Canonical Triangulation of a Graph, with a Coding Application

Luca Castelli Aleardi, Olivier Devillers

▶ **To cite this version:**

Luca Castelli Aleardi, Olivier Devillers. Canonical Triangulation of a Graph, with a Coding Application. RR-5231, INRIA. 2004, pp.24. inria-00070765

HAL Id: inria-00070765

https://inria.hal.science/inria-00070765v1

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Canonical Triangulation of a Graph, with a Coding Application

Luca Castelli Aleardi — Olivier Devillers

N° 5231

8 Juin 2004

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--5231--FR+ENG

INRIA
SOPHIA ANTIPOLIS

Canonical Triangulation of a Graph, with a Coding Application

Luca Castelli Aleardi*, Olivier Devillers †

Thème SYM — Systèmes symboliques
Projet Geometrica

Rapport de recherche n° 5231 — 8 Juin 2004 — 23 pages

Abstract: For compression of 3D meshes, we propose an algorithm to encode the transformation of a polygonal mesh into a triangular mesh, getting a full coder by combination with a triangular coder. In this way, we benefit of the possibility of choosing a triangular coder relevant for a particular kind of mesh. Let G be a 3-connected simple planar graph with e edges. We introduce a special type of triangulation T_G of G, based on the canonical orderings of this class of graphs. We show how to reconstruct the original graph starting from its canonical triangulation T_G, using only the face degrees of G: this detriangulation phase takes linear time and requires at most $e + o(e)$ bits. Our canonical coding of the detriangulation of G can be combined with any triangulation coder to obtain a full coder of the connectivity of any genus 0 polygonal mesh.

Key-words: Mesh compression, connectivity encoding, canonical orderings, 3-connected planar graphs

This work has been supported in part by ARC INRIA Telegeo

* INRIA - Geometrica and LIX, Ecole Polytechnique, 91128 Palaiseau
† INRIA - Geometrica

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Triangulation canonique d'un graphe, avec une application de codage

Résumé : Nous proposons un algorithme de codage pour la transformation d'un maillage 3D polygonal en un maillage triangulaire, obtenant un codeur complet en combinant un codeur de maillages triangulaires. De cette facon, nous allons bénéficier de la possibilité de choisir un codeur intéressant pour une classe spéciale de maillages. Soit G un graphe planaire 3-connexe ayant e arêtes. Nous introduisons un type special de triangulation T_G de G, basée sur les ordres canoniques de cette classe de graphes. Nous montrons comment reconstruire le graphe original en partant de sa triangulation canonique T_G, utilisant seulement les degrés des faces: cette phase de detriangulation prend temps linéaire et nécessite au plus de $e + o(e)$ bits. Notre codage canonique de la detriangulation de G peut être combiné avec n'importe lequel codeur de triangulation afin d'obtenir un codeur complet de la connectivité d'un maillage polygonal de genre 0.

Mots-clés : Compression de maillages, codage de la connectivité, ordres canoniques, graphes planaires 3-connexes

1 Introduction

Polygonal meshes are hugely used in several domains, like computer graphics, CAD, simulation, scientific visualization. Recent works in mesh compression have been motivated by applications that need compact representation for storage or transmission of large 3D models. We consider the case of polygonal meshes that are manifold, with no boundaries and without handles (discrete surfaces that are homeomorphic to a sphere), which is equivalent to the coding of 3-connected planar graphs. Higher genus surfaces or holes can be handled by cutting the surface in pieces and filling the holes. Most encoding techniques try to compute from a deterministic traversal of the mesh an enumeration order of the graph representing the connectivity. Several strategies with heuristic or guaranteed bit rate compression bounds have been proposed for triangle ([5], [20], [1], [17], [14], [7] [19]) or polygon meshes ([10], [9] [13],[15]).

1.1 Contribution

This paper explains a deterministic way to triangulate a polygonal mesh and to encode this transformation in the sequence of face degrees in the original mesh. This technique can be used in combination with any algorithm coding a triangular mesh to get a full coder for a polygonal mesh.

In combination with a guaranteed bit rate triangular coder [16] we got a guarantee of 2.62 bits per edge for coding the mesh.

In combination with a valence based coder [20] we get an algorithm very similar to Khodakovsky *et al.* [13] for which the code consists in a sequence of face degrees a sequence of vertex valences and some accidental codes. The difference is that in our case — the valences refer to the triangulated graph instead of the original graph — the sequences are not interleaved — accidental codes concerns only the coding of the triangulation and thus improvements as proposed by Alliez and Desbrun [1] can be used.

By splitting the coding of a polygonal mesh into coding how it is triangulated and various way of coding the triangulation we are able to provide more adaptability to the user.

1.2 Outline

In section 2 we discuss some previous and related works. Section 3 introduces some definitions and concepts about canonical orderings for 3-connected planar graphs. In section 4 we define and compute the canonical triangulation T_G. In section 5 we describe our encoding/decoding algorithm, that performs a traversing of the graph, driven by the connectivity of its canonical triangulation, and which is independent of the particular way we adopt to encode T_G. In section 7 we provide a theoretical analysis. Finally section 8 shows some experimental results which confirm our theoretical upper bound and the comparison with some previous encoding strategies.

2 Graph encoding and related works

2.1 Triangular meshes

One of the first encoding algorithms for the connectivity of triangular meshes, provided with a theoretical analysis, has been described by Rossignac [17]: its *Edgebreaker* can guaranteed an upper bound of 4 b/v (bits per vertex) in the original version. For this strategy and its theoretic bounds several improvements and generalizations have been proposed ([14], [18], [15]). Touma and Gotsman [20] proposed a new approach to encode the connectivity of a triangular mesh, using the valence (degree) of the vertices, and some "accident codes", called *splits*. In this way they exploit the distribution of vertex valences, allowing to achieve very interesting bit rates compression, in the case of regular meshes. Their method is based on a traversal of the graph and a conquest of the edges incident to a given vertex, called focus, lying on the boundary of the exterior face: the expansion of this face is driven by the connectivity of the edges around the focus in a given order. This algorithm has been improved by Alliez and Desbrun [1], who also showed that the cost of coding the valences sequence is at most asymptotically 3.2451 b/v: however it is not known how to provide an upper bound on the length code for valence-based methods, since in the worst case the number of splits is not negligible (see [6] for a theoretical analysis of the near-optimality of valence/based methods).

More recently an optimal algorithm has been found by Poulalhon and Schaeffer [16]: using a bijection between triangulations and a special class of trees and exploiting the properties of minimal realizers for maximal planar graphs, their strategy achieves asymptotically the theoretical bound of 3.2451 b/v established by Tutte [22].

2.2 Polygonal meshes

In the more general case of 3-connected graphs with e edges, the theoretical bound also given by Tutte [23] is 2 b/e (bits per edge). After some pioneer results ([21], [12]) reaching 4 b/e and 3.58 b/e respectively, the current best approach is due to Khodakovski et al. [13] and Isenburg et al. [9], and extends the valence's approach from triangular meshes. The key idea is that for the class of 3-connected simple planar graphs (corresponding to manifold meshes of genus 0), a graph G and its dual G^* must have the same entropy, concerning the connectivity information, in the sense that G and G^* can be encoded with the same amount of bits. Extending the theoretical analysis of Alliez and Desbrun [1] to the polygonal case, they show that valence/degree based approach is near-optimal: without considering the *split codes* that occurs in the traversal of the mesh, the cost of coding separately the two valences/degrees sequences never exceeds asymptotically the upper bound of 2 b/e given by Tutte for polygonal meshes. Unfortunately, even if experimental evidence that the number of *split* is often small in practice, this is false in full generality.

Canonical orderings, a special type of labeling of vertices, were first used by He et al. [8] to describe the connectivity information of a 3-connected planar graph, obtaining a bit rate compression of 2.835 b/e, that improved previous bounds. The best known result of 2.377

b/e has been established by Chuang et al. [3] using again canonical orderings and succinct representations of multiple parentheses. Some interesting recent work, by Bonichon et al. [2], concerns the encoding of more general planar graphs: using the concept of *well-orderly tree*, a generalization of realizers for triangulations, they present a linear time algorithm with an upper bound of 2.90b/e.

We will see in next sections that these ideas, first that 'dualization' does not add information, second that canonical orderings (and realizers) characterize completely the global planarity information of a graph, will be useful for the definition of our encoding method.

3 Preliminaries and definitions

3.1 Canonical orderings for 3-connected planar graphs

Let us define the rightmost canonical ordering of a 3-connected planar graph, an order on the set of its vertices, introduced by Kant [11], generalization of similar results on triangulations ([4]).

Definition 1. *Let $G = (V, E)$ be a 3-connected planar graph with a vertex v_0 on the exterior face. Let $\pi = (V_1, \ldots, V_K)$ be an ordered partition of the vertices V s.t. $V_1 \cup \ldots \cup V_K = V$ et $V_i \cap V_j = \emptyset$. We consider G_k the sub-graph of G induced by the first k sets $V_1 \cup \ldots \cup V_k$ and we denote C_k the exterior face of G_k. We say that π is a* ***canonical ordering*** *if the following conditions are satisfied (see Figures 1 and 2):*

- *$V_1 = \{v_0, v_1\}$, where v_1 lies on the outer-face of G and $(v_0, v_1) \in E$;*
- *$V_K = \{v_{n-1}\}$, where v_{n-1} lies on the outer face, $(v_0, v_{n-1}) \in E$ and $v_{n-1} \neq v_1$;*
- *each C_k $(k > 1)$ is a cycle containing the edge (v_0, v_1);*
- *each sub-graph G_k is 2-connected and internally 3-connected (if we delete 2 interior vertices G_k remains connected);*
- *for each $k \in \{2, \ldots, K-1\}$, one of the 2 conditions holds:*
  1. *V_k is a singleton $\{z\}$, where z belongs to C_k and has at least one neighbor in $G - G_k$;*
  2. *V_k is a chain of vertices $\{z_1, \ldots, z_l\}$ (ordered clockwise), where each vertex z_i has at least one neighbor in $G - G_k$, and the extremities z_1 et z_l have a neighbor on C_{k-1}, and these are the only two incidences of V_k in G_{k-1}.*

It is known that every 3-connected planar graph, with a fixed first vertex v_0 on the exterior face, admit a canonical ordering and this can be computed in linear time (see [11] for more details). Since in general a canonical ordering is not uniquely determined, let us introduce the concept of ***rightmost*** canonical ordering, which is unique, once the edge (v_0, v_1) is chosen (Figure 1 shows an example of a graph induced with its rightmost canonical ordering)

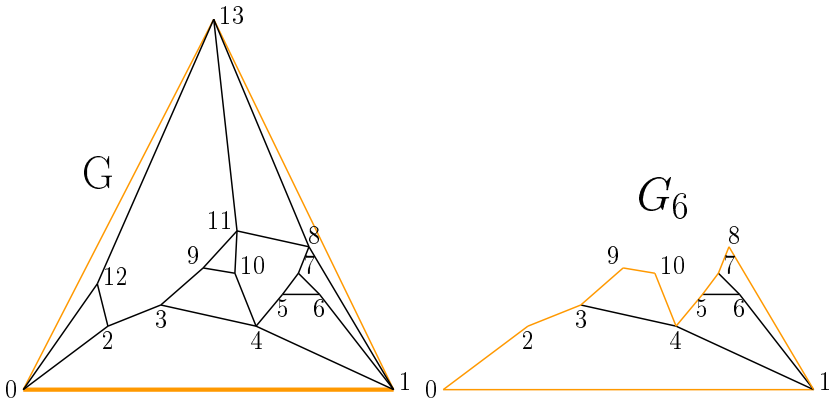

Figure 1: A 3-connected planar graph with its (rightmost) canonical ordering. On the right the subgraph induced by the first 6 sets of the partition π.

Definition 2. *A canonical ordering is* ***rightmost*** *if when we can add at any step two different sets of vertices V_k, $V_{k'}$, with c_l and $c_{l'}$ as left-vertices, and $l < l'$ holds, then $k < k'$ (where l is the index of vertices on the boundary of the exterior face, oriented cw).*

More intuitively, it means that if there are several V_k, sets of vertices or chains to be added to G_{k-1}, then we add always the vertices rightmost, belonging to the exterior face (Figure 2 shows two different canonical ordering of G).

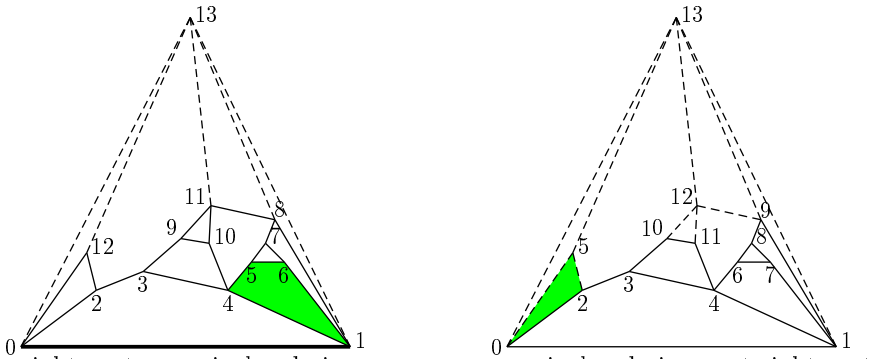

Figure 2: Two different possible canonical orderings for G.

4 Computing the triangulation of G

4.1 Overview

Let us introduce a special kind of triangulation T_G of a 3-connected planar graph G, which will be useful later to define an encoding algorithm for G. We can simply describe our algorithm in term of *edges coloring*. We start with the graph G whose edges are initially black, and we perform a visit of G triangulating it by adding red lines to get T_G at the end. The goal is to visit the faces of the graph following a fixed order (modified version of canonical ordering): during our conquest of the graph we color in blue the original edges existing in G, red lines represents new added edges. As illustrated in Figure 4, the right-most canonical ordering associated to T_G is different from the indexing of the original graph G. We maintain some notations given by Kant [11].

Notations

Let consider a 3-connected planar graph, with a predefined exterior face F_{out} and an incident edge (v_0, v_1).

C_k is the boundary of the outer face at step k: it is a cycle containing the edge (v_0, v_1). C_k defines two distinct regions of G which are simply connected. The vertices lying on C_k (except v_0, v_1) are ordered clockwise.

G_k is the region of G consisting of vertices not yet conquered: edges are black colored (if existing in the original graph) or red colored (if added during the traversal).

G_k will be always 2-connected and internally 3-connected. $G - G_k$ is the region of G already traversed: it consists of red and blue edges.

We call **Focus** the vertex on the boundary C_k we are visiting.

A vertex v ($v \neq v_0, v_1$) on the boundary C_k of the outer face of the subgraph G_k, is **free** if the following conditions are satisfied:

1. v is incident to at least one blue edge (observe that blue edges are in $G - G_k$).
2. its deletion (with its incident edges in G_k) does not change the 2-connexity of the subgraph G_k (black and eventually red edges).

4.2 Description of the algorithm

The algorithm for computing a canonical triangulation of G performs a traversal of the graph, conquering at each step a free vertex and its incident edges in cw order. Our goal is to visit the vertices of G in a "canonical" order and to triangulate its polygonal faces in a given way. Like in previous mesh encoding algorithms ([20], [1], [13]) we visit G extending the boundary C_k (which is also called *cut-border* or *active list* in previous works). The rules that drive our conquest and that determine the choice of the next focus on C_k are based on a modified version of the canonical orderings.

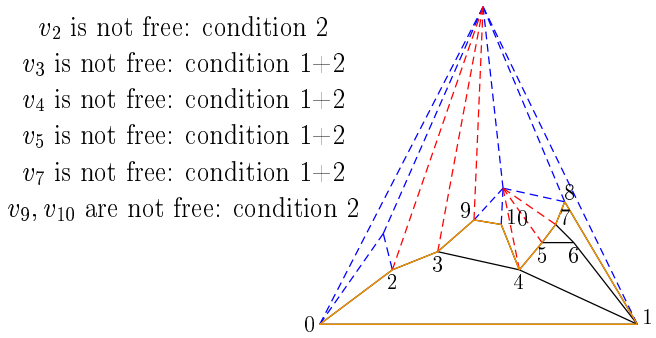

Figure 3: Three vertices have been conquered. This figure shows the subgraph G_{11} with a canonical ordering on its vertices. The boundary of the outer face is drawn in orange: only the vertex v_8 is free in G_{11}, since deleting its incident edges (v_1, v_8) and (v_8, v_7) would not change the 2-connexity of G_{11}.

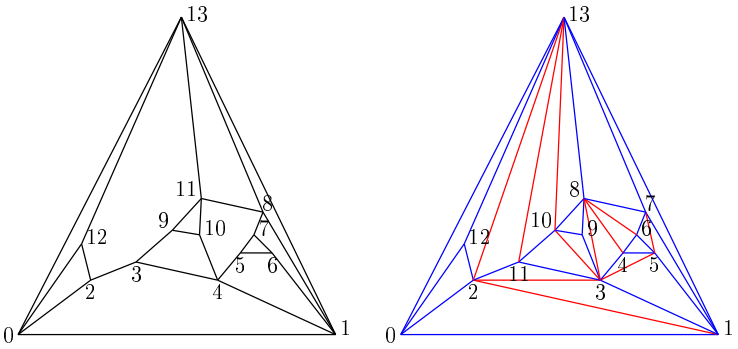

Figure 4: The original graph G and its canonical triangulation T_G: in both cases the two different right-most canonical orderings are shown. Red lines represent *ghost* edges, added to G during its traversal.

Our algorithm never produces auto-intersections of the boundary C_k: this allows us to avoid any accident code for describing the traversal (auto-intersections generate *split* codes in the TG algorithm and its improvements).

Traversal of G

We start with the entire graph G, induced with a face (for sake of simplicity a triangular face) defining the outer face: initially $G_n = G$, $C_n = \{v_0, v_1, v_{n-1}\}$ and all the edges are black. G_{k-1} is obtained from G_k by conquering a vertex and its incident black edges. At each step we consider a vertex focus on C_k (always different from v_0, v_1): depending on the

nature of the focus (whether is a free vertex) and the color of its incident edges, we color and/or triangulate some faces. We compute the sequence of graphs $G_{n-1}, \ldots, G_1$ proceeding as follows:

1. if the focus v is a *free vertex* in G_k then we triangulate its incident faces, by adding the star of *red* edges issued from v.

   Then we update the subgraph G_k by conquering all the edges incident to v: black original edges are colored in blue. The next focus will be the vertex on C_k adjacent to v on its left (this case is illustrated in Figure 5, two first pictures).

2. If the focus v is not *free* but it is incident to at least one blue edge, again we triangulate its incident faces in G_k by adding *red* edges issued from v.

   Since v is not free we cannot delete it without changing the 2-connexity of G_k: then we move the focus on the right on the boundary C_k and we mark v as *not free* and visited (see the third picture in Figure 5, p being the focus).

3. If the focus v is not incident to any blue edge, we skip v without triangulating and we move the focus to its right on C_k (this case is illustrated by the Picture 4 in Figure 5, where the focus is called q).

We iterate this process until all the faces of G have been visited, triangulated if necessary, and colored: at the end the subgraph G_1 is reduced to the edge (v_0, v_1) (Figure 5 illustrates the development of this algorithm and the different rules for triangulating the polygonal faces of G).

The algorithm we have describe always terminates computing a triangulation T_G of G, obtained by adding to G the *red* edges mentioned above: we refer to the proof of Kant [11] which shows that such an algorithm always terminates (this is true because there must be one free vertex on C_k).

Remark 1. *Note that at each step the region $G - G_k$ (already visited) is a triangulation consisting of blue and red edges.*

When we visit a vertex (a new focus) we always triangulate, unless it is not incident to a blue line in $G - G_k$.

We conquest only free vertices, with their incident triangular face in G_k. Their deletion does never change the 2-connexity of G_k.

At each step the vertices lying on C_k to the left of the focus have been all visited and declared not free. *The boundary C_k must be updated only when a free vertex is visited and conquered (coloring its incident edges).*

Note that our traversal does compute a canonical ordering for the triangulation T_G: even if this ordering is not right-most, *we will see in next section that it allows to reconstruct G using only the sequence of face degrees.*

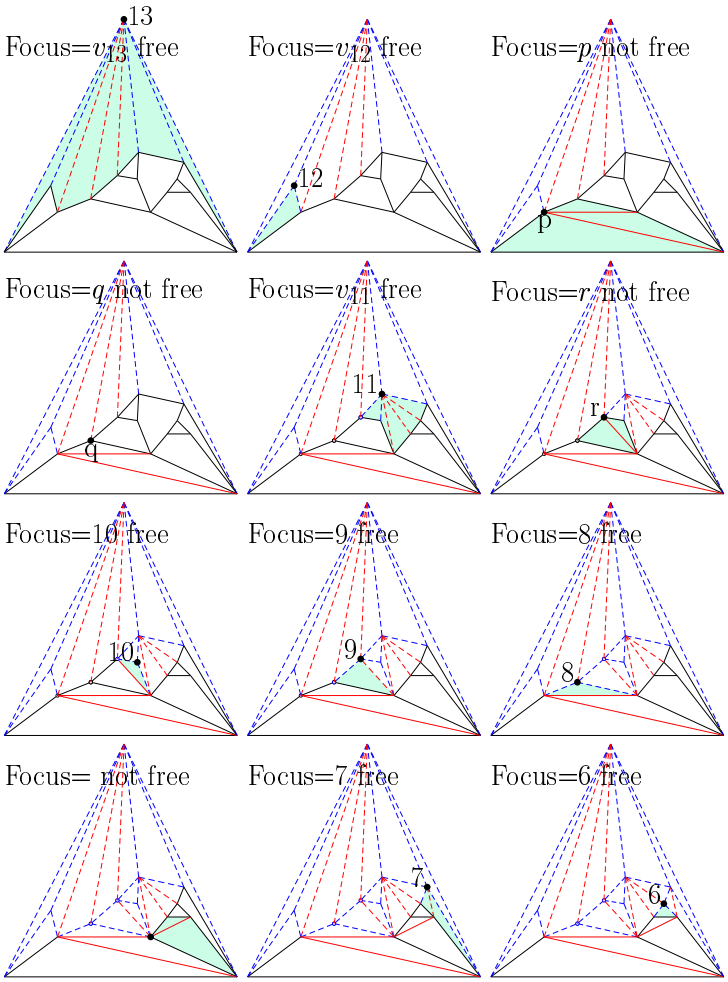

Figure 5: This figure shows the development of the algorithm for computing the canonical triangulation T_G. The filled circle represents the vertex focus, moving on the boundary of the exterior face; its incident faces are filled in light green. Ghost edges are drawn in red. Dashed lines correspond to the region of G already visited.

4.3 Time complexity

Theorem 1. *Let G be a 3-connected planar graph with n vertices. The previous algorithm computes the canonical triangulation T_G in $O(n)$ time.*

Proof. Let us provide an analysis of the traversal phase: for the triangulating phase, which consists in adding and coloring edges, it suffices to observe that only $O(n)$ new edges are added to G, because of Euler's formula.

As noticed before, all the vertices on the boundary C_k lying to the left of the focus have been declared not free (and visited): then, as we look for the left-most free vertex, we do not need to examine every vertex on C_k, but only the left neighbor on C_k of the last free vertex conquered.

As we enter in a vertex $v \in C_k$ going through an incident edge (v_{left}, v) or (v, v_{right}) (on C_k), and each edge is crossed only twice before its conquest, our traversal can be performed in $O(n)$ steps.

Finally we only need to observed that testing if a vertex v is free takes constant time. It is easy to verify if v is incident to a blue edge in $G - G_k$ (maintaining a boolean variable *neighbor(v)*); while for testing if its deletion does change the *connectivity* of G_k we can introduce the variable $out_v(v)$ giving the number of vertices on C_k adjacent to v. Notice that a vertex $v \in C_k$ is free if and only if $out_v(v) = 2$. This variable can easily increased or decreased during the updating of G_k (a similar argument was introduced by Kant [11]). □

5 Detriangulation phase

Before defining an encoding algorithm based on our canonical triangulation, we prefer to describe a strategy for reconstructing G.

During this second phase our *detriangulating* algorithm start from a triangulation T_G and only knows a sequence of face degrees of G. Let us denote $\Sigma_G = \{\sigma_1, \ldots, \sigma_f\}$, with $\sigma_i \in \{3, 4, \ldots, \max_{F \in G} deg(F)\}$ a sequence of the degrees of the faces of G, traversed in a given order. Our goal is to visit T_G and compute a 2-coloration of its edges.

We perform the same conquest of the triangulation, following our modified version of the rightmost canonical ordering: the traversing is driven by the connectivity of T_G and by the information (the color) corresponding to the part of the original graph G, yet traversed and reconstructed.

This algorithm start with the entire triangulation T_G (whose edges are initially black) induced with a fixed initial outer face: at step k of our traversal we conquest a free vertex on the boundary C_k and its incident edges updating the subgraph G_k. The choice of the next *focus* on C_k is given by the same rule of the algorithm in previous section; at any step the vertex to be conquered (with its incident edges) will be the left-most free vertex v on C_k ($v \neq v_0, v_1$).

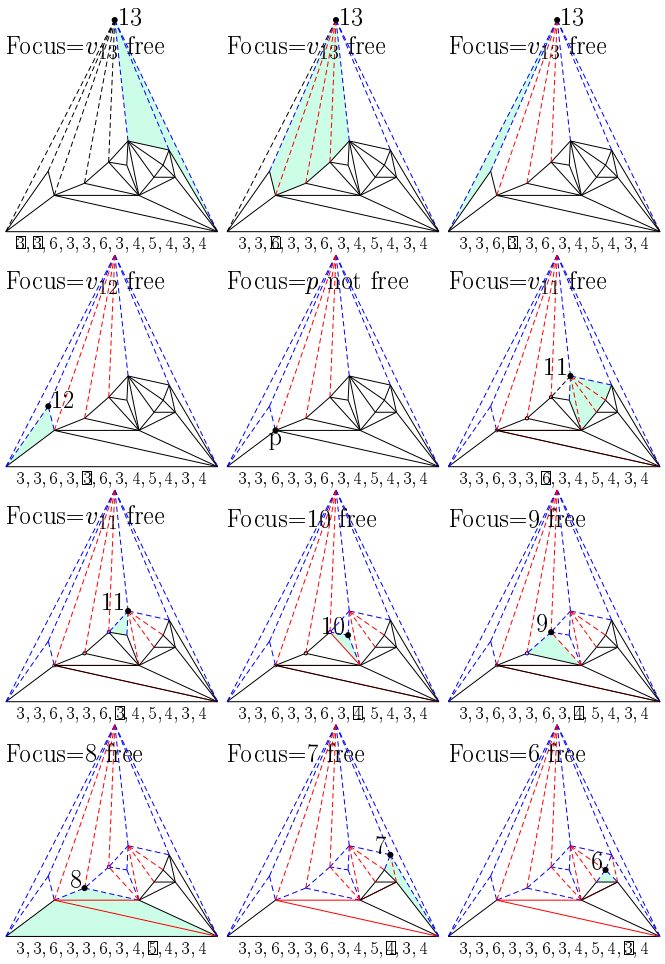

Figure 6: Development of the detriangulating phase. The algorithm starts with a triangulation T_G and a sequence $\Sigma = \{3, 3, 6, 3, 3, 6, 3, 4, 5, 4, 3, 4\}$, performing a traversal of the graph as in the previous section.

Reading the sequence Σ_G and coloring edges

At each step, when we are treating a free vertex v on C_k (and before conquering it) we visit its incident faces in G_k, turning around v in cw order.

Let $\triangle_1, \ldots, \triangle_r$ the triangles incident to v in G_k, listed in cw order, and $(v, p_0), \ldots, (v, p_r)$ the corresponding edges issued from v (see Figure 6, Pictures 1-3).

When we traverse a triangle $\triangle_i$, for the first time, we read a new symbol σ from Σ_G. If $\sigma > 3$ it does mean that $\triangle_i$ has been created during the triangulating phase, by adding some red edges to G.

Then we traverse the next $\sigma - 3$ triangles $\triangle_{i+1}, \ldots, \triangle_{i+\sigma-3}$, assigning red color to edges $(v, p_i), \ldots, (v, p_{i+\sigma-3})$, and coloring in blue the two edges (v, p_{i-1}) and $(v, p_{i+\sigma-2})$. Next triangle to be visited will be $\triangle_{i+\sigma-3+1}$ (see Figure 6, Pictures 1,2,3).

Special case

Let us observe that we have to distinguish a crucial situation, when we read a new symbol σ from Σ_G such that $\sigma - 2$ exceeds the number of incident triangles:this can happen when it remains at most one triangle not yet visited around v (see Figure 6, Pictures 8 and 10).

In this case we mark as visited the triangle $\triangle = (p_r, v, p_{r-1})$, coloring in red (p_r, p_{r-1}), and in blue the others two edges.

We also visit (without removing) the next $\sigma - 2$ triangles incident to p_r in G_k: all these triangles, together with $\triangle = (p_r, v, p_{r-1})$, will be considered as sub-triangles of the *same* face in the original graph G (we only need to color in red the incident edges issued from p_r that separate these triangles). Then we continue our traversal, as described before, choosing as next focus on C_k the vertex p_r.

Remark 2. *Notice that visited faces may be removed only in a later step, when become incident to a free vertex: when a visited triangle will be traversed for the second time it will not produce the scan of a new symbol from Σ (see Figure 6, Picture 9).*

Constraint on the size of the reconstructed faces

Let us observe that the combinatorial information of T_G and C_k can be used at any step to define some constraints on the faces that will be reconstructed.

When we conquest a free vertex v on C_k we know its interior degree (the number of its incident edges in $G - G_k$): for example, if the interior degree of v is 3, this imply that we reconstruct a triangle (or a face of degree at most 4) depending on the next scanned symbol: we will have a triangle if $\sigma = 3$, a quad if $\sigma = 4$; a degree greater then 4 would cause an error in our algorithm (see Figure 6: in Picture 11 the vertex focus v_7 has 3 as interior degree).

If the interior degree of the conquered vertex is $d \geq 3$, the maximum size of the reconstructed face will be $d + 1$. More precisely, the sum of the sizes of the next m faces to be reconstructed incident to v (corresponding to the next symbols $\sigma_t, \ldots, \sigma_{t+m}$) must satisfy:

$$\sum_{j=t}^{j \leq m+t} (\sigma_j - 2) \leq d + 1$$

If the focus v is only incident to a triangle (p_r, v, p_{r-1}), the situation described as *special case* could happen: the previous constraint holds, taking $d = deg_{int}(p_r)$ (the interior degree of its left adjacent vertex on C_k).

6 Encoding application

Finally we can describe an encoding strategy for genus 0 polygonal meshes.

This time we start with the entire graph G and an empty sequence Σ_G: we compute at first the canonical triangulation T_G.

During the traversal of G we generate a sequence Σ_G, consisting of the face degrees: the faces of G (once triangulated) are visited following the rules described in the previous section (detriangulation phase).

When we enter in a face F_j for the first time (in a triangle $\triangle \subset F_j$) we add to Σ_G the symbol $\sigma = degree(F_j)$, corresponding to the degree of the face F_j and we mark as visited the entire face and its internal edges.

The process terminates when all the faces of G have been traversed and T_G is reduced to the last triangle.

It remains to encode the face degree sequence: applying an adaptive arithmetic coder ([24]) to compress the sequence Σ_G, we can achieve asymptotically the theoretical bound given by the entropy of Σ_G (refer to the next section for an entropy analysis).

Finally we only have to encode the connectivity of T_G: this phase does not depend on our algorithm and can be carried out choosing one of the known encoding strategy for triangular meshes ([20], [17], [1], [16]) .

If we use the recent optimal algorithm for planar triangulations by Poulalhon and Schaeffer [16], the total cost does not depend on the way we obtained T_G and it is asymptotically $3.2451\, n$ bits, where n is the number of vertices of G (and of T_G too).

Decoding phase

The decoding algorithm start with the code representing a canonical triangulation T_G, and the code corresponding to the sequence Σ_G. At first we decode in linear time T_G with a decoder for triangle mesh, and the sequence Σ_G. Then it suffices to apply our detriangulation algorithm to obtain the original graph G.

7 Theoretical analysis

7.1 Asymptotic bit rate upper bound

Finally we give an upper bound to the amount of bits necessary to reconstruct a graph G starting from its canonical triangulation T_G. Since only the sequence of face degrees (traversed in a given order) is needed during the detriangulation phase, we would interested in evaluating the entropy of this sequence.

For this we refer to the theoretical analysis provided by [13] concerning the total entropy of the valence/degree sequences in a 2-manifold polygon mesh of genus 0. Our result can be expressed in the following way:

Theorem 2. *Let T_G a canonical triangulation of a 3-connected simple planar graph with v vertices, f faces and e edges. By detriangulating T_G we can reconstruct the original graph G in linear time using at most*

$$e \; + o(e) \; \textit{bits}$$

Proof. The key idea of our argument is that dualization does not add or remove combinatorial information: a planar graph corresponding to an arbitrary 2-manifold mesh with no boundaries is uniquely determined by its dual.

Let us observe at first that if G is $3-$*connected* then its dual G^* is also $3-$*connected*. Furthermore, if G is simple (no multiple edges and self-loops), then G^* is a simple graphe. This imply that we can compute a canonical triangulation of G^* too.

Let us consider Σ and Σ^*, respectively the two sequence of face degrees and vertex valences associated to the canonical triangulations T_G, T_{G^*}.

Using an arithmetic encoder to compress separately Σ and Σ^* we can achieve asymptotically the theoretical upper bound given by the entropy of the valence/degree sequences. If we denote by p_i and q_j the relative frequencies of vertex valence i and face degree j in the two sequences Σ and Σ^*, their entropy are respectively expressed by:

$$\mathcal{E} = \sum_{j=3}^{\infty} q_j \log_2 \frac{1}{q_j} \, [\text{b/f}] \;\; \text{and} \;\; \mathcal{E}^* = \sum_{i=3}^{\infty} p_i \log_2 \frac{1}{p_i} \, [\text{b/v}]$$

We can now imagine to apply our triangulating algorithm to both G and G^*, and to use one more bit to chose between T_G and T_{G^*}, depending on the code sizes of Σ, Σ^* (corresponding asymptotically to $f \cdot \mathcal{E}$ and $n \cdot \mathcal{E}^*$).

If $f \cdot \mathcal{E} > n \cdot \mathcal{E}^*$ we compute T_{G^*} and we return Σ^*, with 0 as extra bit; otherwise we apply our algorithm to G and we return an 1 as extra bit. During the detriangulation phase is suffices to read the extra bit and eventually to compute the dual of the graph reconstructed (if we read 0 as extra bit).

Now it remains only to apply the analysis by Khodakovsky et al. [13], concerning the maximum possible entropy for the two sequences Σ, Σ^* (encoded independently): solving a constrained maximization problem they discover an asymptotic upper bound for the sum of the two entropies, for a worst case distribution of the p_i and q_j.

Let us introduce the parameter $r = f/n \in [0.5, 2]$, representing the ratio between the number of faces and vertices in a 3-connected planar graph. As shown by Khodakovski et al., it is possible to express the total bit rate of the two sequences Σ,Σ^* in term of b/e:

$$\mathcal{E}_{tot}(r) = \frac{1}{r+1}\mathcal{E} \; + \; \frac{r}{r+1}\mathcal{E}^*$$

The function $\mathcal{E}_{tot}(r)$ takes its maximum in $[0.5, 2]$ for $r = 1$, precisely $\mathcal{E}_{tot}(1) = 2$ b/e. Finally, choosing between T_G and T_{G^*}, allows to pay at most:

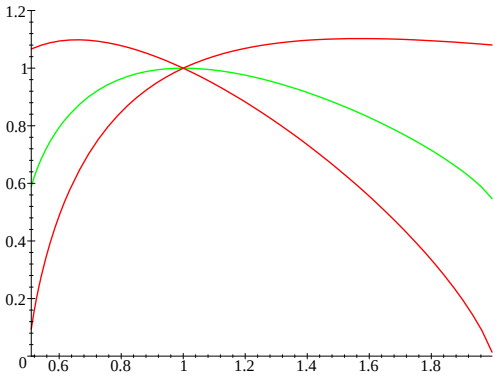

Figure 7: Plot of the function $\frac{f\cdot\mathcal{E}+n\cdot\mathcal{E}^*}{2}$ (green curve), corresponding to a worst case distribution of the $\{p_i\}$,$\{q_j\}$. Red curves represent $f\cdot\mathcal{E}, n\cdot\mathcal{E}^*$. For $0.5 \leq r \leq 1$ is more convenient coding G; for $1 < r \leq 2$ we code G^*.

$$cost(G) \;=\; \min\{f\cdot\mathcal{E}, n\cdot\mathcal{E}^*\} \;\leq\; \frac{f\cdot\mathcal{E}+n\cdot\mathcal{E}^*}{2} \;=$$
$$\frac{e+2}{2}\left(\frac{1}{r+1}\mathcal{E} \;+\; \frac{r}{r+1}\mathcal{E}^*\right) \leq e \;+\; o(e) \text{ bits}$$

□

In figure 7 we plot the two functions $\frac{1}{r+1}\mathcal{E}$ and $\frac{r}{r+1}\mathcal{E}^*$ representing the entropy of the two sequences Σ,Σ^* coded independently.

Comparison with a trivial method

The reader now could ask about the efficiency of our algorithm, face degree based. We could compare our strategy, for example, to the trivial method consisting in coding every red edge in T_G with a '0', and blue edges with a '1' (needing one binary symbol for every edge of the triangulation T_G). Following an argument similar to the one presented in [13], we are able to show that our method achieves always better bit rates compression.

Let us compare our worst bit rates compression with the performance of the trivial method, that produces a string $\Sigma_{01} \in \{0,1\}^*$ of length e': where e' is the number of edges in the computed canonical triangulation; in general $e' > e$.

At first we have to estimate the entropy $\mathcal{E}(\Sigma_{01}) = p_0 \log(\frac{1}{p_0}) + p_1 \log(\frac{1}{p_1})$ of the string Σ_{01} (here, p_0, p_1 are the relative frequencies of symbols $0, 1$ in Σ_{01}).

Since the number of total edges in the triangulation T_G is $e' = 3n$ and it must satisfy $e' = n + f - 2$, the above relative frequencies and the corresponding entropy, for large n are given by:

$$p_0 = \frac{n + nr - 2}{3n} \approx \frac{1 + r}{3} \quad \text{and} \quad p_1 = 1 - p_0 \approx \frac{2 - r}{3}$$

$$\mathcal{E}(\Sigma_{01}) = \frac{1 + r}{3} \log(\frac{3}{1 + r}) + \frac{2 - r}{3} \log(\frac{3}{2 - r})$$

In the same way we estimate the entropy of $\mathcal{E}(\Sigma^*_{01})$, the string representing the edges coloring of the dual G^*.

It remains only to express the above quantities in term of the number of edges in the initial graph G (resp. G^*); this number depends on the parameter r. Finally the bit rates compression achieved by the trivial detriangulating algorithm for G (resp. G^*) are:

$$\mathcal{E}(\Sigma_{01})(r) \cdot \frac{3n}{n(r + 1)} \quad \text{and} \quad \mathcal{E}(\Sigma^*_{01})(r) \cdot \frac{3f}{\frac{f}{r}(r + 1)}$$

Figure 8 shows the plot of these curves, compared with the function $\min\{f \cdot \mathcal{E}, n \cdot \mathcal{E}^*\}$ characterizing the bit rates compression of our detriangulating algorithm (worst case).

In both cases (coding of G or G^*) our bit rates are smaller than ones achieved by the trivial method, for $0.5 < r < 2$. As one could expect the results coincide when G (resp. G^*) is a triangulation, for $r = 0.5$ (resp. $r = 2$).

7.2 Asymptotic upper bound for polygonal mesh encoding

Let us observe that we can apply any coder for triangulations for representing the canonical triangulation T_G: in particular, choosing the the recent optimal coder by Poulalhon and Schaeffer [16], we are able to provide an asymptotic bit rate upper bound for the polygonal mesh encoding described in the previous section. Theorem 2 and some ideas by Chuang and al. [3], allow us to state the following

Corollary 1. *Let G be a 3-connected simple planar graph with e edges (f faces and n vertices). Our linear time encoding of G based on its canonical triangulation takes at most asymptotically $2.62e + o(e)$ bits.*

Proof. The first phase our algorithm consist of computing a canonical triangulation of the graph: if we suppose to apply an optimal coder for triangulation, T_G can be encoded always with 3.2451 b/v. This amount of bits is fixed, and it does not depend on the particular way we triangulate G (we only add *red* edges, no new vertex is needed).

Let us suppose to write the total amount of information needed to represented G and G^* separately, consisting of the combinatorial information of their canonical triangulations

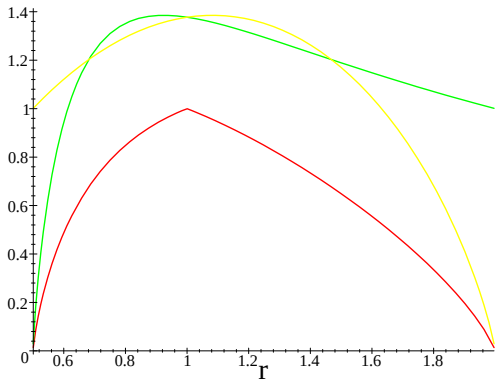

Figure 8: Gain obtained with our strategy. We plot over $[0.5, 2]$ the functions representing bit rates compression of the trivial algorithm and of ours (red curve). The function $\min\{f \cdot \mathcal{E}, n \cdot \mathcal{E}^*\}$ is always below $\mathcal{E}(\Sigma_{01})(r) \cdot \frac{1}{n(r+1)}$ and $\mathcal{E}(\Sigma^*_{01})(r) \cdot \frac{3f}{\frac{f}{r}(r+1)}$. Bit rates compression are expressed in b/e unit.

and of the two sequences Σ, Σ^*; let l_G (resp. l_{G^*}) be the size of the code for G (resp. for G^*)

$$l(G) + l(G^*) = \ (3.24n \ + \ f \cdot \mathcal{E}) \ + \ (3.24f \ + \ n \cdot \mathcal{E}^*)$$

We now rewrite this expression in term of b/e using the Euler's formula, which states $n - e + f = 2$, for the class of meshes that are manifold, with one connected component, no boundary and of genus 0. Following the similar argument described in theorem 2 and using the result by Khodakovsky et al. [13] we have:

$$\text{total bits} = \ \min\{l_G, l_{G^*}\} + 1 \ \leq \ \frac{l_G + l_{G^*}}{2} + 1 =$$
$$= \frac{3.24(e+2)}{2} + \frac{f \cdot \mathcal{E} + n \cdot \mathcal{E}^*}{2} + 1 \ \leq \ 2.62e + o(e) \text{ bits}$$

(again we have to use a bit to choose between encoding G and G^*) □

8 Experimental results

We have written a C++ implementation (based on the geometric library CGAL) of our encoding strategy and of the others cited algorithms concerning canonical orderings and realizers. The experimental results obtained running our algorithm on some real-world models confirm our theoretical analysis. Concerning the time complexity, our algorithm is quite simple to implement and takes less than 0.1 second on the tested meshes, running on a 2.4GHz, 512MB PC.

8.1 Adaptive behavior and asymptotic upper bound

In Table 1 we show some bit rates compression obtained using an optimal algorithm for the triangulation T_G. The choice of an optimal algorithm for the first encoding phase, allows us to guaranteed an asymptotic upper bound. We use a 0-order arithmetic encoder for the sequence Σ_G: this does not benefit of the constraints concerning the maximum size of a reconstructed face. Even if our codes are in average slightly longer than those of other strategies ([13], [9]), the results in the Table 1 show the adaptive characteristic of our algorithm, that achieves good bit rates compression in the case of regular meshes.

3D model	V	F	E	G	G^*
Triceratops	2832	2834	5664	2.14	**1.65**
Cupie	2968	3032	5988	**2.07**	2.15
Lion head	2109	2312	4419	**2.03**	2.16
Shark	2560	2562	5120	1.98	**1.62**
Al	3616	3442	7016	**2.08**	2.15
Beethoven	2634	2816	5438	**2.17**	2.32
David	2646	2563	5207	2.14	**2.10**
Ear	3747	3701	7446	**1.72**	1.92

Table 1: Experimental results obtained with our algorithm on some common 3D polygonal meshes. Last columns show the bit rates compression expressed in bit/edge, for the original graph G and its dual G^*. The canonical triangulation T_G has been encoded using the optimal algorithm by Poulalhon and Schaeffer.

Since the first phase of our encoding algorithm (the construction of the T_G), does not depend on the way we code the triangulation T_G, we can choose to use some other popular and efficient triangle mesh compressor. Better results concerning bit rates compression showed above can be obtained combining our detriangulation strategy, for example, with the Touma-Gotsman coder or its improvement by described Alliez and Desbrun [1]. While we cannot guaranteed an upper bound on the length of our code, the adaptive behavior of valence-based method allows to achieve in practice good bit rates for regular meshes, close to the performance of other polygonal mesh coders.

3D model	G b/e	G^* b/	T_G b/e	T_{G^*} b/e
Triceratops	0.02	0.51	0.90	1.14
Lion head	0.45	0.46	0.95	1.31
Beethoven	0.63	0.50	1.06	1.31
David	0.49	0.41	1.15	1.04
Ear	0.31	0.005	0.88	0.73

Table 2: The entropy of the vertex valences sequence does change after triangulating G or G^*. Results are expressed in bit/edge unit.

3D model	E	our+TG	our+AD	IS
Triceratops	5664	1.36	1.22	0.59
Lion head	4419	1.61	1.45	0.87
Beethoven	5438	1.61	1.38	0.99
David	5207	1.97	1.69	0.78
Ear	7446	1.05	0.97	0.29

Table 3: Better results can be obtained combining our triangulating strategy with valence-based triangle mesh coders. Last column shows the comparison with the Polygon Mesh Benchmark Compressor by Isenburg.

Recall that during the first phase, the construction of T_G, our algorithm changes the vertex valences, by adding some *red edges*. This implies that our final bit rates compression will depend on the new distribution of the vertex valences. In Table 2 we show the entropy of the vertex valences sequence, before and after triangulating the original graph G: observe that in general the dispersion of the p_i changes, increasing the new entropy. Finally in Table 3 we compare our new results obtained adopting the TG coder (and its improvement given by Alliez and Desbrun), to the bit rates compression of another valence-based polygonal mesh coder [9].

9 Conclusion and future work

We have presented a method for the compression of a polygonal mesh that exploits the low dispersion of the relative face degree distribution, and for which we can guaranteed an asymptotic bit rate upper bound, improving some previous theoretical results. The experimental results, which confirm our theoretical analysis based on the argument by [13], show good bit rates compression, often lower than the Tutte entropy for polygonal graphs, in particular in the case of regular meshes. We think that further improvements concerning our

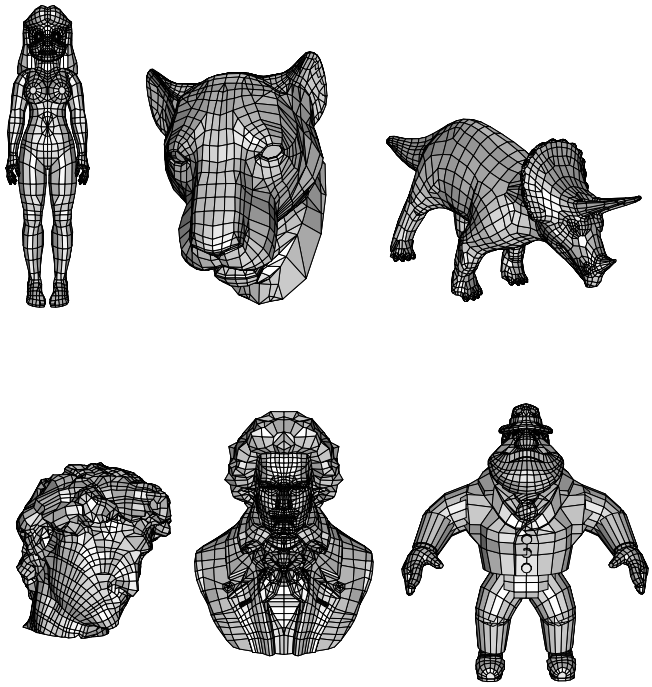

Figure 9: Some of the 3D meshes tested with our algorithm.

Algorithm	planar	triangulation	3-conn.
Turan ('84)	$4e$		
Keeler ('95)	$3.58e$	$1.53e$	$3e$
Chuang ('98)		$\frac{4}{3}e$	$2.37e$
He ('99)		$\frac{4}{3}e$	$2.83e$
Bonichon ('03)	$2.90e$	$3.37n$, $1.12e$	$2.90e$
PS ('03)		$3.24n$, $1.08e$	
our + PS		$3.24n$, $1.08e$	**2.62e**

Table 4: Comparison of the asymptotic upper bounds of some well known algorithms for graph encoding.

bit rate compression are possible, since in our analysis we did not consider the constraints given by the *interior degree* of a vertex: this could be exploited with an arithmetic encoding and a prediction of the maximum range of the scanned symbol in Σ_G.

References

[1] P. Alliez and M. Desbrun. Valence-driven connectivity encoding of 3d meshes. In *Proceedings of Eurographics*, pages 480–489, 2001.

[2] N. Bonichon, C. Gavoille, and N. Hanusse. An information-theoretic upper bound of planar graphs using triangulation. In *STACS*, 2003.

[3] R.C.-N Chuang, A. Garg, X. He, M.-Y. Kao, and H.-I. Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. *Automata, Laguages and Programming*, pages 118–129, 1998.

[4] H. De Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[5] P. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. on Graphics*, 21:372–379, 2002.

[6] C. Gotsman. On the optimality of valence-based connectivity coding. *Computer Graphics Forum*, 22:99–102, 2003.

[7] S. Gumhold and W. Strasser. Real time compression of triangle mesh connectivity. In *SIGGRAPH 98 Conference Proceedings*, pages 133–140, 1998.

[8] X. He, M.-Y. Kao, and H.-I Lu. Linear-time succint encodings of planar graphs via canonical orderings. *SIAM J. on Discrete Mathematics*, 12:317–325, 1999.

[9] M. Isenburg. Compressing polygon mesh connectivity with degree duality prediction. In *Proceeding of Graphics Interface*, 2002.

[10] M. Isenburg and J. Snoeyink. Face fixer: Compressing polygon meshes with properties. In *ACM SIGGRAPH Conference Proceedings*, pages 263,270, 2000.

[11] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.

[12] K. Keeler and J. Westbrook. Short encondings of planar graph and maps. *Discrete Appl. Math.*, 58:239–252, 1995.

[13] A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schroder. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Journal of the Graphical Models*, 64:3–4, 2002.

[14] D. King and J. Rossignac. Guaranteed 3.67v bit encoding of planar triangle graphs. In *11th Canad. Conf. on Computational Geometry*, pages 146–149, 1999.

[15] B. Kronrod and C. Gotsman. Efficient coding of non-triangular meshes. In *Pacific Graphics*, 2000.

[16] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. submitted to ICALP, 2003.

[17] J. Rossignac. Edgebreaker: connectivity compression for triangle meshes. In *IEEE Trans. on Visualization and Computer Graphics*, pages 47–61, 1999.

[18] A. Szymczak, D. King, and J. Rossignac. An edgebreaker efficient compression scheme for regular meshes. In *12th Canadian Conference on Computational Geometry*, pages 257–265, 2000.

[19] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17:84–115, 1998.

[20] C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface 98*, pages 26–34, 1998.

[21] G. Turan. On the succint representation of graphs. *Discrete Applied Mathematics*, 8:289–294, 1984.

[22] W. T. Tutte. A census of planar triangulations. *Canad. J. Math.*, 14:21–38, 1962.

[23] W. T. Tutte. A census of planar maps. *Canad. J. Math.*, 15:249–271, 1963.

[24] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Comm. ACM*, 30:520–540, 1987.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399