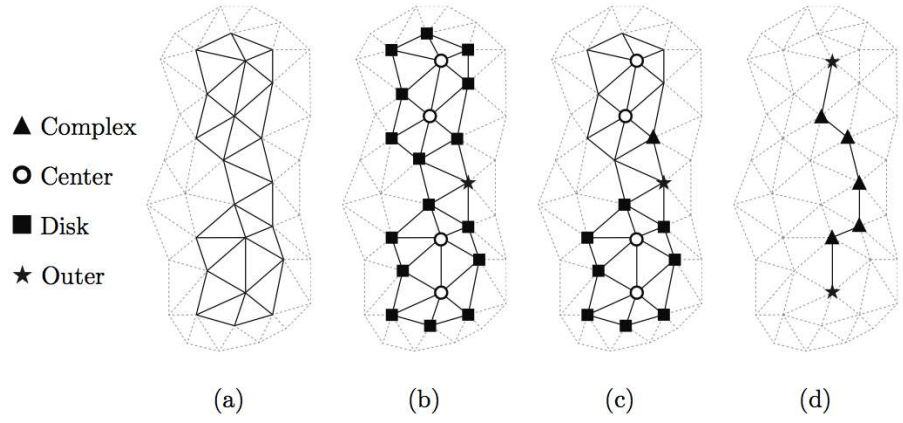

Figure 3.6: Illustration of the proposed approach: (a) region R, (b) vertex classification, (c) execution of the thinning algorithm with update, (d) final skeleton fully connected.

Finally, the objective is to thin the set, corresponding to potential feature parts of the mesh, in order to obtain lines describing the geometry and the topology of the object.

Figure 3.11 illustrates the process of feature line detection. The obtained characteristic lines are fully connected and describe accurately the topology of the sets. Then, due to the use of second-order differential properties (*i.e.*, the mean curvatures), the feature extraction is more robust.

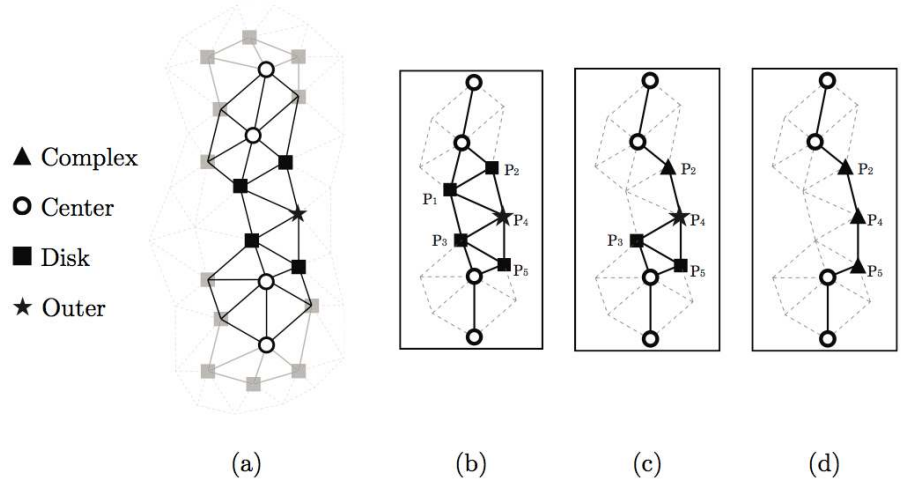

Figure 3.7: Detailed view of the thinning process: (a) vertex classification, (b) execution of the skeletonization operator, (c) update of vertex classes after deletion, (d) final skeleton.

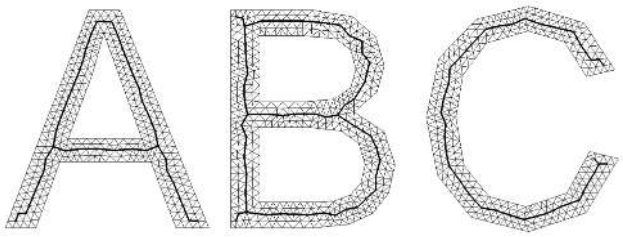

Figure 3.8: Application of the skeletonization algorithm on regular triangulated 3D meshes.

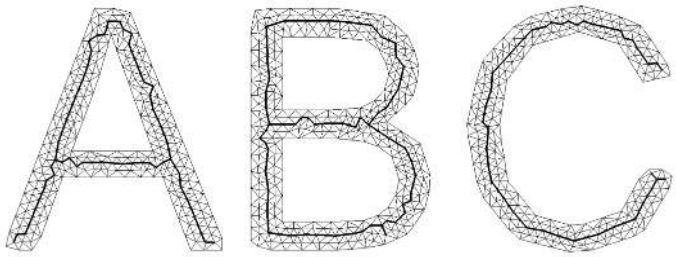

Figure 3.9: Skeleton extraction on irregular 3D meshes.

Moreover, this type of approach allows to generate intersections between feature lines, which is not possible with classical approaches (Figure 3.12).

Acknowledgments. The authors would like to thank the French Institute of Petroleum (IFP) for their financial support of the study exposed section 2 and Jean Borgomano and Yves Guglielmi of the Geology of Carbonate Systems and Reservoirs laboratory for their precious help and pieces of advice for the second application (section 3). The models in section 3.4 were provided courtesy of Caltech Multi-Res Modeling Group (Feline) and Cyberware (Dinosaur).

4. Conclusion

We illustrated the discrete curvatures concepts with two applications. In the first one, our algorithm proposes an alternative to vertex clustering simplification methods and to iterative edge collapse methods, by a compromise between both approaches. Regarding the results presented

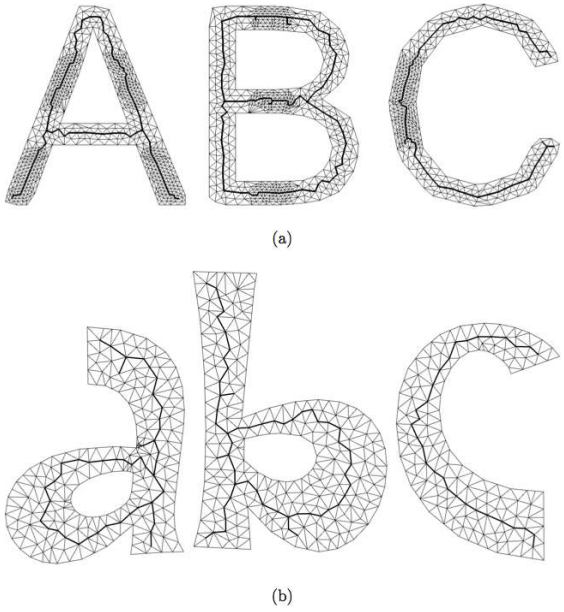

Figure 3.10: Extraction of the skeletons on meshes with mixed and unstructured meshes.

in section 2.5, this objective is reached. The main interest of this approach is to provide results of high quality (very similar to those obtained by an iterative edge collapse method) but with lower running times (by factors around 3 and up to 5) and memory consumption. Actually, our algorithms behaves well as for the average errors between the original and the simplified mesh and the maximal errors are significantly reduced compared to [11]. Moreover, the heuristic we apply in order to avoid the well known topological problems resulting from simplification based on vertex clustering proves quite efficient.

Our second application is an efficient and general new algorithm to extract the skeleton of a set R defined on a triangulated mesh by topological thinning. This approach relies on the definitions presented by Rössl *et al.* [21]. However, the latter generates, for some mesh configurations, unexpected skeletons that are generally more disconnected than they should. To overcome this issue, an additional definition of vertex categories has been added. Then, we have improved the thinning process by integrating a priority between vertex classes. Tests applied on different categories of meshes illustrate the efficiency of the approach. As future work, a formal proof based on [5] and issued from the notion of *simple vertices* (by analogy to *simple points*) may need to be considered. The Rössl *et al.* article does not include formal validations because the vertices classification is incomplete. With the changes made, the *disk* vertices truly correspond to simple points lying on a discrete 2-manifold. Thus it will be possible to transpose the notion of geodesic neighborhood to define topological numbers associated with simple vertices. A second prospect is related to the position of the skeleton nodes. Indeed, the defined operators do not integrate any geometrical information and the extraction of the skeleton only relies on a one-ring neighborhood. However, as the position of the skeleton is generally easier to correct than the topology, post-processing steps could be envisaged to optimize the skeleton position. In this way, the resulting skeleton will describe in a better way both the topology and the geometry of the set lying on the mesh.

Even if interesting and relevant results can already be obtained, we are aware than many theoretical works a practical experiments are still required to handle the open issues linked to large and noisy discrete objet analysis.

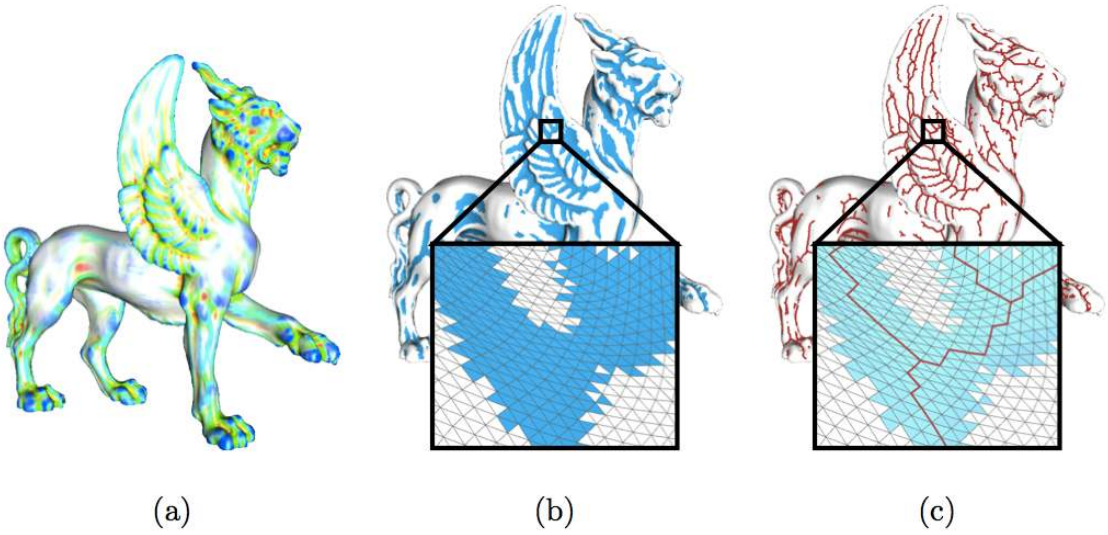

Figure 3.11: Algorithm of feature lines extraction: (a) curvature estimation, (b) definition of the set R, (c) extraction of lines from R by the proposed thinning approach.

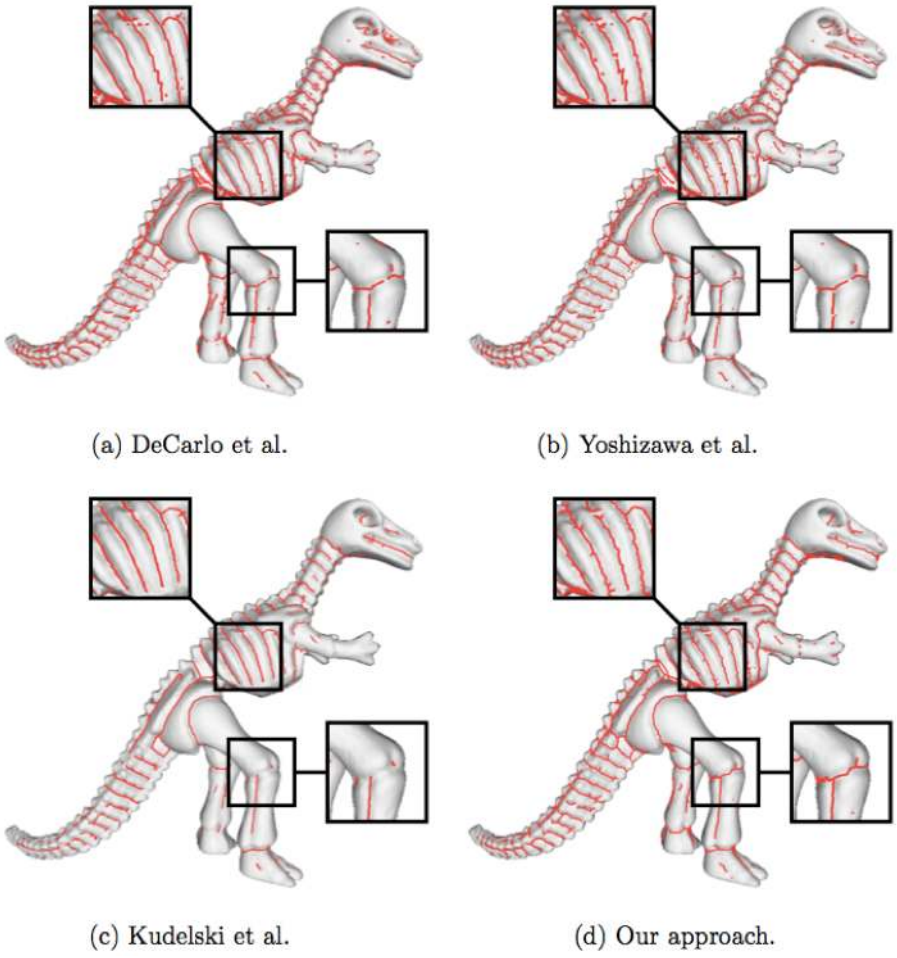

Figure 3.12: Comparison of results obtained from feature detection applied on *Dinosaur*.

References

[1] M. Daniel A. Bac and J.L. Maltret. 3D modeling and segmentation with discrete curvatures. *Medical Informatics and Technology*, pages 13–24, 2005.

[2] A. Alexandrov. Intrinsic geometry of surfaces. *Transactions of mathematical monograph AMS*, 1967.

[3] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Transaction on Graphics*, 27(3):1–10, August 2008.

[4] A. Bac, N-V. Tran, M. Daniel, and J-F. Rainaud. Traitement de surfaces géologiques pour la construction de modèles 3D. In *journées du GTMG*, pages 22–23, Poitier, 03 2005.

[5] Gilles Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Patterns Recognition Letters*, 15:pp. 1003–1011, 1994.

[6] Gilles Bertrand. A boolean characterization of three-dimensional simple points. *Pattern Recognition Letters*, 17:115–124, 1996.

[7] D. Brodsky and B. Watson. *Model Simplification In Reverse, Vector Quantization*. PhD thesis, University of Alberta, 2000.

[8] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Comput. Aided Geom. Des.*, 22(2):121–146, 2005.

[9] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 1746–1753. IEEE Computer Society, June 2009.

[10] M. Garland. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University, 1999.

[11] M. Garland and E. Shaffer. A multiphase approach to efficient surface simplification. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 117–124, Washington, DC, USA, 2002. IEEE Computer Society.

[12] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transaction on Graphics*, 23(1):45–63, 2004.

[13] I. Jolliffe. *Principal component analysis*. Springer Verlag, 1986.

[14] Dimitri Kudelski, Jean-Luc Mari, and Sophie Viseur. 3D feature line detection based on vertex labeling and 2D skeletonization. In *IEEE International Conference on Shape Modeling and Applications (SMI'10)*, pages 246–250. IEEE Computer Society, June 2010.

[15] Dimitri Kudelski, Jean-Luc Mari, and Sophie Viseur. Extraction of feature lines with connectivity preservation. In *Computer Graphics International (CGI'11 electronic proceedings)*, June 2011.

[16] T.C. Lee, R.L. Kashyap, and C.N. Chu. Building skeleton models via 3-D medial surface/axis thinning algorithms. *Graphical Models and Image Processing*, 56(6):462–478, November 1994.

[17] P. Lindstrom. Out-of-core simplification of large polygonal models. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 259–262, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[18] Jean-Luc Mari. Surface sketching with a voxel-based skeleton. In *15th IAPR International Conference on Discrete Geometry for Computer Imagery (DGCI'09)*, volume 5810 of *Lecture Notes in Computer Science*, pages 325–336. Springer, 2009.

[19] M. Meyer, M. Desbrun, P. Schroeder, and A.H. Barr. Discrete differential geometry operators for triangulated 2-manifolds. *VisMath*, 2002.

[20] J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complexe scences. *Geometric Modeling In Computer Graphics*, pages 455–465, 1993.

[21] Christian Rössl, Leif Kobbelt, and Hans-Peter Seidel. Extraction of feature lines on triangulated surfaces using morphological operators. In *AAAI Spring Symposium on Smart Graphics*, volume 00-04, pages 71–75, March 2000.

[22] E. Shaffer and M. Garland. Efficient adaptative simplification of massive meshes. *In Proceedings of IEEE Visualization 2001*, pages 127–134, October 2001.

[23] Kaleem Siddiqi and Stephen Pizer. *Medial Representations. Mathematics, Algorithms and Applications*. Computational Imaging and Vision, Vol. 37. Springer, 2008.

[24] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. *Fifth International Conference on Computer Vision*, pages 902–907, 1995.

[25] W. T. Vertterling, S. A Teukolsky, W. H. Press, and B. P. Flannery. *Numerical Recipe in C/C++, The Art of Scientific Computing*. 2003.

[26] G. Xu. Convergence analysis of a discretization scheme for gaussian curvature over triangular surfaces. *Comput. Aided Geom. Des.*, 23(2):193–207, 2006.

[27] Kai Yu, Jiangqin Wu, and Yueting Zhuang. Skeleton-based recognition of chinese calligraphic character image. In *Advances in Multimedia Information Processing (PCM'08)*, volume 5353 of *Lecture Notes in Computer Science*, pages 228–237. Springer, 2008.

[28] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, March 1984.

Aix-Marseille University, LSIS, UMR CNRS 7296, Marseille, France • alexandra.bac@univ-amu.fr • jean-luc.mari@univ-amu.fr • viseur@cerege.fr • marc.daniel@univ-amu.fr