A hybrid simplification algorithm for triangular meshes

Alexandra. Bac* Nam Van Tran† Marc Daniel‡
LSIS Marseille LSIS Marseille LSIS Marseille

Abstract

This paper deals with a hybrid simplification method for triangular meshes. Our approach is based on a first simplification step where vertices are clustered, followed by an iterative edge collapse step. More precisely, vertices are first clustered into surface patches through an adaptive segmentation process (using both absolute discrete curvature and principal component analysis); the edge collapse process is based on quadratic error metrics.

Keywords: Simplification, segmentation, adaptive, quadratic error metric, PCA, large data sets

1 Introduction

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 [Bac et al. 2005]).

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 [Garland 1999]), 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 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, we present related works in the field of simplification. In section 3, we introduce our two step method, while sections 4 and 5 respectively detail each step. Section 6 emphasizes the very interesting results we obtained and we conclude in section 7.

*e-mail: alexandra.bac@esil.univmed.fr
†e-mail: van.tran-nam@esil.univmed.fr
‡e-mail: marc.daniel@esil.univmed.fr

2 Related works

In the last ten years, the average size of the geometric models handled has drastically increased, therefore, the issue of mesh simplification has become more and more important. Many studies deal with this problem; they can be classified into two families: vertices clustering or iterative edge contraction.

Vertex clustering approaches are based on the following idea: the initial mesh is split into small patches (called cells); all the vertices of a cell are then replaced by a unique representative vertex. One of the simplest method has been proposed by Rossignac and Borrel (see [Rossignac and Borrel 1993]). The mesh is segmented by subdivision of its bounding box with a regular cubic grid; the representative vertex of each cell is taken to be the barycenter of the vertices it contains. The main advantage of this method is that it is extremely fast, but the quality of its results is however insufficient. Such insufficiency mostly results from the choice of the representative vertex (barycenter) which hardly takes into account the geometry of the surface and cloud of points.

Many other algorithms improve and optimize this choice. Let us mention the approach of Peter Lindstrom (see [Lindstrom 2000]) in which the representative vertex is computed by minimization of a quadratic form, namely, the quadratic error metric (see [Garland 1999]). This metric gives for each point x in $\mathbb{R}^3$, the sum of the squared distances between x and the faces of the cell. It is closely related to the 2-norm of the principal curvatures and this heuristic gives, in practice, results of better quality than previous methods.

All these methods, based on vertex clustering by means of a uniform grid, are very quick. However using such a uniform grid can produce approximation errors (whatever the quality of the choice of the representative vertex). Indeed, any detail of the original model smaller than the size of the grid is lost by the simplification process. Therefore, it is tempting to introduce the local curvature of the surface in this process: in order to improve the quality of the simplified surface, more details should be kept in the strongly bent areas. The R-simp algorithm by Brodsky and Watson [Brodsky and Watson 2000] groups vertices by means of an adaptive process. Starting from a single cell containing the initial mesh, cutting planes are iteratively inserted by means of a principal component analysis of the normals of the cell. This analysis provides approximate minimal and maximal curvature directions for the cell. Subdivisions are performed until the desired resolution is reached; the representative vertex of each cell is then computed by minimization of its quadratic error metric (same as in [Lindstrom 2000]).

Besides the R-simp algorithm, Shaffer and Garland ([Garland and Shaffer 2002]) presented their own mesh simplification method consisting of two steps. First, the initial mesh is segmented using a uniform grid; simultaneously, all the quadratic error metrics of the cells are computed and then used to compute the representative vertices. Second, an edge collapse step is applied (based on quadratic error metric) to improve again simplification while preserving the quality of the resulting mesh.

Let us conclude this section by some elements concerning simplification algorithms based on iterative edge collapse. Most of these algorithms use a greedy approach to chose their next target edge. A cost function is attached to each edge and at each step of the algorithm, the lower cost edge is chosen and collapsed. The difference

between the various algorithms (in terms of quality and computational efficiency) lies in the choice of this cost function as well as in that of the vertex towards which the edge is collapsed. Let us mention Garland and Heckbert ([Garland and Heckbert 1997]) where both the cost and the target vertex are obtained with quadratic error metrics, and more recently Borouchaki and Frey ([Borouchaki and Frey 2005]) who use an estimate of the Hausdorff distance between the original and the simplified mesh.

Other approaches are, of course, possible: Hoppe algorithm ([Hoppe et al. 1993]) is based on the minimization of an energy function (the number of vertices, their position and topology are modified in order to decrease the energy of the mesh). This algorithm leads to good visual results (similar to those obtained with quadratic error metrics) but it proves both time and space consuming.

The "memory less" algorithm of Lindstorm and Turk ([Lindstrom and Turk 1998]) minimizes geometric deviations between the original and simplified meshes (such as volume, area); this minimization is expressed by linear constraints (their resolution is far lighter than Hoppe algorithm).

3 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. This scheme is summarized in figure 1 (where N denotes the number of vertices of the mesh and B the expected number of vertices).

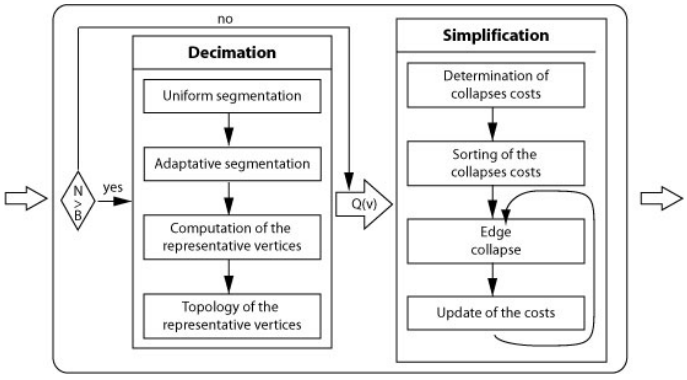

Figure 1: *General view of our simplification algorithm*

3.1 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 4.2).

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. [Meyer et al. 2002], see section 4.1). 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 4.2.3).

3.2 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 5).

4 Vertex clustering : adaptive segmentation

4.1 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 [Cazals and Pouget 2005], [Meyer et al. 2002], [Taubin 1995]). We chose to use the definition by M. Meyer and al. ([Meyer et al. 2002]) as it constitutes a good trade-off between quality and complexity (convergence results have been formally obtained by G. Xu in [Xu 2006]).

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 presents absolute curvature fields for both a geological surface and the well known rocker arm model.

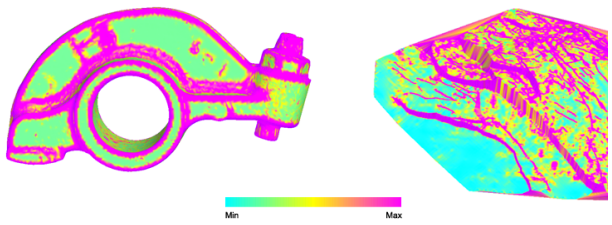

Figure 2: *Discrete absolute curvature fields: left, a rocker arm - right, a geological surface*

4.2 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.

4.2.1 Initialization

After loading the mesh, the initialisation 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).

4.2.2 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 [Garland and Shaffer 2002]), we use a principal component analysis of the normals of the cell (see [Jolliffe 1986]).

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 3). However, contrarily to figure 3, 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.

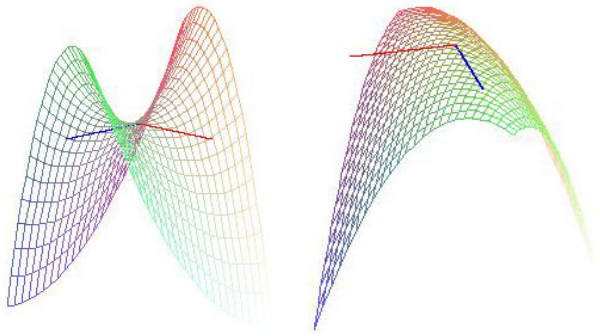

Figure 3: *Principal curvatures on smooth surfaces: (in red, direction of maximal curvature - in blue, direction of minimal curvature)*

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 [Vertterling et al. 2003]) . 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 4).

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

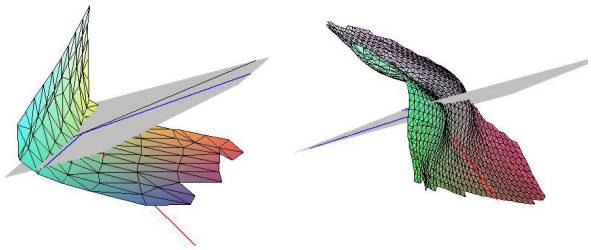

Figure 4: *Splitting planes for small cells (top) and a 300 vertices cell*

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' neighbour 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 5 presents both the uniform cells and those obtained after the adaptive subdivision process.

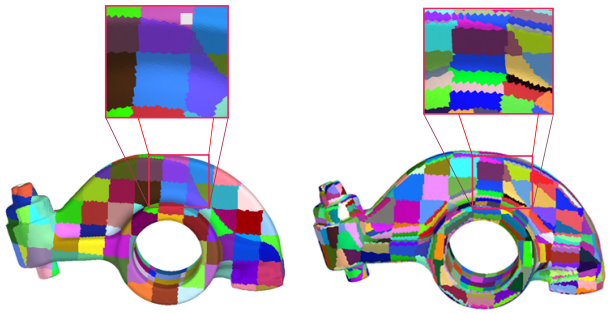

Figure 5: *Results of the adaptive subdivision process: left, uniform clustering - right, adaptive clustering*

4.2.3 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 [Brodsky and Watson 2000], [Lindstrom 2000] and [Shaffer and Garland 2001]. 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 6 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].

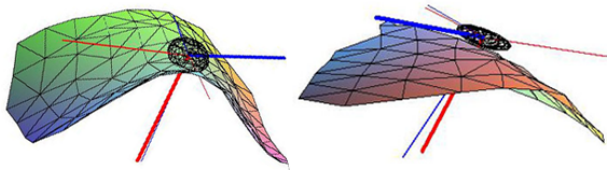

Figure 6: *Quadratic error metric for different cells - top: a saddle cell - bottom: a convex cell*

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:

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

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.

[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.

Let us point out that this post-processing (also used by [Brodsky and Watson 2000], [Lindstrom 2000] and [Shaffer and Garland 2001]) does not guarantee the manifoldness of the result (only that generally, it is manifold). The following example (figure 7) 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 non-manifold. Flipping edge (e, i) solves the problem (see right figure).

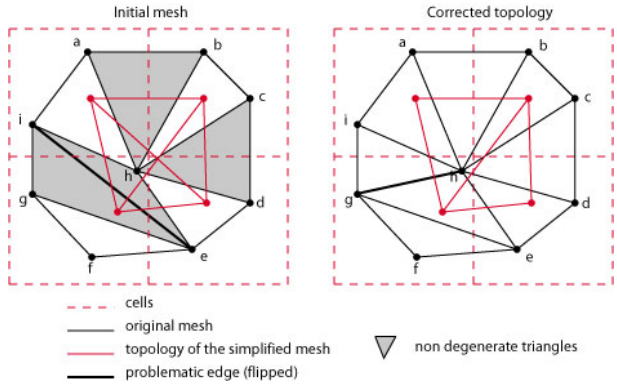

Figure 7: *Heuristic for the well known topological problem (non-manifoldness): left, the original mesh - right, the corrected mesh (an edge has been flipped) which gives rise to a manifold simplified mesh*

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.

5 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. ([Garland 1999]) with the quadratic error metrics previously computed.

Contracting a pair of vertices $(v_1, v_2) \rightarrow \bar{v}$ 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 neighbours 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 $(v_1, v_2) \rightarrow \bar{v}$ 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-neighbour ring of $\bar{v}$*

6 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 mustn't 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 8 and 9, 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 these characteristic lines are of particular interest for the geological interpretation of surfaces.

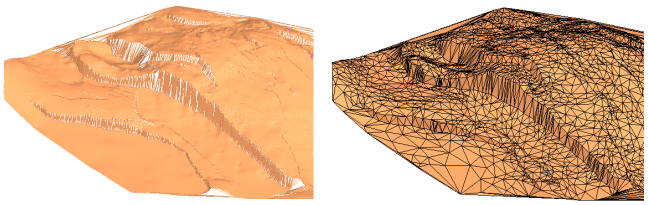

Figure 9: *A geological surface simplified with our method: initial model, 112k vertices (left) - simplified model, 3k vertices - size of the uniform grid: 151x188x27, size of the intermediate mesh: 56136 vertices*

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 ([Garland and Shaffer 2002]). The tests have been performed with two models: the "lucky lady" model (500k vertices) and the "dragon" model (437k vertices). Table 1 presents the numerical results obtained for this comparison. Figure 11 and 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.

In order to estimate the quality of our simplified meshes, we have compared them with surfaces obtained by the pure iterative edge collapse algorithm ([Garland 1999]). Figure 12 presents running times and error maps for both of these algorithms.

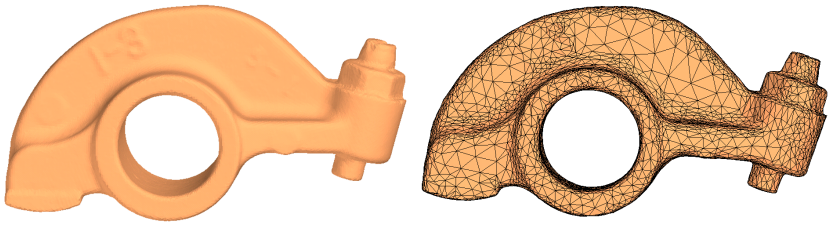

Figure 8: *The rocker arm model simplified by our method: initial model, 40k vertices (left) - simplified model, 5k vertices,* $D_{max} = 0.00029$*,* $D_{avg} = 0.0000345$ *(right) - size of the uniform grid: 41x24x78, size of the intermediate mesh: 20088 vertices*

Therefore, the quality of our results is similar to [Garland 1999] whereas our running time is three times lower.

7 Conclusion

The purpose of our algorithm, was to propose an alternative to vertex clustering simplification methods and to iterative edge collapse methods, by a compromise between both approaches. Regarding the results presented in section 6, 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 [Garland and Shaffer 2002]. Moreover, the heuristic we apply in order to avoid the well known topological problems resulting from simplification based on vertex clustering proves quite efficient.

In the geological field (in which our questions originated), this hybrid method allowed us to solve our initial problem, namely, to simplify very voluminous data while preserving strongly bent areas and curvatures.

8 Acknowledgement

The authors would like to thank to French Institute of Petrol for supporting this research and Jean-Phillipe Pernot, Phillipe Veron and the Digital Michelangelo Project for providing the models shown in this paper.

References

ASPERT, N., SANTA-CRUZ, D., AND EBRAHIMI, T. 2002. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, vol. I, 705–708. `http://mesh.epfl.ch`.

BAC, A., TRAN, N.-V., DANIEL, M., AND RAINAUD, J.-F. 2005. Traitement de surfaces géologiques pour la construction de modèles 3d. In *journées du GTMG*, 22–23.

BOROUCHAKI, H., AND FREY, P. 2005. Simplification of surface mesh using hausdorff enveloppe. *Comput. Methods Appli. Mech. Engrg. 194*, 4864–4884.

BRODSKY, D., AND WATSON, B. 2000. *Model Simplification In Reverse, Vector Quantization*. PhD thesis, University of Alberta.

CAZALS, F., AND POUGET, M. 2005. Estimating differential quantities using polynomial fitting of osculating jets. *Comput. Aided Geom. Des. 22*, 2, 121–146.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. In *Computer Graphics*, vol. 31, 209–216.

GARLAND, M., AND SHAFFER, E. 2002. A multiphase approach to efficient surface simplification. In *VIS '02: Proceedings of the conference on Visualization '02*, IEEE Computer Society, Washington, DC, USA, 117–124.

GARLAND, M. 1999. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. In *Computer Graphics*, vol. 27, 19–26.

JOLLIFFE, I. 1986. *Principal component analysis*. Springer Verlag.

LINDSTROM, P., AND TURK, G. 1998. Fast and memory efficient polygonal simplification. In *VIS '98: Proceedings of the conference on Visualization '98*, IEEE Computer Society Press, Los Alamitos, CA, USA, 279–286.

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

MEYER, M., DESBRUN, M., SCHROEDER, P., AND BARR, A. 2002. Discrete differentialgeometry operators for triangulated 2-manifolds. *VisMath*.

ROSSIGNAC, J., AND BORREL, P. 1993. Multi-resolution 3d approximation for rendering complexe scences. *Geometric Modeling In Computer Graphics*, 455–465.

SHAFFER, E., AND GARLAND, M. 2001. Efficient adaptative simplification of massive meshes. *In Proceedings of IEEE Visualization 2001* (October), 127–134.

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

VERTTERLING, W. T., TEUKOLSKY, S. A., PRESS, W. H., AND FLANNERY, B. P. 2003. *Numerical Recipe in C/C++, The Art of Scientific Computing*.

XU, G. 2006. Convergence analysis of a discretization scheme for gaussian curvature over triangular surfaces. *Comput. Aided Geom. Des. 23*, 2, 193–207.

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	$D_{avg} = 0.17E-3$ $D_{max} = 4.75E-3$	
Fig. 10			(2)	62x84x74	20975	31507	84	14	$D_{avg} = 0.17E-3$ $D_{max} = 2.37E-3$	$G_{avg} = 2,13\%$ $G_{max} = 50,12\%$
Venus	134k	3k	(1)	62x84x74	20975	20975	190	12	$D_{avg} = 0.39E-3$ $D_{max} = 4.94E-3$	
			(2)	52x70x62	15072	20975	114	16	$D_{avg} = 0.38E-3$ $D_{max} = 2.80E-3$	$G_{avg} = 2,92\%$ $G_{max} = 43,46\%$
Lucky	500k	45k	(1)	182x311x105	119713	119713	5457	83	$D_{avg} = 0,21$ $D_{max} = 5,35$	
			(2)	145x249x84	79190	119700	1587	86	$D_{avg} = 0,19$ $D_{max} = 2,04$	$G_{avg} = 8,24\%$ $G_{max} = 61,93\%$
Lucky	500k	100k	(1)	242x414x139	197251	197251	4579	82	$D_{avg} = 0,12$ $D_{max} = 5,33$	
Fig. 11			(2)	182x311x105	119713	194725	866	96	$D_{avg} = 0,11$ $D_{max} = 1,16$	$G_{avg} = 7,20\%$ $G_{max} = 78,28\%$

(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[Aspert et al. 2002] for more details)
- *Non-manifold edges* is the number of non-manifold edges resulting from vertex clustering (our heuristic aims at avoiding them)

Table 1: *Numerical comparison between our algorithm and Garland and al. 2002*

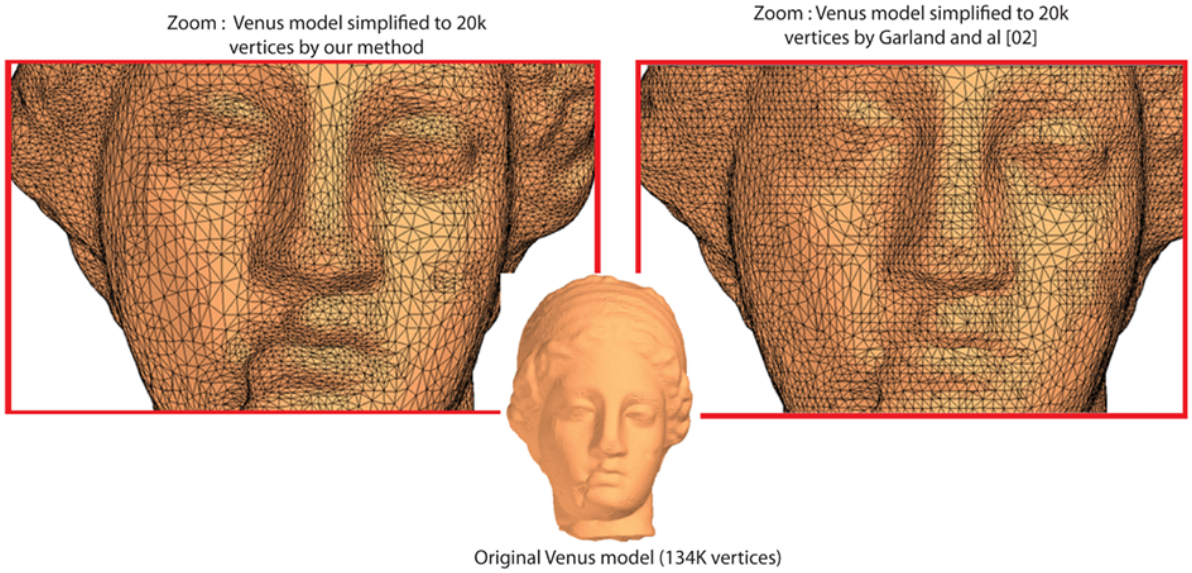

Figure 10: *Comparison of our method and Garland and al. 2002 - the "venus" model (134k vertices) - simplified model: 20K vertices*

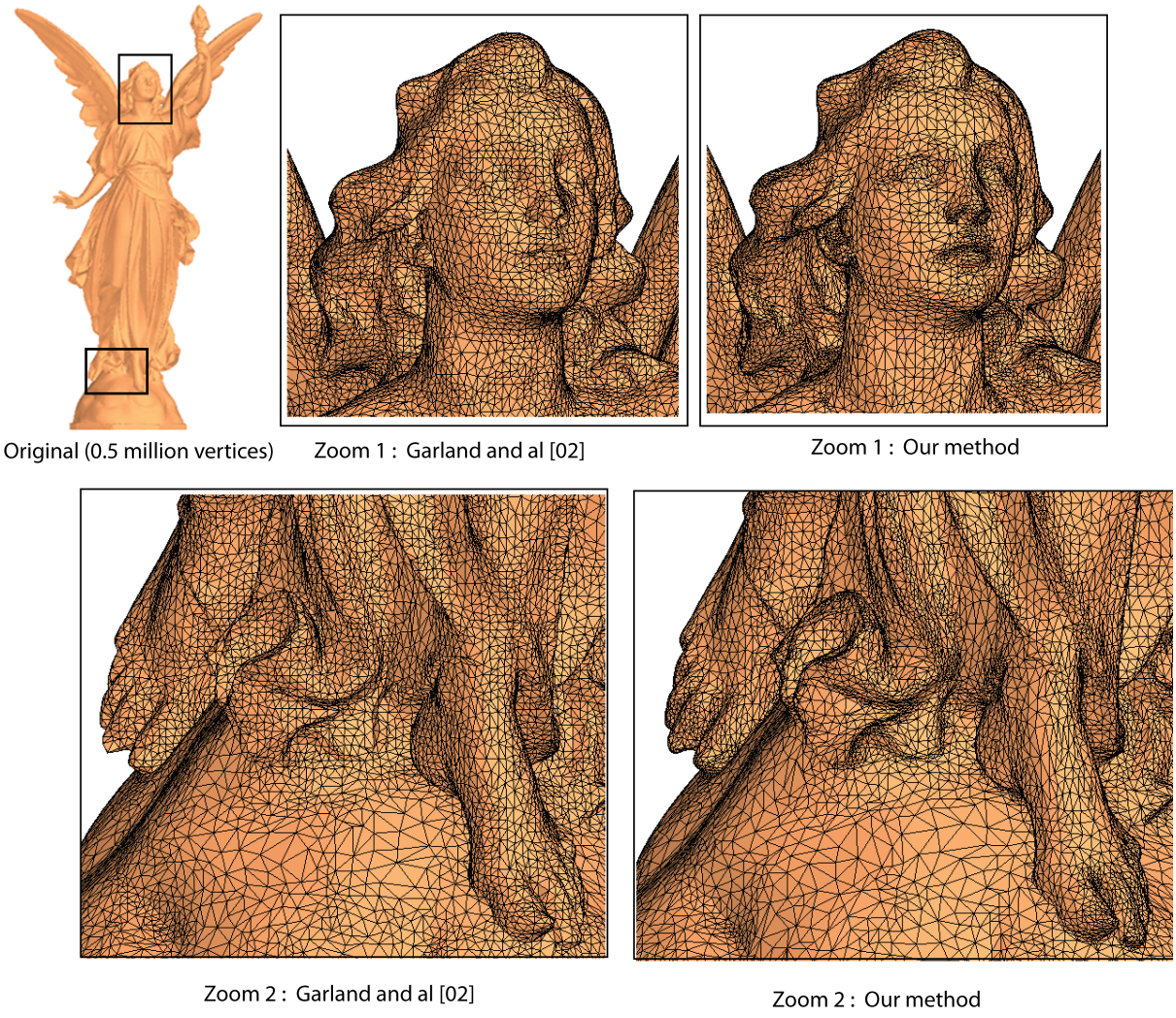

Figure 11: *Comparison of our method and Garland and al. 2002 - the "lucky lady" model (500k vertices) - simplified model: 100K vertices*

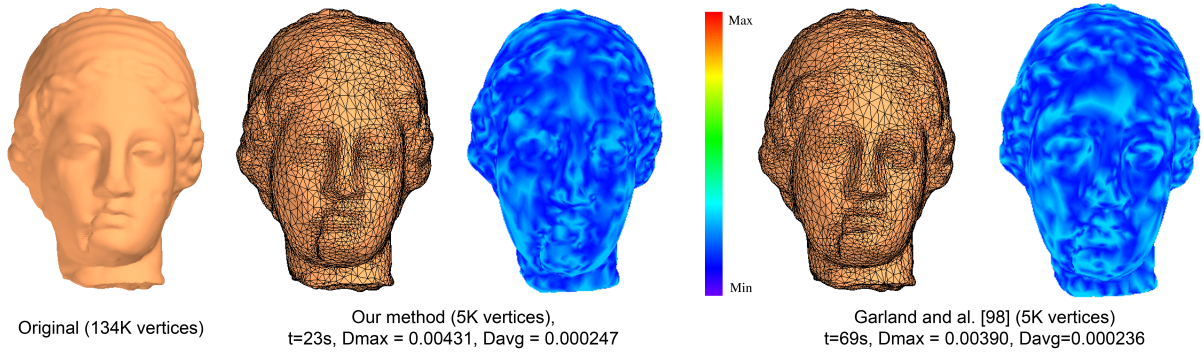

Figure 12: *Comparison of the map of errors for our method and Garland and al. 1999 - the "venus" model (134k vertices)*