Reconstruction Algorithms as a Suitable Basis for Mesh Connectivity Compression

Raphaëlle Chaine, Pierre-Marie Gandoin and Céline Roudet

***Abstract*— During a highly productive period running from 1995 to about 2002, the research in lossless compression of surface meshes mainly consisted in a hard battle for the best bitrates. But for a few years, compression rates seem stabilized around** 1.5 **bit per vertex for the connectivity coding of usual triangular meshes, and more and more work is dedicated to remeshing, lossy compression, or gigantic mesh compression, where memory access and CPU optimizations are the new priority. However, the size of 3D models keeps growing, and many application fields keep requiring lossless compression. In this paper, we present a new contribution for single-rate lossless connectivity compression, which first brings improvement over current state of the art bitrates, and secondly, does not constraint the coding of the vertex positions, offering therefore a good complementarity with the best performing geometric compression methods. The initial observation having motivated this work is that very often, most of the connectivity part of a mesh can be automatically deduced from its geometric part using reconstruction algorithms. This has already been used within the limited framework of projectable objects (essentially terrain models and GIS), but finds here its first generalization to arbitrary triangular meshes, without any limitation regarding the topological genus, the number of connected components, the manifoldness or the regularity. This can be obtained by constraining and guiding a Delaunay-based reconstruction algorithm so that it outputs the initial mesh to be coded. The resulting rates seem extremely competitive when the meshes are fully included in Delaunay, and are still good compared to the state of the art in the case of scanned models.**

***Note to Practitioners*— A 3D triangle mesh is composed of a geometric part (the vertex coordinates) and a connectivity part (the description of the triangles). In this article, we show how to reencode such surface meshes in order to obtain near zero connectivity cost for some class of surface meshes (and very good rates in the general case), while guaranteeing in the same time state-of-the-art geometry encoding cost. This method can be useful in all application areas where the mesh size is a bottleneck (typically network or storage applications). The best results are obtained for meshes made from 3D scans (in contrast to CAD meshes). The main current limitations of the method are the computing times (about 1 second per 1000 points part of the mesh, for compression which can be done off-line) and the memory footprint.**

***Index Terms*— Mesh, Compression, Reconstruction, Lossless, Connectivity**

I. Introduction

For several years, meshes have played a leading role in computer graphics, and their ability to model the world throws them in the heart of advanced applications in all the fields of science, arts or leisure. If some of these applications can tolerate a limited loss of information (provided that this loss is well controlled and does not damage a certain visual realism), the others, for practical or even legal reasons, impose to work continuously with exact copies of original objects. In this case, the only way of optimizing the storage and the transmission of 3D information is to have recourse to lossless compression methods.

A mesh is defined by a set of points (we speak of the geometry of the mesh), and by a combinatorial structure describing the relations between these points, using geometric objects of higher dimensions: edges, facets (we speak of the connectivity or the topology of the mesh). To this fundamental information are sometimes added attributes allowing to improve the rendering: normals, colors or textures. If we put aside these attributes (they relate only to a subset of the meshes met in practice), all the difficulty for compressing 3D objects is to efficiently process both the geometry and the connectivity of a wide class of meshes. However, most of the methods proposed so far stress on one of these two aspects (usually the connectivity), generally to the detriment (or at least, not to the advantage) of the other one, which is constrained by a description order rarely optimal in terms of compression. In this article, we propose a new single-rate connectivity coder which is not only general and efficient, but also does not impose any constraint on the coding of the geometry. In particular, it is totally compatible with the position coders that currently propose the best compression rates.

We took it as given that very often, the main part of a mesh connectivity can be deduced from its vertex set using reconstruction methods. This remark has already been made in the past, but the problem was to find an effective way of describing the difference between the initial mesh and its reconstructed version. Indeed, within the framework of lossless compression, it is indispensable to be able to correct these errors in order to obtain a decoded mesh perfectly identical to the original mesh. But rather than describe the difference to the initial object at the end of the reconstruction phase, we suggest adapting an existing algorithm so that it can accept occasional codes modifying its default behavior at the moments when it would commit an ”error” of reconstruction with regard to the initial mesh.

Having placed our work in the historical context of 3D compression and 3D reconstruction (Sec. II), we will expose the general principle of the method (Sec. III), first within the restricted framework of Delaunay meshes, then generalized to arbitrary triangular meshes (independent of the genus,

Université de Lyon, Lyon, F-69003, France ; université Lyon 1, Villeurbanne, F-69622, France, université Lyon 2, Bron, F-69676, France, CNRS, UMR 5205, LIRIS, Villeurbanne, F-69622, France
FirstName.LastName@liris.cnrs.fr

the number of connected components, the regularity and the manifoldness). Then we will detail the coding techniques and optimization steps leading to better compression rates (Sec. IV), before presenting some comparative results and comments on the method performances (Sec. V).

II. Context and Previous Works

A. Mesh Compression

As mentioned above, a mesh is composed of both a geometric part (the vertex positions), and a topological part (the vertex connectivity). Now, by looking at the whole scientific production in mesh compression since 1995 until now, we notice clearly that the connectivity coding has motivated most of the proposed methods. The usual scheme consists in describing the combinatorial structure by enumerating its vertices in a precise order, designed to minimize the size of the code: each vertex is coupled with a variable size symbol defining the way it is connected to the rest of the mesh. Consequently, the geometric coder has to work with this predefined order of the vertices, but it can exploit the connectivity to predict their location. The position of the currently transmitted vertex is estimated from its neighbors, and the prediction error is the sole information to be coded. But the order imposed by the connectivity coding is not necessarily favorable to this prediction, as shown by the results obtained on usual meshes by classical algorithms: the best of them reduce the connectivity information to less than 1 or 2 bits per vertex, while for the geometric part, the rates rarely go down under 90% of the initial size (except for very low quantizations). Among the works that follow this principle, we focus here in single-rate methods, which code and decode the mesh in one pass and do not allow progressive visualization [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. To understand how these methods compare, the reader can also refer to the following surveys [20], [21], [22]. It is worth to mention here that the relative stability in compression rates observed these days can be explained by the new interest of the community for gigantic meshes: in this framework, the challenge is to improve the efficiency of the compression in terms of CPU and memory requirements rather than in terms of bitrates [23], [24], [25].

B. Prioritizing the Geometric Coding

After this first wave of works, in the knowledge that the geometry of a mesh weighs generally much more than its connectivity, the researchers began to propose methods giving the priority to geometric compression. Some of them deviate from the lossless framework by proposing algorithms that often impose a complete remeshing of the object before applying spectral decomposition tools like wavelets, subdivision schemes, or classic geometric methods of compression [26], [27], [28], [29], [30], [31], [32], [33], [34], [35].

On the other hand, the works introduced by Gandoin and Devillers and improved by Peng and Kuo [36], [37], [38] describe a progressive and lossless compression method, centered on the geometry, which interests us particularly within the framework of this article. Indeed, this method was originally designed to code an unstructured point cloud, with the underlying idea that in many cases, the connectivity information could be deduced from the geometry thanks to reconstruction algorithms. So, the first version of this method [36] proposed a multiresolution compression algorithm for an unstructured point cloud, guaranteeing a theoretical minimal gain of $n(log_2 n - 2.4)$ (where n denotes the number of points), and very competitive practical performances, even compared to the best single-rate algorithms. The method was then extended to deal with the connectivity compression, while remaining centered on the geometry [37]. Indeed, the kd-tree decomposition scheme involved in the heart of the geometric coder, is enriched with a connectivity coder that uses some of the classical mesh simplification operations introduced by Hoppe *et al.* [39], [40]). Eventually, the method has recently been resumed by Peng and Kuo [38] who improve its performances by imposing a priority on the cell subdivision order in the octree, and by proposing a more effective prediction model for the point distribution in the sub-cells generated by a subdivision.

C. Compression and Reconstruction

The idea to entrust a reconstruction algorithm with the task of computing the connectivity of a mesh from its vertices has already been used in the context of mesh compression. Within the framework of terrain models, Kim *et al.* [41] suggest transmitting only a fraction of the edges composing the object, and using constrained Delaunay triangulation — in 2D, since the terrain models are projectable — to find the whole connectivity. Devillers and Gandoin [36] also mention the compression of GIS as an application of their geometric coder, and develop an edge coder well suited for this context, which results in compression rates as low as 0.2 bits per vertex for the complete connectivity. Besides, Devillers *et al.* show that the minimal set to be transmitted to guarantee the exact reconstruction of the initial model by constrained Delaunay triangulation is constituted by the edges that are non locally Delaunay [42].

Unfortunately, the generalization from 2.5D to 3D meshes is not straightforward: since the mesh is not projectable any more, the use of the constrained Delaunay triangulation is impossible. Of course, it is possible to recourse to local projections of the mesh [43], but we are left with the problem (and the overcost) of joining the different parts together. Nevertheless, the idea to use a reconstruction method driven by a partial description of the connectivity remains extremely promising in terms of compression results. Provided that it could be possible to find a method both powerful and capable of taking advantage of partial connectivity data to guarantee an exact reconstruction whatever the initial model may be.

A first attempt to use a reconstruction algorithm to encode connectivity information has been proposed by Lewiner *et al.*

[44]. The geometry is coded independently, through a kd-tree based algorithm derived from [37] and [45] and the connectivity is coded through an advancing front triangulation inspired of the ball-pivoting strategy [46]. This algorithm requires a code for each edge of an active border that is initialized to a triangle. If the geometry of the mesh meets good sampling conditions, the entropy of each code will be extremely low. In Sec. V, we compare this method to ours regarding the compression rates.

D. *Reconstruction by Convection*

The problem of reconstructing a surface from a set of points has received considerable attention during the last decade [47]. Interesting and outstanding algorithms have been issued both in computer graphics and computational geometry, but we have decided to focus on the algorithms of the second category, since their combinatorial concerns are more suitable for lossless compression purposes. Most of computational geometry algorithms exploit the geometric properties of structures such as the Delaunay triangulation, the Voronoi diagram or the power diagram of the input point set, assuming auspicious properties on the way they were sampled on the surface (ε-sample [48]). A consistent set of facets can then be extracted from the geometric structure, sometimes using global heuristics or local analysis to solve ambiguities. The existing algorithms are numerous and an attempt to classify and distinguish them has recently been proposed in the state of the art by Cazals and Giesen [49]. Most of these algorithms produce triangular meshes, which makes them good candidates to use for compression purposes.

The convection algorithm we use in our method is based on a similar notion of flow as it was developed in the Wrap algorithm by Edelsbrunner [50], and the flow complex algorithm by Giesen and John [51]. Indeed, this reconstruction algorithm has been inspired from a surface convection process described by Zhao *et al.* [52]. They use it to initialize their surface before running an energy minimization process in the level set framework. Given an evolving surface S enclosing an input point set P, the convection process makes each point of S move inwards, along the direction of its normal, towards its closest point in P. However, the numerical scheme they propose can be translated in the discrete setting of a 3D Delaunay triangulation, to make it depend on the geometry of the input data set only, and not on the precision of some grid around the surface. A demonstration of this result is presented by Chaine [53] together with a subsequent convection algorithm. A Delaunay triangulation of P is a partition of space into tetrahedra so that the ball circumscribed to each tetrahedron does not contain any point of P. The convection process is run directly in the 3D Delaunay triangulation of P, with an evolving surface S composed of oriented Delaunay facets. S is initialized to the convex hull of P. An oriented facet of S that does not meet the oriented Gabriel property — *i.e.* the inwards half of its enclosing sphere is not empty — is attracted towards the 3 other facets of the incident Delaunay inner tetrahedra (see Fig. 1, for an illustration in 2D where the evolving surface is replaced by an evolving curve, and the Delaunay tetrahedra are replaced by Delaunay triangles). During the convection process, thin parts can appear (see Fig. 1, c and d), on which a surface embedded 2D version of the convection is run. A deeper explanation of this algorithm will be presented in Sec. III while revisiting it for compression purposes.

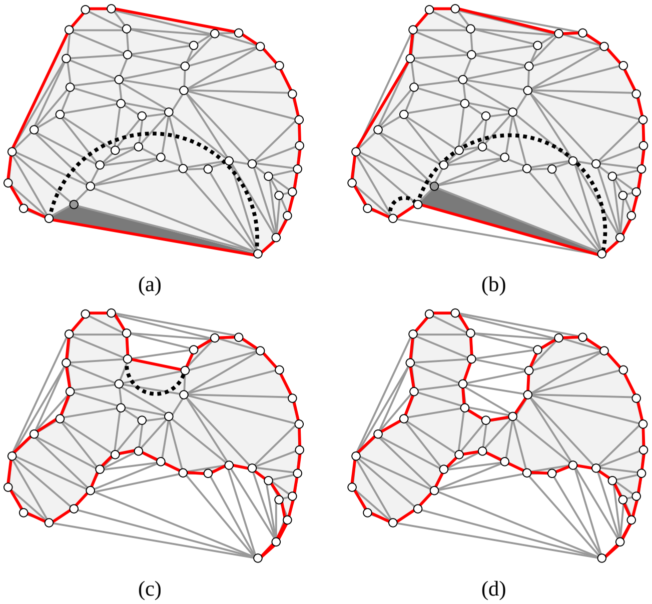


Fig. 1. Geometric convection on a 2D point set : (a) the evolving curve C is initialized to the convex hull, (b) C locally evolves at the level of an edge if the half-circle associated to this edge is not empty, (c) result of the initial convection process, (d) the convection process is locally enhanced to hollow a pocket out.

An interesting property of the convection algorithm is that it is driven locally, in the Delaunay triangulation of the points, without involving a global heuristic. The topology of the evolving surface may change during the convection process, so that it can handle surfaces with boundaries and surfaces of high genus. A drawback of the convection process is that it can locally be stuck in presence of pockets [54] hiding facets, but a simple and local point density analysis permits to hollow them out [53].

III. Principle of the Compression Algorithm

A. *Introduction of the Convection Abilities*

The benefit of using a 3D reconstruction method for compression purposes is double: first, this allows to obtain very low costs for the coding of the connectivity — ideally, a null cost if the reconstruction algorithm is able to find the exact structure of the original mesh by itself —, and secondly, unlike the previous methods of topological compression, no constraint is imposed on the order of the vertices to the geometric coder, which constitutes an important efficiency token.

The main difficulty consists in being able to help the reconstruction algorithm at a reasonable cost: indeed, it is highly improbable to design an algorithm capable of finding the complete connectivity of a mesh from its vertex set. It is thus necessary to be able to alter the course of the reconstruction process, by occasionally changing the default behavior of the algorithm to drive it in a sure way to a result known in advance: the structure of the initial mesh.

Among the plethora of available methods, the reconstruction by convection is distinguishable from others by two important assets: its qualities in terms of reconstruction — practical accuracy and faithfulness to the original model, handling of complex topologies, computation times —, and above all, its ability to be effectively driven by means of simple instructions. The first asset guarantees that the algorithm will not need to be guided too often (small number of codes), the second guarantees that this can be done at lower cost (compactness of the codes).

In return, as many algorithms in computational geometry, the convection algorithm is based on a Delaunay triangulation, and it may be difficult to force it out of this structure. This imposes a condition over the initial mesh: all its facets have to belong to the 3D Delaunay triangulation of its vertex set. It is quite a strong property that is not verified by all the 3D objects met in practice. We will see in second stage (Sec. III-D) how to break it.

Intuitively, the reconstruction by convection consists in embedding the vertex set in a block of marble that will be sculpted gradually, facet after facet, tetrahedron after tetrahedron, until it takes on the shape of the object. The algorithm begins by preparing the sculpting material: an enclosing block which is nothing but the convex hull of the point cloud, as well as the network of galleries through which it is going to dig until the final shape is reached. This network is composed of the tetrahedra of the 3D Delaunay triangulation, and every stage towards the object shape consists in examining a new facet of the current surface and deciding if it is necessary or not to open it and excavate the inside of its associated tetrahedron. When a facet is opened, it is removed from the current surface of the 3D object under construction, and replaced by the three other facets of the excavated tetrahedron.

As mentioned above, the criterion that decides on this opening is purely local: it consists of observing whether the Gabriel half-sphere associated to the current oriented facet contains or not the fourth vertex of the tetrahedron. If it is the case, the oriented facet is opened and replaced in the current surface by the 3 other facets of the associated tetrahedron. (If this one is already excavated, the surface locally vanishes through the current facet.) Otherwise, it is maintained on the surface.

Note that if all the facets of the initial object are in its 3D Delaunay triangulation, they are reachable by this sculpture process from the convex hull. The problem is to make sure that the algorithm will not dig a facet belonging to the initial mesh, or that, conversely, it will not be retained before having reached such a facet. Hence the need for additional codes allowing to modify the behavior of the convection algorithm, and to drive it towards the object to be coded. Even if the convection algorithm was designed to compute the structure of a sufficiently dense mesh with a good accuracy, it will most likely need occasional assistance to guarantee the perfect reconstruction of any mesh.

B. *Compliant Meshes*

Definition: A mesh is said to be *compliant* if all its facets are in the 3D Delaunay triangulation of its vertex set, and if it is a manifold (that is to say without borders nor thin parts).

To present our method in a progressive way, we focus in this section on compliant meshes only. Given such a mesh M, here is the general principle of the compression algorithm:

1 Build the 3D Delaunay triangulation D of the point set P,

2 Mark the facets of D belonging to the initial mesh M,

3 Initialize the current surface S with the convex hull of P (included in D),

4 Launch the convection process on S: every oriented facet f in S is examined in turn, and depending on whether its Gabriel half-sphere is empty or not, f is, by default, maintained on S or replaced by its 3 neighbors in D. To modify this default behavior of the convection reconstruction, it is necessary to locate the oriented facets of S for which the algorithm has to act differently. It is thus indispensable to define a completely deterministic traversal of the facets in S, so that the n^{th} oriented facet met during the compression would also be the n^{th} oriented facet met during the decompression. More generally, it is necessary to ensure the synchronization of the algorithms of compression and decompression so that the n^{th} action of the coder matches the n^{th} action of the decoder. Thanks to these reference marks, the behavior of the convection process can be safely altered in the following two circumstances:

   a) when the convection asks for the opening of a facet f that belongs to M, the coder forbids this opening, and codes the index of f (more exactly, the moment when f is met in the algorithm) to warn the decoder that at this precise moment of the reconstruction, it has to break the rules of the convection algorithm. We will call this a RDG event (Retain Despite the Geometry),

   b) at the end of this first step, it is possible that some oriented facets of S — those whose Gabriel half-spheres are empty and thus the convection did not decide to dig —, do not belong to M. Therefore, it is necessary to specify to the decompression algorithm that these oriented facets must be forced, against the convection rules: for each remaining oriented facet of S not belonging to M, the coder transmits its index (*i.e.* the moment when it is met) and locally relaunches the convection process by forcing its opening. We call this a ODG event (Open Despite the Geometry). Note that by thus delaying the enforced opening of facets in a second step, rather than opening them the first time they are met, some facets that were due to be forced can disappear by autointersection of S: in some cases, the convection process reaches and automatically opens the opposite side (or half-facet) of a facet previously encountered but not opened then. This permits to save some ODG codes.

Fig. 2 is a step by step illustration of the algorithm in 2D, in the case where the deterministic traversal of the facets is induced by a breadth-first traversal of the Delaunay triangulation, starting from the convex hull. To be more illustrative, time has been incremented at each step of the algorithm, but in practice, it is enough to increase it each time the convection is willing to open a facet or each time a retained facet is tested for confirmation.

A detailed description of step 4 can thus be given through a recursive function *Convection* applied to the current surface S. An other difference with the step by step example is also that the facets are yet visited through a depth first traversal of the Delaunay triangulation. For each oriented facet f, let f_{asso} denote the oriented facet associated to f, that is to say the oriented facet constructed on the same vertices as f, but with the opposite orientation. When f and f_{asso} are both in the surface S, that means they belong to a thin part of S. Besides, let f_1, f_2 and f_3 denote the 3 neighbors of f in the 3D Delaunay triangulation D, *i.e.* the 3 other facets of the tetrahedron that is removed when f is opened. Finally, we would like to draw the reader's attention to the fact that the reference marks transmitted to the decoder are not exactly absolute moments of events in the compression process, but rather intervals between two such moments (which explains the presence of the instructions "$time \leftarrow 0$" in the detailed algorithms). This well-known technique of differential coding allows to reduce the size of the transmitted codes.

```
Function $Convection(S : Surface)$:
  while $S \neq \emptyset$ do
    $f \leftarrow$ pop first oriented facet in $S$
    if Gabriel half-sphere of $f$ is empty then
      push $f$ at the end of $S_{temp}$
    else
      if $f \in M$ then {RDG event}
        output $time$
        $time \leftarrow 0$
        push $f$ at the end of $S_{final}$
      else
        if $f$ such that $f_{asso} \in S$ (resp. $S_{temp}$) then
          remove $f_{asso}$ from $S$ (resp. $S_{temp}$)
        else
          push $\{f_1, f_2, f_3\}$ at the end of $S$
        end if
        $time \leftarrow time + 1$
      end if
    end if
  end while
```

With this recursive definition of the *Convection* function, the step 4 of our algorithm amounts to the following:

```
Main function:
  $S \leftarrow$ convex hull of $P$
  $S_{border} \leftarrow \emptyset$ {set of facets creating a thin part}
  $S_{final} \leftarrow \emptyset$ {set of facets that are in $M$}
  $S_{temp} \leftarrow \emptyset$ {set of facets candidates to be in $M$}
  $time \leftarrow 0$
  $Convection(S)$
  while $S_{temp} \neq \emptyset$ do
    $f \leftarrow$ pop first oriented facet in $S_{temp}$
    if $f$ such that $f_{asso} \in S_{temp}$ then
      remove $f_{asso}$ from $S_{temp}$
      push $\{f, f_{asso}\}$ at the end of $S_{border}$
    else
      if $f \in M$ then
        push $f$ at the end of $S_{final}$
        $time \leftarrow time + 1$
      else {ODG event}
        output $time$
        $time \leftarrow 0$
        $S \leftarrow \{f_1, f_2, f_3\}$
        $Convection(S)$
      end if
    end if
  end while
```

At the end of the compression algorithm, the convection has been driven towards the initial mesh M, and the correcting data have been stored for the decompression algorithm. However, a last stage remains that consists in cleaning thin parts possibly generated by the convection. Indeed, since we first assumed that the initial mesh was a manifold, it suffices to delete all these thin parts, that is to say each facet of S whose associated facet is also on S. In the algorithm described above, the thin parts exactly match the set S_{border}.

C. Non Manifold Meshes

In this section, we are going to extend the previous algorithm to non manifold, thin parts meshes. It suffices to modify the final stage regarding the treatment of thin parts created by the algorithm. This stage is replaced by a new convection process, but this time in its 2D version, and on thin parts only. In this framework, the convection updates a curve C constituted by the edges composing the boundaries of the thin parts. The oriented edges of C are examined in turn: an oriented edge whose Gabriel half-circle is empty will be kept on C, whereas an oriented edge in the opposite case will be removed and replaced in C by the two other oriented edges of its incident triangle (e_1 and e_2 in the algorithm detailed below).

The general principle remains the same as in the 3D version of the algorithm, and each edge that is opened against the convection rules (ODG event) launches a new 2D convection process. Note that there is no set C_{border} similar to the previous S_{border}. Here is a detailed version of the portion of the compression algorithm dedicated to the processing of the thin parts. It adds to the steps 1 to 4 described in the previous section:

```
5 Processing of thin parts
  Function $Convection\_2D(C : Curve)$:
    while $C \neq \emptyset$ do
      $e \leftarrow$ pop first oriented edge in $C$
      if Gabriel half-circle of $e$ is empty then
        push $e$ at the end of $C_{temp}$
      else
        if $e \in M$ then {RDG event}
          output $time$
          $time \leftarrow 0$
          push $e$ at the end of $C_{final}$
        else
          if $e$ such that $e_{asso} \in C$ (resp. $C_{temp}$) then
            remove $e_{asso}$ from $C$ (resp. $C_{temp}$)
          else
            push $\{e_1, e_2\}$ at the end of $C$
          end if
          $time \leftarrow time + 1$
        end if
      end if
    end while
  Main 2D function:
    $C \leftarrow$ boundaries of $S_{border}$
    $C_{final} \leftarrow \emptyset; C_{temp} \leftarrow \emptyset; time \leftarrow 0$
    $Convection\_2D(C)$
    while $C_{temp} \neq \emptyset$ do
      $e \leftarrow$ pop first oriented edge in $C_{temp}$
      if $e$ such that $e_{asso} \in C_{temp}$ then
        remove $e_{asso}$ from $C_{temp}$
      else
        if $e \in M$ then
          push $e$ at the end of $C_{final}$
          $time \leftarrow time + 1$
        else {ODG event}
          output $time$
          $time \leftarrow 0$
          $C \leftarrow \{e_1, e_2\}$
          $Convection\_2D(C)$
        end if
      end if
    end while
```

D. Non Delaunay Meshes

As previously described, the method can only be applied to meshes whose facets belong to the 3D Delaunay triangulation of their vertex set. Indeed, we saw that this structure constitutes the support of the convection algorithm, and that it is thus impossible for the current surface to reach a non Delaunay facet. Nevertheless, most of the meshes met in practice contain a fraction of non Delaunay facets (see the unfavorable case of Fig. 3). So, to make the method widely usable, it is necessary to manage the coding of such facets. A simple and efficient way to do this is to code explicitly all the non Delaunay facets of the initial mesh in the same time as its vertices, using the method of Gandoin and Devillers [37] (which we will note GD in the following), or any other method efficient at coding sparse facets. More precisely, non Delaunay facets constitute patches on the surface of the mesh. Before launching the convection process, the connectivity of these patches is transmitted to the GD coder. Then, the algorithm previously described is applied to the mesh minus the non Delaunay facets. Consequently, even if the initial mesh is a manifold, the convection process is going to perform on a mesh with boundaries. As shown in the previous section, our method can handle this case, but the presence of boundaries induces a significant increase in the number of driving codes: indeed, each facet of the mesh will be reached twice, first through a facet oriented outwards, then through the associated oriented facet lying on the internal side of the object surface. We propose several heuristics in order to limit this phenomenon. The intuitive idea is to block the convection surface temporarily, when it is about to cross a patch — a facet opening is delayed if the tetrahedron behind it intersects the mesh —, so as to favor the discovery of the initial mesh facets from outside (see Fig. 4). Thus, transmitting the total number of mesh facets to the decoder will allow to stop the convection process as soon as they are all discovered, which will drastically reduce the number of corrective codes.

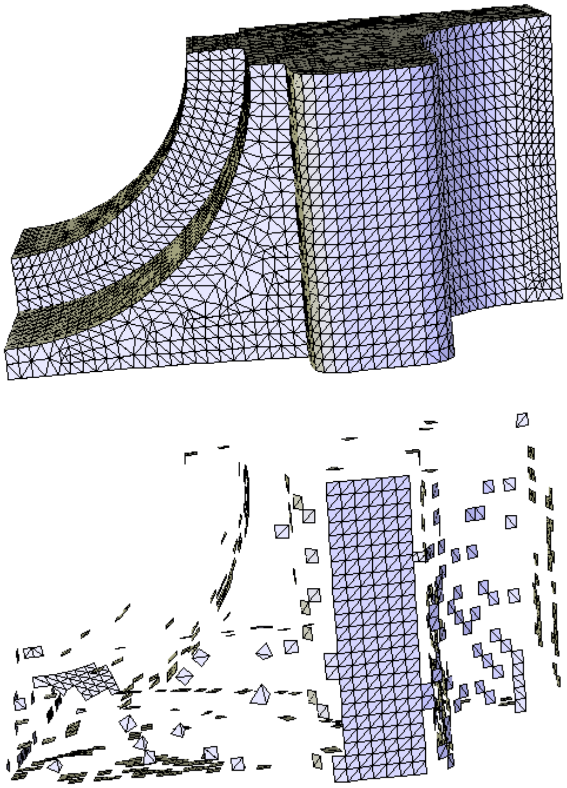


Fig. 3. Fandisk: complete (12946 facets), then showing the non Delaunay facets only (1104 facets representing 8.5% of the set)

In the decoding stage, the vertex set is first obtained from the GD decoder, as well as the non Delaunay patches connectivity. Then the mesh without the patches is reconstructed by the algorithm of driven convection. At last, the patches connectivity is merged into the reconstructed leaky mesh. We thus see that our method improves the results of the method of Gandoin and Devillers only for meshes whose facets are mainly Delaunay. In the limit case where all the mesh facets would be non Delaunay, we would come across the GD coder compression rates.

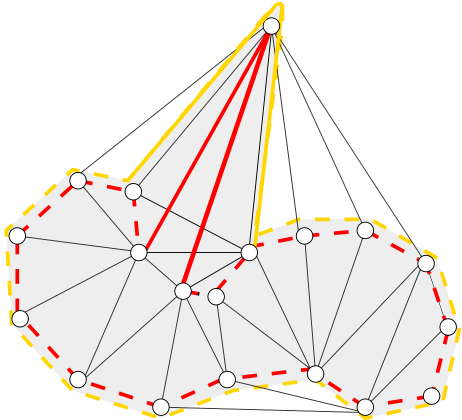


Fig. 4. 2D Example: the original curve to be coded is in red thick line, including non Delaunay patches (parts in solid line). The Delaunay triangulation is in black thin line. The current evolving convection curve, in orange thick line, is temporarily blocked (parts in solid line) to avoid intersecting the patches. The remaining hidden Delaunay edges of the original curve will thus be discovered later, when the convection process will be relaunched to intersect the patches.

IV. Coding and Optimization

We saw in Sec. III that the codes resulting from the compression algorithm are a sequence of positive or null integers, representing the number of facets met between two special interventions of the algorithm, that is to say between two facets for which the algorithm does not follow the rules of the reconstruction by convection (ODG and RDG events). To minimize the output size, we have chosen to transmit these numbers to an arithmetic coder, which is able to code a sequence of symbols in a nearly optimal way, given a probability model. Thus, if $p(s)$ denotes the probability of appearance of the symbol s, the arithmetic coder will encode s on $log_2(1/p(s))+\varepsilon$. In particular, this entropic coder is capable of coding a symbol on a fractional number of bits.

The main difficulty consists in defining a good probability model for the integers to be coded. The first solution consists in computing statistical data for each mesh and transmitting the probability table in the header of the compressed file. But ideally, to save the transmission of such a table, we would like to model the behavior of the sequence to be coded for a wide class of meshes, or alternatively, to design an adaptive model recomputing the probability of a symbol each time it occurs. Several kinds of model can be used, according to the size of the context from which the probability is estimated. For the order-0 model, each integer has an absolute probability for the whole coding sequence, independent from the context. For the order-1 model, the probability of an integer depends on the value of the previously transmitted integer, and so on. We have finally opted for an adaptive order-1 model specifically designed to take advantage of the particular structure of the integer sequence. For instance, this model handles long runs of null integer frequently occurring in the sequence.

As a matter of fact, these runs often correspond to large patches of facets that have to be forced (ODG events) because the Gabriel criterion is too restrictive compared to the local density of the mesh. Statistics illustrate that the convection process is rarely wrong when it decides to open a facet; on the contrary, at the end of the convection, a lot of facets have been retained but do not belong to the initial mesh. To encourage the convection algorithm to open more facets, we have relaxed the Gabriel criterion under some conditions. Intuitively, a facet will be opened when its size is ”big” with regard to the vertex density in its neighborhood [53]. By setting this ratio to a near optimal value, the improvements can be drastic, particularly in the case of poorly sampled meshes where the number of driving codes can be lowered by about 50%.

V. Experimental Results

The table I shows the results of the method applied to some usual meshes (the rates are in bits per vertex). The objects *fandisk*, *blob*, and *horse* are there essentially for comparison purposes, since they have been used for example by Touma and Gotsman [5]. The Stanford *bunny* is also a very classical mesh, widely used in 3D compression for about ten years. However, these models are not very representative of today meshes, whose number of vertices is much higher. The last meshes are more typical of what can be found nowadays. The *hand* (see Fig. 5) and *max_planck* (see Fig. 5) can be found on *http://shapes.aim-at-shape.net*, while the *triple_hecate* (see Fig. 5) comes from *Le Louvre C2RMF lab*.

TABLE I
Experimental results on usual meshes

model (number of vertices)	number of facets: total / non Del.	connectivity rate in bpv (Del. + non Del. facets)	computing time in seconds: compression / decompression
fandisk (6 475)	12 946 8.5 %	2.08 (1.22 + 0.86)	14 14
blob (8 036)	16 068 5.2 %	2.68 (1.98 + 0.70)	17 15
horse (19 851)	39 698 3.8 %	2.00 (1.48 + 0.52)	26 22
bunny (35 947)	71 890 0.0 %	0.08 (0.08 + 0.00)	18 14
triple_hecate (75 729)	151 462 0.0 %	0.04 (0.04 + 0.00)	46 37
ford (97 887)	188 024 57.0 %	8.43 (3.43 + 5.00)	40 29
max_planck (199 169)	398 043 1.7 %	0.90 (0.69 + 0.21)	201 157
ajax (273 383)	547 117 0.0 %	0.14 (0.14 + 0.00)	154 117
hand (327 290)	654 596 0.7 %	0.19 (0.11 + 0.08)	342 288
UNC powerplant (11 070 509)	12 748 510 48.9 %	> 2.16 (2.16 + -)	16316 15538

The first remark is that the rates obtained for the models *fandisk*, *blob*, *horse*, *ford*, *UNC powerplant* are not especially competitive with regard to the best current methods (for example, [18], whose algorithm yields 0.74 bpv for the *fandisk*, and 0.96 bpv for the *horse*). The main reason is that the vertex set of these meshes are clearly not ε-sample and therefore, not favorable to the convection algorithm (but note that on these meshes, we gain on the geometry coding by using the GD coder, as shown in [37]). On the contrary, the *hand* and *max_planck* are correctly sampled and give a quite good idea of what the method can achieve when the point densities are reasonable. The *triple_hecate* and *ajax* models are highly compliant meshes, not only because their facets are fully in Delaunay, but also because they were constructed from a set of points, using some reconstruction algorithm similar to the convection process. However, this kind of mesh is more and more widespread since a large class of objects are obtained from scanning and reconstruction.

Besides, we can notice that the rates associated to the coding of non Delaunay facets are quite bad facing their small number, and heavy penalize the global rates. Indeed, the GD coder is not optimal for sparse connectivity, and it should be possible to find a better way to code this small set of facets.

The last column of the table shows the computing user times in seconds for the connectivity compression / decompression of the meshes on a Pentium IV 3.0 Ghz 2 Go RAM computer. Globally, these times probably higher compared to classical methods that explicitly encode the whole connectivity, using some surface-based traversal of the mesh vertices. One can incriminate the precomputation and the 3D traversal of the Delaunay triangulation required by the convection process. This constraint particularly intends our method to applications where storage space or network bandwidth are more limited resources than processing power. However, a second version of the algorithm could be developed where Delaunay computation is not explicit any more. The current version also encounters timings limitations when the mesh is not entirely included in Delaunay. This is due to intersection determinations between the Delaunay triangulation and non-Delaunay mesh facets.

We can compare our algorithm to the only other compression method using reconstruction through the 3 meshes we have in common: the method [44] obtains 1.19 bpv for the horse, 2.63 bpv for the fandisk, and 1.18 bpv for the bunny, which is globally slightly higher than the rates of the table I. Regarding the methods [15], [16], [18], derived from the Touma and Gotsman's coding principle [5], they obtain rates around 1.5 bpv for usual meshes, and can achieve very low rates for highly regular meshes where vertex degrees are almost constant. We don't have any result of these algorithms for meshes fully included in Delaunay, but there is no particular reason why their rates would be better in such cases.

Another limitation of the algorithm is related to the memory footprint. Indeed, in its current implementation (using the *CGAL* library [55]), the method needs to store the entire Delaunay tetrahedralization enriched with additional information. The memory cost of this structure is 30 bytes per vertex (geometry: 12, OFF file production: 8, Delaunay tetrahedralization: 4, vertex density information: 4 + 2 booleans) and 35 bytes per tetrahedron (Delaunay tetrahedralization: 32, surface and compression information: 3), given that for meshes meeting good sampling conditions (*ie.* meshes whose local density of vertices is proportional to the distance to the skeleton), the number of tetrahedra is nearly proportional to the number of vertices [56]. Practically, this limits our method to meshes containing about 10 millions of triangles. Nevertheless, out-of-core or even streaming adaptations are conceivable since it has been shown that the convection algorithm could be applied to a data stream organized in successive layers [57]. In the same way, the GD method used in this article to compress the non Delaunay facets is limited for now to meshes under about 1 million of vertices, but an out-of-core adaptation is currently studied. Therefore we cannot give the compression ratio of the non Delaunay submesh for the *UNC powerplant*.

Tab. I contains numerical information about 4 models generated by *CAD* softwares: *fandisk*, *horse*, *ford* and *UNC powerplant*. As said above, our method is clearly not appropriate for this kind of model. Indeed, *CAD* meshes are often characterized by large triangles whose size is absolutely not related to the distance to the skeleton (see Fig. 6 (a) and (b)). Moreover, many of these triangles do not belong to the 3D Delaunay triangulation, even if, in numerous models, a single flip into cocircular quadruplets or a slight perturbation over vertex coordinates would drastically reduce the ratio of non Delaunay facets. This situation particularly occurs in large planar regions containing a lot of coplanar quadruplets, as can be seen on Fig. 3. For this kind of model, some other lossless compression methods should give better results [5], [15], [37], although none of them has been specially designed for this class of meshes.

VI. Conclusion and Future Work

We have presented a new method of lossless single-rate connectivity compression based on a semiautomatic reconstruction process able to deduce most of the mesh connectivity from its sole vertex set, and occasionally guided through compact codes that alter its default behavior when necessary. This method is originally suitable for Delaunay embedded meshes. We have also described a generalization removing this constraint inherent to the convection algorithm, using a separated coding of non Delaunay patches, as well as heuristics minimizing the number of driving codes during the convection process. So the final algorithm is able to compress any kind of 3D mesh, including non manifold ones, without any constraint on the topological genus or the number of connected components. After a probabilistic modelling and an entropic coding of the output, the numerical results show a substantial improvement above the current state of the art, with an average rate of 1 bit per vertex, reaching below 0.1 bpv for well sampled meshes. In addition, the algorithm can be used in parallel with currently most performing geometric compression methods, which results in very competitive overall rates.

The main limitations of the method might be its computing times, relatively high compared to some of the state-of-the-art algorithms, and its memory footprint, which exclude for now the compression of gigantic meshes on standard 32-bit personal computers. As a result, the optimization of the compression, and above all, of the decompression algorithm remains a primary perspective for this work.

The compression rates could probably be improved through more accurate probability models for the entropic coder, and even more, through a better coding of the non Delaunay facets. Similarly, it could be possible to keep improving the behavior of the convection algorithm, by opening the facets more accurately, according to some smarter criterion than the local vertex distribution. However, we are currently working on the generalization of this work to multiresolution, which is surely the most important perspective of this paper. This will be made possible by progressive insertion of vertices in the convection surface [58], under the constraints imposed by a kd-tree type progressive geometric coder. About the GD coder, a last significant perspective consists in integrating it to the convection process, in order to improve the geometric prediction by using connectivity information.

Finally, we would like to add that the impressive results obtained with the most favorable models illustrate the relevance of the research fields focused on Delaunay meshes, in particular the methods that aim at producing such models from point clouds or from non Delaunay meshes [59], [60].

Acknowledgment

This research takes place into the ACI projects *Eros3D* and *Triangles*, supported by the French *Centre National de la Recherche Scientifique* (*CNRS*). Some of the 3D models appearing in this paper are provided courtesy of *AIM@Shape* (*IMATI* and *INRIA*). The *triple_hecate* model is provided courtesy of *Christian Lahanier* from *C2RMF* (*Musée du Louvre*, *Département des Antiquités Etrusques et Romaines*, marbre, *MA 2594*, *C2RMF: FZ31452*). The *ajax* model is provided courtesy of *Jotero GbR*. The algorithms have been implemented thanks to the *CGAL* (*Computational Geometry Algorithms Library*) library [55]. Finally, we would like to thank the anonymous reviewers for their constructive appreciations and helpful suggestions.

References

[1] M. Deering, "Geometry compression," in *SIGGRAPH 95 Conference Proc.*, 1995, pp. 13–20.

[2] F. Evans, S. Skiena, and A. Varshney, "Optimizing triangle strips for fast rendering," in *IEEE Visualization 96 Conference Proc.*, 1996.

[3] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Transactions on Graphics*, vol. 17, no. 2, 1998.

[4] S. Gumhold and W. Strasser, "Real time compression of triangle mesh connectivity," in *SIGGRAPH 98 Conference Proc.*, 1998, pp. 133–140.

[5] C. Touma and C. Gotsman, "Triangle mesh compression," in *Graphics Interface 98 Conference Proc.*, 1998, pp. 26–34.

[6] J. Li and C.-C. J. Kuo, "A dual graph approach to 3d triangular mesh compression," in *IEEE International Conference on Image Processing Proc.*, 1998.

[7] C. Bajaj, S. Cutchin, V. Pascucci, and G. Zhuang, "Error resilient streaming of compressed vrml," University of Texas, Austin, Tech. Rep., 1999.

[8] C. Bajaj, V. Pascucci, and G. Zhuang, "Single resolution compression of arbitrary triangular meshes with properties," *Computational Geometry : Theory and Applications*, 1999.

[9] S. Gumhold, S. Guthe, and W. Strasser, "Tetrahedral mesh compression with the cut-border machine," in *IEEE Visualization 99 Conference Proc.*, 1999.

[10] J. Rossignac, "Edgebreaker : Connectivity compression for triangle meshes," *IEEE Transactions on Visualization and Computer Graphics*, pp. 47–61, 1999.

[11] J. Rossignac and A. Szymczak, "Wrap&zip : Linear decoding of planar triangle graphs," *Computational Geometry : Theory and Applications*, 1999.

[12] D. King and J. Rossignac, "Guaranteed 3.67v bit encoding of planar triangle graphs," in *Canadian Conference on Computational Geometry Proc.*, 1999.

[13] M. Isenburg and J. Snoeyink, "Mesh collapse compression," in *Symposium on Computational Geometry*, 1999.

[14] M. Isenburg, "Triangle fixer : Edge-based connectivity encoding," in *16th European Workshop on Computational Geometry Proc.*, 2000.

[15] P. Alliez and M. Desbrun, "Valence-driven connectivity encoding for 3d meshes," in *Eurographics 2001 Conference Proc.*, 2001.

[16] H. Lee, P. Alliez, and M. Desbrun, "Angle-analyzer: A triangle-quad mesh codec," in *Eurographics 2002 Conference Proc.*, 2002.

[17] V. Coors and J. Rossignac, "Delphi : Geometry-based connectivity prediction in triangle mesh compression," *The Visual Computer*, vol. 20, no. 8-9, pp. 507–520, 2004.

[18] F. Kaelberer, K. Polthier, U. Reitebuch, and M. Wardetzky, "Freelence - coding with free valences," in *Eurographics 2005 Conference Proc.*, 2005.

[19] B. S. Jong, W. H. Yang, J.-L. Tseng, and T. W. Lin, "An efficient connectivity compression for triangular meshes," in *ACIS-ICIS*, 2005, pp. 583–588.

[20] P. Alliez and C. Gotsman, *Recent Advances in Compression of 3D Meshes*. Springer, 2004.

[21] C. Gotsman, S. Gumhold, and L. Kobbelt, *Simplification and Compression of 3D Meshes*. Springer, 2002.

[22] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3d mesh compression : A survey," *ELSEVIER Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688–733, 2005.

[23] M. Isenburg and S. Gumhold, "Out-of-core compression for gigantic polygon meshes," in *SIGGRAPH 2003 Conference Proc.*, 2003.

[24] M. Isenburg and P. Lindstrom, "Streaming meshes," in *IEEE Visualization 2005 Conference Proc.*, 2005.

[25] M. Isenburg, P. Lindstrom, and J. Snoeyink, "Streaming compression of triangle meshes," in *Eurographics Symposium on Geometry Processing*, 2005.

[26] M. H. Gross, O. G. Staadt, and R. Gatti, "Efficient triangular surface approximations using wavelets and quadtree data structures," *IEEE Transactions on Visualization and Computer graphics*, 1996.

[27] A. Certain, J. Popović, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle, "Interactive multiresolution surface viewing," in *SIGGRAPH 96 Conference Proc.*, 1996.

[28] M. Lounsbery, T. DeRose, and J. Warren, "Multiresolution analysis for surfaces of arbitrary topological type," *ACM Transactions on Graphics*, 1997.

[29] O. G. Staadt, M. H. Groos, and R. Weber, “Multiresolution compression and reconstruction,” in *Eurographics 98 Conference Proc.*, 1998.

[30] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder, “Normal meshes,” in *SIGGRAPH 2000 Conference Proc.*, 2000.

[31] A. Khodakovsky, P. Schröder, and W. Sweldens, “Progressive geometry compression,” in *SIGGRAPH 2000 Conference Proc.*, 2000.

[32] A. Khodakovsky and I. Guskov, “Normal mesh compression,” in *to appear*, 2000.

[33] Z. Karni and C. Gotsman, “Spectral compression of mesh geometry,” in *SIGGRAPH 2000 Conference Proc.*, 2000.

[34] A. Szymczak, J. Rossignac, and D. King, “Piecewise regular meshes : Construction and compression,” *Graphical Models*, vol. 64, no. 3-4, pp. 183–198, 2002.

[35] M. Attene, B. Falcidieno, M. Spagnuolo, and J. Rossignac, “Swingwrapper : Retiling triangle meshes for better edgebreaker colmpression,” *ACM Transactions on Graphics*, vol. 22, no. 4, pp. 982–996, 2003.

[36] O. Devillers and P.-M. Gandoin, “Geometric compression for interactive transmission,” in *IEEE Visualization 2000 Conference Proc.*, 2000.

[37] P.-M. Gandoin and O. Devillers, “Progressive lossless compression of arbitrary simplicial complexes,” in *ACM SIGGRAPH Conference Proc.*, 2002.

[38] J. Peng and C.-C. J. Kuo, “Geometry-guided progressive lossless 3d mesh coding with octree decomposition,” in *ACM SIGGRAPH Conference Proc.*, 2005.

[39] H. Hoppe, “Progressive meshes,” in *SIGGRAPH 96 Conference Proc.*, 1996.

[40] J. Popović and H. Hoppe, “Progressive simplicial complexes,” in *SIGGRAPH 97 Conference Proc.*, 1997.

[41] Y.-S. Kim, D.-G. Park, H.-Y. Jung, and H.-G. Cho, “An improved TIN compression using Delaunay triangulation,” in *Pacific Graphics 99 Conference Proc.*, October 1999.

[42] O. Devillers, R. Estkowski, P.-M. Gandoin, F. Hurtado, P. Ramos, and V. Sacristán, “Minimal set of constraints for 2D constrained Delaunay reconstruction,” *International Journal of Computational Geometry and Applications*, vol. 13, pp. 391–398, 2003.

[43] M. Gopi, S. Krishnan, and C. T. Silva, “Surface reconstruction based on lower dimensional localized delaunay triangulation,” *Comput. Graph. Forum*, vol. 19, no. 3, 2000.

[44] T. Lewiner, M. Craizer, H. Lopes, S. Pesco, L. Velho, and E. Medeiros, “Gencode : Geometry-driven compression in arbitrary dimension and co-dimension,” in *Sibgrapi*, 2005.

[45] M. Botsch, A. Wiratanaya, and L. Kobbelt, “Efficient high quality rendering of point sampled geometry,” 2002. [Online]. Available: citeseer.ist.psu.edu/botsch02efficient.html

[46] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, “The ball-pivoting algorithm for surface reconstruction,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, /1999. [Online]. Available: citeseer.ist.psu.edu/bernardini99ballpivoting.html

[47] R. Mencl and H. Müller, “Interpolation and approximation of surfaces from three-dimensional scattered data points,” in *State of the Art Reports, Eurographics*, 1998, pp. 51–67.

[48] N. Amenta and M. Bern, “Surface reconstruction by voronoi filtering,” *Discrete Comput Geom.*, vol. 22, no. 4, pp. 481–504, 1999.

[49] F. Cazals and J. Giesen, “Delaunay triangulation based surface reconstruction : Ideas ans algorithms,” Technical Report 5393, INRIA, Tech. Rep., 2002.

[50] H. Edelsbrunner, “Surface reconstruction by wrapping finite point sets in space,” *B. Aronov, S. Basu, J. Pach and Springer-Verlag M. Sharir, editors, Ricky Pollack and Eli Goodman Festschrift*, pp. 379–404, 2002.

[51] J. Giesen and M. John, “Surface reconstruction based on a dynamical system,” in *Proc. Eurographics*, 2002.

[52] H.K.Zhao, S. Osher, and R. Fedkiw, “Fast surface reconstruction using the level set method,” in *Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM)*, 2001.

[53] R. Chaine, “A geometric convection approach of 3-d reconstruction,” in *Eurographics Symposium on Geometry Processing, Aachen, Germany*, 2003, pp. 218–229.

[54] H. Edelsbrunner, M. Facello, and J.Liang, “On the definition and the construction of pockets in macromolecules,” *Discrete Appl. Math*, vol. 88, pp. 83–102, 1998.

[55] “CGAL, Computational Geometry Algorithms Library,” http://www.cgal.org.

[56] D. Attali and J.-D. Boissonnat, “A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces,” in *Symposium on Solid Modeling and Applications*, 2002, pp. 139–146.

[57] R. Allègre, R. Chaine, and S. Akkouche, “A streaming algorithm for surface reconstruction,” in *Symposium on Geometry Processing*, 2007, pp. 79–88.

[58] ——, “Convection-driven dynamic surface reconstruction,” in *Proc. Shape Modeling International, IEEE Computer Society Press*, 2005, pp. 39–48.

[59] R. Dyer, H. Zhang, and T. Möller, “Delaunay mesh construction,” in *Symposium on Geometry Processing*, 2007, pp. 273–282.

[60] ——, “Voronoi-delaunay duality and delaunay meshes,” in *Symposium on Solid and Physical Modeling*, 2007, pp. 415–420.

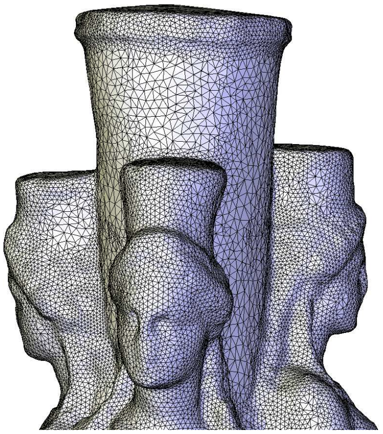
Raphaëlle Chaine ’s research interests include geometric modeling and computational geometry. After completing her PhD in computer science from the Université Claude Bernard in the domain of point set surfaces analysis, Raphaëlle joined the PRISME team at INRIA as she got hired by the Université de Nice as an associate professor. Since 2003, she is now involved in the LIRIS (CNRS) research laboratory and associate professor in Computer Science at the University of Lyon. She graduated with an engineer degree of the ”Ecole Centrale” of Lyon.

Pierre-Marie Gandoin received the Ph.D. degree in computer science with a dissertation on the progressive and lossless compression of meshes, from the University of Nice - Sophia Antipolis, France, in 2001.

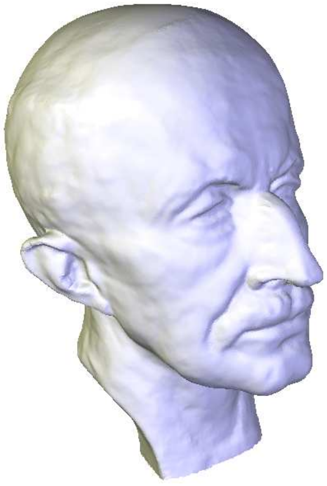
He is currently an associate professor at the University of Lyon. His research interests include design of algorithms and software systems for mesh modeling, compression and visualization.

Céline Roudet received her Engineering Degree in Computer Science in 2004 and her M.S. degree in Image Processing in 2005 from INSA of Lyon (France). She is actually Ph.D. student at LIRIS Laboratory in the University Claude Bernard of Lyon (France).

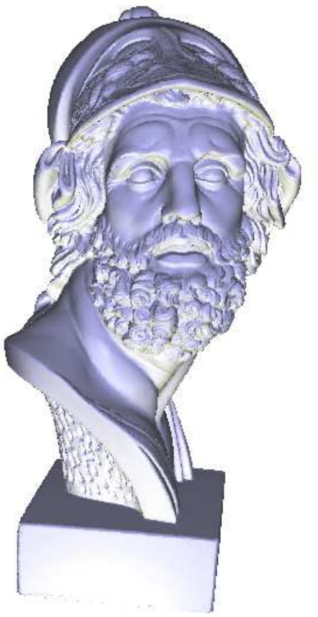
Her research interests include 3D digital image processing, geometric modeling and more precisely multiresolution analysis, remeshing and progressive compression of meshes.



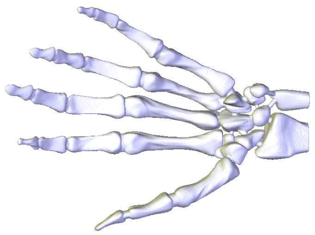
(a)



(b)

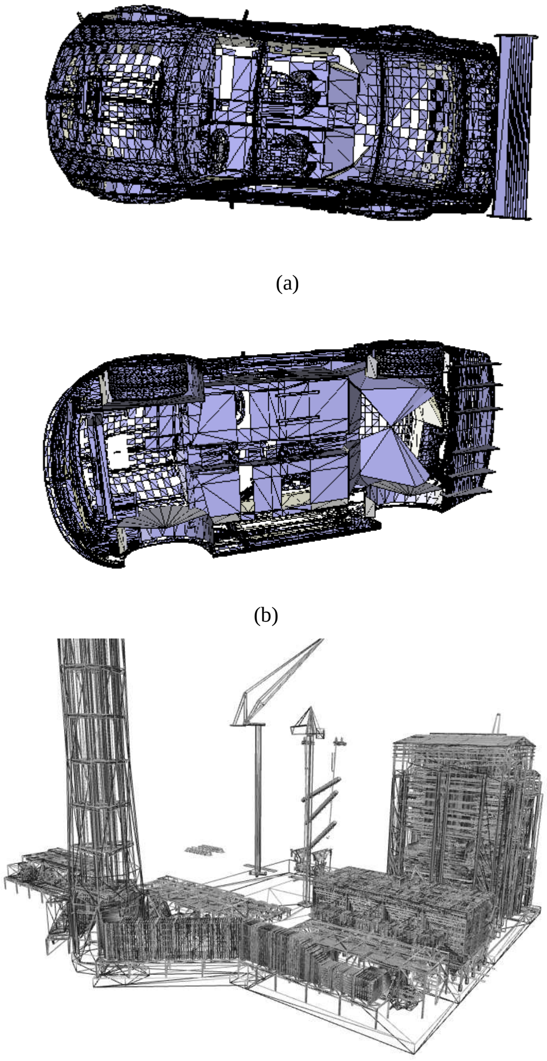


(c)



(d)

Fig. 5. (a) triple_hecate (detail): 75729 vertices and 151462 triangles coded with 0.04 bits per vertex, (b) max_planck: 199169 vertices and 398043 triangles, 0.90 bits per vertex, (c) ajax: 273383 vertices and 547117 triangles, 0.14 bits per vertex, (d) hand: 327290 vertices and 654596 triangles, 0.19 bits per vertex. These models resulting from scans are usually characterized by a small number of non Delaunay triangles. The vertex density locally reflects the distance to the skeleton and these meshes are probably produced by Delaunay based reconstruction algorithms.

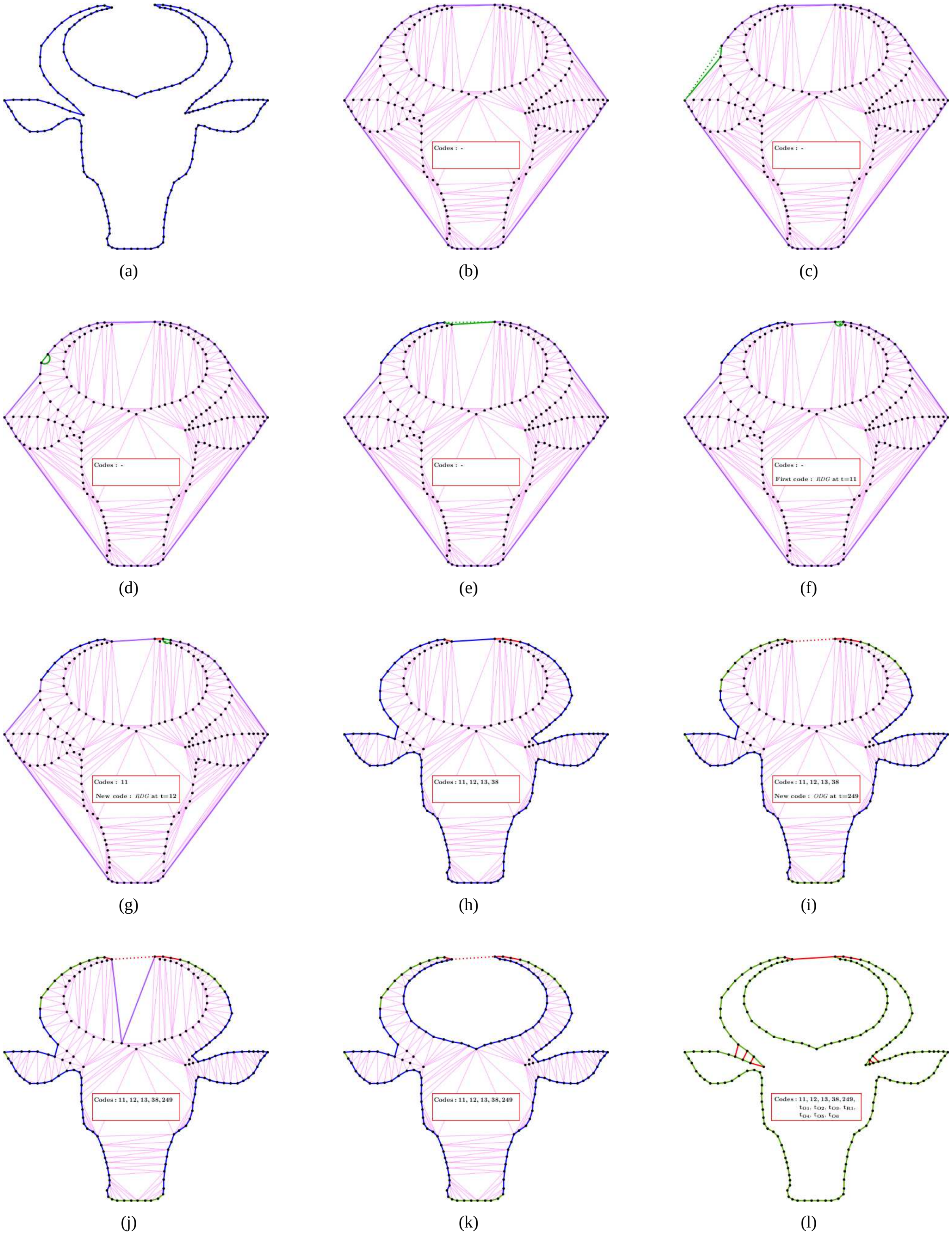


(a)

(b)

(c)

Fig. 6. CAD meshes are essentially unfavorable to our method: (a) and (b) non Delaunay triangles of the *ford* model, (c) UNC powerplant: 11070509 vertices and 12748510 triangles coded with more than 2.16 bits per vertex.


Codes : -
Codes : -
Codes : -
Codes : -
Codes : -
First code : RDG at t=11
Codes : 11
New code : RDG at t=12
Codes : 11, 12, 13, 38
Codes : 11, 12, 13, 38
New code : ODG at t=249
Codes : 11, 12, 13, 38, 249
Codes : 11, 12, 13, 38, 249
Codes : 11, 12, 13, 38, 249,
t_{O1}, t_{O2}, t_{O3}, t_{R1},
t_{O4}, t_{O5}, t_{O6}

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Fig. 2. a) Mesh M to be coded b) Delaunay triangulation of M vertices and initialization of S c) The first visited facet has been opened (t=1) d) The second facet has been retained (t=2) e) The tenth visited facet has been opened (t=10) f) Production of a RDG event at t=11, against the convection rules g) RDG event at t=12 h) Two kinds of facets in S at stabilization of the convection process : red facets for which RDG events have occurred, and blue facets that have been retained (whereas some of them are not present in M) i) Each retained facet (in blue) is considered in turn. An ODG event is produced for the 27th facet of that kind at time t=249 (previously examined facets are green colored) j) The convection process is locally relaunched at the level of the opened facet k) The local convection process is stabilized l) ODG events combined with local convection processes are pursued until the initial mesh is reached. t_{Oi} and t_{Ri} denote time steps when further ODG and RDG events occur.