Compression interactive de maillages triangulaires arbitraires

Olivier Devillers, Pierre-Marie Gandoin

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Compression interactive de maillages triangulaires arbitraires

Olivier Devillers — Pierre-Marie Gandoin

N° 4158

Avril 2001

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4158--FR

Compression interactive de maillages triangulaires arbitraires

Olivier Devillers , Pierre-Marie Gandoin

Thème 2 — Génie logiciel
et calcul symbolique
Projet Prisme

Rapport de recherche n° 4158 — Avril 2001 — 25 pages

Résumé : En quelques années, les maillages ont conquis une position prédominante parmi les différents modes de représentation informatique d'objets géométriques. Plus particulièrement, les maillages à base de simplexes — les triangles pour la représentation de surfaces plongées en 3D, les tétraèdres pour la représentation de volumes — semblent être actuellement les plus répandus. Le développement rapide des applications manipulant ces structures géométriques dans des domaines aussi divers que le calcul par éléments finis ou les jeux vidéo a très vite soulevé le problème d'un codage efficace et adapté à la visualisation. L'expansion du *World Wide Web*, qui nécessite une représentation compacte et progressive des données pour garantir la convivialité de l'interface homme/machine, a conféré à ce problème une place centrale dans la recherche informatique. Ainsi, depuis 1995, de nombreux algorithmes ont été proposés pour la compression de maillages triangulaires. Dans ce rapport, nous décrivons une méthode de codage progressive, sans perte d'information, adaptée à des maillages triangulaires arbitraires (non nécessairement manifold, de genre quelconque), et généralisable à n'importe quelle dimension. Pour le cas particulier de maillages triangulaires surfaciques, des taux moyens autour de $3,6$ bits par sommet sont atteints sur des modèles usuels pour le codage de la connectivité.

Mots-clés : géométrie, compression, codage, triangulation, maillage, connectivité, topologie, progressivité, interactivité

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Interactive Compression of Arbitrary Triangular Meshes

Abstract: Since a few years, meshes have conquered a predominant status amongst the different type of computer data structure for geometric objects. More precisely, the meshes composed of simplices — triangles for the modelisation of surfaces embedded in 3D, tetrahedra for the modelisation of volumes — seem to be the most widespread at the present time. The rapid growing of applications using these geometric structures in such various fields as finite elements computing or video games has quickly raised the problem of an efficient coding, well suited for storage or visualisation. The boom of the *World Wide Web*, which needs a compact and progressive representation of the data in order to guarantee the user-friendliness of the exchanges and communications. Therefore, since 1995, a number of algorithms have been proposed for the compression of triangular meshes. In this report, we describe a coding method which is progressive, lossless, adapted to arbitrary triangular meshes (any gender, non necessarily manifold), and generalizable to any dimension. For the particular case of surface meshes, about 3.6 bits per vertex are necessary on average to code the connectivity of usual models.

Key-words: geometry, compression, coding, triangulation, mesh, connectivity, progressivity, interactivity

1 Introduction

1.1 Motivations

Ce rapport est consacrée au codage progressif sans perte de maillages surfaciques triangulaires. Nous verrons, dans la Section 1.3, que cette structure géométrique est celle qui a donné lieu au plus grand nombre de travaux dans le domaine de la compression. D'une manière générale, les maillages triangulaires sont au centre des recherches actuelles dans tous les domaines de l'informatique manipulant des surfaces dans l'espace tridimensionnel. Parmi les diverses applications des maillages triangulaires, on peut citer la conception assistée par ordinateur, la modélisation (en médecine, en géologie ou en cartographie par exemple), la simulation et le calcul par éléments finis, ou encore les multiples applications de la réalité virtuelle (visites virtuelles dans les sites culturels ou commerciaux, jeux vidéo en $3D$, réalité augmentée, etc.).

La prédominance du triangle peut s'expliquer notamment par les remarques suivantes:

- le triangle constitue la primitive géométrique élémentaire: tout polygone peut se ramener à un ensemble de triangles par triangulation, et par suite, un maillage polygonal arbitraire peut être réduit à un maillage triangulaire sans changer de géométrie ni de classe topologique,
- certaines de ses propriétés intrinsèques sont utilisées de manière extensive par les algorithmes de rendu ou de calcul scientifique: coplanarité, interpolation linéaire directe entre les trois sommets, contrôle de la surface, etc.,
- à l'instar du pixel pour les images bitmap, le triangle est l'unité d'affichage standard des cartes accéleratrices graphiques actuelles dédiées à la gestion de scènes tridimensionnelles: quel que soit son mode de représentation originel, le maillage est triangulé au moment de l'affichage. Ainsi, il est parfois préférable de fixer a priori une manière de trianguler le modèle (par exemple, selon un critère de proximité à la surface modélisée) plutôt que de laisser le contrôle à l'afficheur.

1.2 Notions et définitions elémentaires

1.2.1 Géométrie, topologie et attributs

Il nous faut tout d'abord définir ce que nous regroupons dans ce rapport sous la dénomination de structure géométrique. Le noyau d'une structure géométrique est un ensemble P de n points en dimension d. Dans la suite, on parlera de la *géométrie*

de l'objet, ou encore de ses *positions* pour faire référence à cet ensemble P où chaque point est repéré par ses d coordonnées.

Une structure géométrique peut se limiter à sa géométrie, mais souvent, elle contient un deuxième type d'information qui consiste en un graphe A constitué des relations d'incidence entre les éléments de P (les arêtes, ou simplexes de dimension 1). Cet ensemble A constitue la *topologie* (ou *connectivité*) de la structure géométrique.

Bien que l'ensemble A contienne généralement l'information suffisante pour reconstruire les éventuels simplexes de degré supérieur à 1 (triangles, tétraèdres, etc.) ou les polyèdres présents dans l'objet (par recherche de cycle dans un graphe), l'information topologique est souvent complétée en pratique par la description explicite de ces éléments. Ainsi, un simplexe de degré 2 (triangle) sera représenté par un triplet de points de P. Enfin, dans certains cas, la topologie de l'objet pourra être enrichie des relations d'adjacence entre les différents simplexes et polytopes.

Enfin, une structure géométrique peut contenir un troisième type de données, que l'on appelle ses *attributs*, ou *propriétés*. Il s'agit d'informations diverses généralement associées aux points de P: couleurs, normales, matériaux, textures, etc.

1.2.2 Maillages surfaciques

Nous proposons dans cette section quelques définitions élémentaires sur les maillages surfaciques.

Définition 1 *Une structure géométrique constitué d'un graphe topologique G de sommets, d'arêtes et de faces, et d'un ensemble P de coordonnées de sommets décrivant le plongement de G dans l'espace est appelée un maillage.*

Définition 2 *Un maillage M est dit* manifold *si toute arête de G est incidente à deux faces exactement, et les faces incidentes à tout sommet forment un cône simple. Autrement dit, tout sommet de M a un voisinage homéomorphe à un disque.*

Définition 3 *Un maillage est dit* manifold avec bords *s'il vérifie les propriétés d'un maillage* manifold *partout sauf pour un ou plusieurs cycles d'arêtes incidentes à une seule face chacune. De telles arêtes sont les arêtes de bord et les cycles d'arêtes de bord sont appelés trous. Les sommets de bord (i.e. les extrémités des arêtes de bord) ont un voisinage homéomorphe à un demi-disque.*

Définition 4 *Un maillage M est de genre n s'il contient n trous.*

Définition 5 *Un maillage est dit* simple *s'il est* manifold *et sans bord (i.e. de genre nul), ce qui est topologiquement équivalent à une sphère.*

Définition 6 *Soit s un sommet d'un maillage M. Le degré de s est le nombre de voisins de s dans M, ou encore le nombre d'arêtes incidentes à s dans M.*

Un graphe de s sommets, a arêtes et f faces plongé sur une sphère vérifie la relation d'Euler:

$$v - e + f = 2$$

1.2.3 Précision numérique et quantification

Dans la plupart des structures géométriques rencontrées en pratique, le format original — sous forme binaire (format *brut*) ou sous forme ASCII (format *VRML* par exemple) — code la position de chaque sommet en dimension d par un ensemble de d nombres flottants représentant ses coordonnées cartésiennes. Ces nombres flottants correspondent à une précision de 24 bits en notation normalisée [17, 1] pour chaque coordonnée. À notre connaissance, aucune des méthodes de compression géométrique publiées jusqu'à maintenant ne manipule les coordonnées des sommets sous cette forme. Dans un premier temps, les coordonnées sont converties en nombres entiers codés sur k bits, c'est-à-dire compris entre 0 et $2^k - 1$. Cette conversion peut s'opérer sans perte d'information à condition de choisir le nombre k suffisamment grand pour contenir la précision des nombres flottants originaux (en pratique, 32 bits s'avèrent généralement suffisants). Cependant, les méthodes décrites dans la Section 1.3 effectuent cette conversion préalable avec des valeurs de k comprises entre 8 et 12, réalisant ainsi une *quantification* qui s'accompagne d'une perte d'information considérable. Ces méthodes ne sont donc *sans perte* qu'au niveau du codage de la connectivité du modèle. L'argument communément évoqué par les auteurs pour justifier cette quantification radicale est que la précision du modèle — même lorsque celui-ci représente une forme polyèdrale par nature — est souvent limitée par les erreurs numériques durant le processus de création. S'il est vrai que la résolution des scanners ou les erreurs d'arrondi dans les calculs d'intersection doivent être prises en compte, on peut légitimement penser que la précision sur l'objet modélisé dépasse largement les 12 bits par coordonnée. De plus, si des applications fondées sur la visualisation peuvent tolérer certaines approximations sur les positions des sommets, d'autres, en particulier dans des domaines du calcul scientifique comme l'analyse par éléments finis, réclament généralement une meilleure précision.

Le problème de la perte d'information par quantification se présente également pour le codage des attributs du modèle. Là encore, les nombres flottants codant sur 64 bits les propriétés de couleurs, normales ou matériaux, sont convertis en nombres

entiers. Cette conversion s'accompagne d'une perte de résolution dans la plupart des méthodes de compression géométrique exposées dans la Section 1.3.

1.2.4 Codage arithmétique

La méthode classique de codage entropique que nous avons choisi d'utiliser ici est le codage arithmétique. Développé dans les années 80 [33, 40], ce principe de compression permet de coder un symbole en fonction de sa probabilité d'occurrence, sur un nombre de bits non nécessairement entier, ce qui constitue un avantage décisif sur les célèbres codes de Huffman. Fondamentalement, le principe de la compression arithmétique est de coder une séquence de symboles par un unique nombre réel appartenant à l'intervalle [0,1[. Pour chaque symbole rencontré, l'intervalle initial [0,1[est raffiné en fonction de la probabilité estimée du symbole, conduisant finalement à un petit intervalle dont n'importe quel nombre code l'ensemble de la séquence. Cette méthode permet de coder chaque symbole s de la séquence sur $\log_2 \frac{1}{P} + \epsilon$ bits, où P est la probabilité estimée de s, et ϵ une quantité négligeable devant $\log_2 \frac{1}{P}$. Ainsi cette technique peut être très performante si elle est couplée à une modélisation statistique efficace des données à coder. C'est l'utilisation du codage arithmétique qui nous permet, dans la suite de ce rapport, de nous affranchir des parties entières supérieures appliquées aux logarithmes dans les analyses de coût théorique. Pour alléger les notations, nous avons choisi de ne pas faire figurer non plus les ϵ associés à ces mêmes logarithmes.

1.2.5 Taux de compression

Il existe dans la littérature de nombreux moyens d'évaluer l'efficacité d'un algorithme de compression. Si V_I désigne le volume des données initiales et V_c le volume des données compressées, on peut choisir d'utiliser un facteur de compression, qui exprime combien de fois les données compressées sont plus petites que les données originales ($F = V_I/V_C$), ou de mettre en évidence le gain obtenu à l'issue de la compression de manière absolue ($G_A = V_I - V_C$) ou relativement aux données initiales ($G_R = G_A/V_I$).

Dans la suite de ce rapport, nous utiliserons le taux de compression

$$T = \frac{V_C}{V_I} 100$$

qui exprime le volume des données compressées en pourcentage du volume des données initiales. Un algorithme sera donc d'autant plus efficace que le taux de compression associé est bas.

D'autre part, pour le cas particulier de la compression géométrique, il est souvent plus lisible d'exprimer le volume des données nécessaire au codage d'une structure géométrique en fonction d'un nombre caractérisant cette structure. Ainsi, la taille d'un objet géométrique ne sera pas exprimée en bits ou en octets mais en bits par sommet (ou *bps*), en bits par arête (*bpa*), ou en bits par face (*bpf*). Cette diversité des notations rend plus difficile les comparaisons entre les différents travaux issus de la communauté scientifique. Cependant, pour le cas particulier des maillages triangulaire, la relation d'Euler permet de passer facilement d'une unité à l'autre: 1 bps $\approx$ 1/2 bpf $\approx$ 1/3 bpa. Dans ce rapport, nous exprimerons la taille d'un objet géométrique en bits par sommet.

1.3 Précédents travaux

Comme nous l'avons expliqué dans la Section 1.2.1, une structure géométrique est essentiellement définie par sa connectivité et ses positions. La principale difficulté de la compression géométrique est de parvenir à coupler un codage efficace de la connectivité avec une représentation compacte des positions. À notre connaissance, toutes les méthodes de compression publiées jusqu'à présent donnent la priorité à la compression topologique. Leur idée intuitive commune est de décrire un arbre couvrant des sommets et un arbre couvrant des faces (deux structures duales l'une de l'autre), en énumérant les sommets suivant une stratégie déterministe. Cette stratégie construit une séquence où chaque sommet (défini par ses coordonnées) est accompagné d'un code explicitant la manière dont il est connecté aux sommets précédemment décrits. La topologie du maillage est donc contenue dans ces codes d'assemblage, mais aussi dans l'ordre des sommets dans la séquence. C'est à partir de cet ordre imposé par la connectivité que la partie géométrique du codeur doit travailler. Le codeur géométrique utilise généralement le codage différentiel et la prédiction de position. En effet, s'il est vrai que l'ordre des sommets dans la séquence codante n'est pas optimisé pour la compression des positions, il favorise néanmoins la proximité géométrique de deux sommets successifs. Ainsi, au lieu d'exprimer la position du sommet courant de manière absolue, on l'exprime relativement au sommet précédent, par un vecteur de déplacement (ou vecteur de différence). Il est également possible de prédire la position du sommet courant à partir des positions des n sommets précédents dans la séquence (en pratique, il s'agit souvent de prédiction linéaire, avec $n = 2$ ou $n = 3$), et de coder uniquement le résidu (i.e. l'erreur entre la position prédite et la position réelle) par codage entropique.

La plupart des méthodes de compression de maillages triangulaires en simple résolution exploitent ce schéma général [9, 13, 38, 18, 39, 34, 35, 23, 22, 21, 26,

5, 7]. Elles permettent d'atteindre, pour les plus efficaces d'entre elles, des taux de compression moyens de l'ordre de 3 bits par sommet. Dans la mesure où l'objet de ce rapport est la compression *progressive*, nous ne détaillerons pas ici le fonctionnement de ces algorithmes.

La transmission progressive repose sur la création d'une séquence de raffinements de la topologie et de la géométrie du maillage qui permet au décodeur de tronquer le flux de donnée à tout moment pour créer la meilleure approximation possible du modèle original. La difficulté est que l'organisation hiérarchique de la structure géométrique permettant cette progressivité (ou *échelonnabilité*) tend à augmenter sensiblement la taille de la structure. C'est pourquoi les premiers algorithmes progressifs n'étaient pas conçus pour la compression et utilisaient un nombre important de bits par sommet.

Par la suite, plusieurs méthodes de compression en simple résolution ont pu être étendues à la compression progressive. Ainsi, Taubin et al. ont proposé une extension progressive [37] aux travaux originaux de Taubin et Rossignac [38], et Bajaj et al. ont adapté leur algorithme de décomposition en couches [7] à la transmission progressive [6]. Les techniques les plus récentes, en regroupant les opérations de raffinement en paquets, permettent d'obtenir des taux de compression proches de ceux des techniques non progressives (la méthode de Pajarola et Rossignac [29, 30] atteint moins de 8 bits par sommet en moyenne).

Les travaux de Cohen-Or et al. [8] combinent des techniques de simplification séquentielle par suppression de sommet pour la connectivité (cf Section 3.1), avec la prédiction des positions pour la géométrie. Une technique efficace de coloration indiquant au décodeur les *patchs* (trous polygonaux retriangulés) où les sommets doivent être insérés, permet d'atteindre les meilleurs taux de compression actuels pour le codage progressif de la connectivité (environ 6 bits par sommet en moyenne).

La méthode décrite par Li et Kuo [27] se distingue des précédentes en introduisant de la progressivité dans la transmission de la géométrie du modèle. Ainsi, la séquence codante raffine simultanément la connectivité, en insérant des sommets dans des faces, et les positions, en augmentant la précision sur les coordonnées des sommets. Ce problème du compromis optimal entre le nombre de sommets d'un maillage et la précision sur les positions de ces sommets a été étudié en détail par King et Rossignac [24].

Alliez et Laurent [4, 3] proposent une méthode hybride qui compresse la topologie de l'objet en simple résolution, mais offre un codage progressif des positions. Pour la topologie, la technique est fondée sur l'exploitation des degrés des sommets utilisée par Touma et Gotsman [39] et améliorée par une heuristique sur l'ordre de conquête

des sommets conduisant à un codage extrêmement compact sur des maillages réguliers. En ce qui concerne la géométrie, les auteurs raffinent les techniques de prédiction habituellement utilisées en compression géométrique et adaptent efficacement le codage par plan de bits à des nombres flottants pour obtenir l'échelonnabilité.

Remarque 1 *Les méthodes décrites dans cette section sont idéalement conçues pour des maillages simples (cf Section 1.2.2). Les auteurs proposent généralement des extensions qui permettent de coder les maillages avec bords, voire les maillages non-manifold (par exemple en dupliquant certains sommets), mais dans ces cas, les performances des codeurs sont considérablement dégradées. En outre, les résultats sont fortement conditionnés par la régularité du maillage. Les meilleurs taux de compression sont atteints pour des maillages très réguliers, qui optimisent la prédiction géométrique et favorisent le codage entropique de la description topologique.*

Remarque 2 *Dans ce survol des différentes méthodes de compression précédemment publiées, nous donnons très peu de résultats concernant la compression des positions. En effet, la comparaison est rendue très difficile dans la mesure où il s'agit d'une compression avec perte, du fait de la quantification préalable des données géométriques (cf Section 1.2.3). Cette quantification des positions se faisant à des résolutions différentes (et donc, entraînant des degrés de perte variables), un nombre de bits par sommet pour le codage de la géométrie du modèle n'a pas de sens dans l'absolu. Pour la comparaison détaillée des méthodes les plus efficaces avec notre algorithme en terme de compression des positions, nous renvoyons le lecteur à la Section 4.2.*

1.4 Cadre

Ce rapport fait suite aux précédents travaux publiés par Devillers et Gandoin sur la compression géométrique. La méthode présentée ici reprend le principe général du codeur géométrique développé dans *Compression géométrique pour une transmission progressive* [10] (nous rappelons brièvement son fonctionnement dans la Section 2) et bénéficie par conséquent de toutes ses caractéristiques. Un nouveau codeur topologique est construit sur le principe de subdivision de cellules, moins général que celui proposé dans *Geometric Compression for Interactive Transmission* [11], mais plus efficace dans le cas particulier des maillages simpliciaux.

2 Rappel sur le codeur géometrique

Dans le but de simplifier la description de l'algorithme, nous commençons par traiter le cas de la dimension 1. Nous verrons ensuite que la généralisation à la dimension d est directe.

Nous décrivons d'abord la partie codage de l'algorithme. Soit S un ensemble de n points situés sur un segment de droite, entre 0 et 2^b (les coordonnées des points sont donc codées sur b bits). L'algorithme commence par coder le nombre total de points sur un nombre de bits arbitrairement fixé (par exemple 32). Puis il entre dans la boucle principale qui consiste à subdiviser le segment courant en deux demi-segments et à coder le nombre de points contenus dans l'un d'entre eux (celui de gauche par exemple) sur un nombre de bits optimal: si le segment courant contient p points, le nombre de points dans le demi-segment sera codé sur $\log_2(p+1)$ bits.

L'algorithme maintient donc une liste de segments constitués de:

- la longueur du segment,
- la position du segment,
- une liste des points situés sur le segment.

Chaque segment est retiré de la liste, subdivisé en 2 demi-segments insérés en fin de liste s'ils sont non vides, et donne lieu à un code en sortie correspondant au nombre de points contenus dans le demi-segment gauche. L'algorithme termine lorsqu'il n'y a plus de segments divisibles dans la liste courante, c'est-à-dire plus de segments de longueur supérieure à 1. Le pseudo-code ci-dessous détaille le fonctionnement de la partie codage.

Algorithme *Codage de points sur un segment de droite*

1. $\mathcal{L} \leftarrow$segment de droite original $\mathcal{S}_0$
2. écrire le nombre de points situés sur $\mathcal{S}_0$ sur 32 bits
3. **tant que** $\mathcal{L}$ non vide
4. **faire**
5. $\mathcal{S} \leftarrow$extraire le premier segment de $\mathcal{L}$
6. $n \leftarrow$nombre de points situés sur $\mathcal{S}$
7. $\mathcal{S}_1 \leftarrow$moitié gauche de $\mathcal{S}$
8. $n_1 \leftarrow$nombre de points situés sur $\mathcal{S}_1$
9. $\mathcal{S}_2 \leftarrow$moitié droite de $\mathcal{S}$
10. $n_2 \leftarrow$nombre de points situés sur $\mathcal{S}_2$
11. écrire n_1 sur $\log_2(n+1)$ bits
12. **si** $n_1 > 0$
13. **alors** ajouter $\mathcal{S}_1$ à la fin de $\mathcal{L}$
14. **si** $n_2 > 0$
15. **alors** ajouter $\mathcal{S}_2$ à la fin de $\mathcal{L}$

Ainsi, les seuls codes écrits par l'algorithme sont les nombres de points situés sur les segments successifs. Les positions de ces points sont cachées dans l'ordre des codes. En fait, cet ordre contient implicitement une structure d'arbre binaire.

La partie décodage de l'algorithme répond exactement à sa partie codage. Une liste de segments est maintenue, mais cette fois un segment de droite est constitué de:

- la longueur du segment,
- la position du segment,
- le nombre de points situés sur le segment.

Pour chaque segment de longueur supérieure à 1 dans la liste, l'algorithme lit un nombre dans le flot de données comprimé, correspondant au nombre de points situés sur le demi-segment gauche. Le nombre de points situés sur le demi-segment droit est déduit du nombre de points sur le segment entier et du nombre lu. Ensuite, le segment courant est retiré de la liste, et le ou les demi-segments non vides sont ajoutés en fin de liste. L'algorithme termine lorsqu'il n'y a plus de segment divisible dans la liste. L'ensemble de la partie décodage est détaillé ci-dessous.

Algorithme *Décodage de points sur un segment de droite*

1. lire le nombre de points sur le segment de droite original $\mathcal{S}_0$ sur 32 bits
2. $\mathcal{L} \leftarrow \mathcal{S}_0$
3. **tant que** $\mathcal{L}$ contient des segments de longueur supérieure à 1
4. **faire**
5. $\quad \mathcal{S} \leftarrow$extraire le premier segment de $\mathcal{L}$
6. $\quad n \leftarrow$nombre de points situés sur $\mathcal{S}$
7. $\quad$ lire le nombre de points n_1 situés sur le demi-segment gauche de $\mathcal{S}$ sur $\log_2(n+1)$ bits
8. $\quad n_2 \leftarrow n - n_1$
9. $\quad$ **si** $n_1 > 0$
10. $\quad\quad$ **alors** $\mathcal{S}_1 \leftarrow$demi-segment gauche de $\mathcal{S}$
11. $\quad\quad\quad$ ajouter $\mathcal{S}_1$ à la fin de $\mathcal{L}$
12. $\quad$ **si** $n_2 > 0$
13. $\quad\quad$ **alors** $\mathcal{S}_2 \leftarrow$demi-segment droit de $\mathcal{S}$
14. $\quad\quad\quad$ ajouter $\mathcal{S}_2$ à la fin de $\mathcal{L}$

Au fur et à mesure que l'algorithme progresse, les données lues permettent de localiser les points avec plus de précision. Il est donc possible de visualiser l'ensemble des points aux étapes intermédiaires du décodage, avec une précision sur leurs coordonnées égale à la longueur courante des segments. Pour chaque segment $\mathcal{S}_i$, il suffit de générer n_i points (uniformément distribués par exemple) entre ses extrémités.

Pour généraliser cet algorithme à n'importe quelle dimension, définissons une cellule comme l'objet géométrique contenant les points à coder. En dimension 1, 2 et 3, les cellules sont respectivement le segment de droite, le rectangle, et le parallélépipède rectangle. La seule partie de l'algorithme qui diffère d'une dimension à l'autre est la subdivision de la cellule. En dimension d, une cellule doit être subdivisée d fois (le long de chacun des d axes). Par conséquent, un ordre de subdivision pour les cellules doit être fixé au moment du codage (nous reviendrons sur la question du choix optimal par la suite), et porté à la connaissance du décodeur par un code d'en-tête.

La Figure 1 représente un exemple bidimensionnel. Les nombres de points transmis par le codeur sont accompagnés du nombre de bits correspondant en dessous, et les nombres de points déductibles (donc non transmis) sont écrits entre parenthèses. La Figure 2 montre le code résultant.

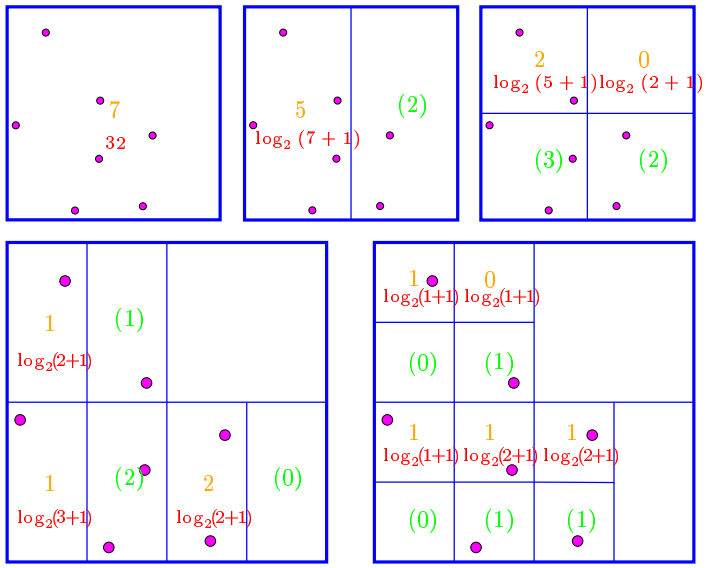

FIG. 1 – *L'algorithme de codage sur un exemple bidimensionnel*

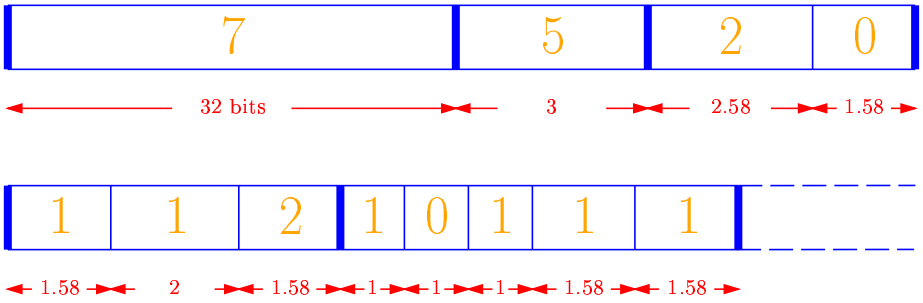

FIG. 2 – *Le code correspondant à l'exemple bidimensionnel*

Une analyse théorique montre que cet algorithme engendre un gain minimal de $\log_2 n - 2{,}402$ par point, soit, pour l'ensemble, si l'on néglige la constante additive, $n \log_2 n$. Ce gain correspond exactement à l'information d'ordre sur les points ($\log_2 n$ bits sont nécessaires pour coder le numéro d'un point parmi n). Autrement dit, l'algorithme fait l'économie du codage de l'information d'ordre sur les points. Il est à noter que ce gain théorique est une borne inférieure: la distribution uniforme des

points constitue le cas le pire pour l'algorithme. En effet, la méthode tire parti de distributions non uniformes, et son efficacité est d'autant plus grande que les données traitées sont structurées, ce qui est cohérent avec la théorie de l'information.

En outre, il est possible d'utiliser un prédicteur pour améliorer les taux de compression. Nous avons vu que pour une cellule c contenant p points, le nombre de points contenus dans la première sous-cellule était codé sur $\log(p+1)$ bits. Cela sous-entend que tous les nombres de points entre 0 et p sont équiprobables, de probabilité $1/(p+1)$. Or, si l'on suppose que la distribution des points à l'intérieur de c est corrélée à la distribution des points dans les cellules voisines, il est possible d'affecter aux valeurs $[0..p]$ des probabilités calculées à partir de l'analyse du voisinage de c. Le prédicteur que nous avons implanté suivant ce principe améliore la compression géométrique de 5 à 10% sur un ensemble d'objets 3D usuels.

3 Principe du codeur topologique

3.1 Simplification de maillages

Le codeur topologique que nous proposons s'inspire des procédés utilisés en simplification séquentielle de maillages. La technique commune aux différents algorithmes de simplification est la *décimation*. On part du maillage original et on lui applique itérativement une opération élémentaire (ou primitive) de simplification consistant à supprimer un élément du maillage (sommet, arête ou face) pour obtenir des versions de plus en plus grossières de l'objet initial. L'aspect progressif de ces algorithmes de simplification les rattachent naturellement à notre champ d'étude. En revanche, il existe une différence fondamentale qui les en écartent: ce type de codage ne débouche pas nécessairement sur une représentation plus compacte des modèles. Au contraire, les transformations appliquées au maillage pour le simplifier progressivement ont tendance à *augmenter* le volume de données total (nous entendons par là le volume de données nécessaire à la reconstruction du maillage original).

Dans le cas le plus largement étudié où les mailles sont triangulaires, on distingue essentiellement cinq opérateurs de simplification, qui permettent de classifier les méthodes existantes:

- la suppression de sommet (cf Figure 3): cette opération consiste à supprimer un sommet et ses arêtes incidentes, créant ainsi un trou polygonal. Pour conserver une connectivité de triangulation, le polygone est retriangulé de manière à satisfaire un critère d'optimisation pour l'approximation du maillage original [36].

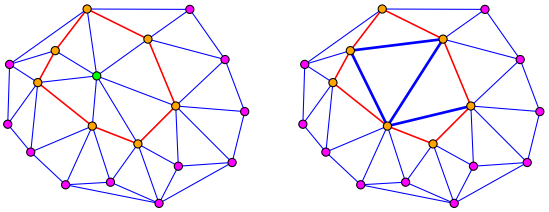

FIG. 3 – *La suppression de sommet*

– la contraction d'arête (cf Figure 4): les deux sommets de l'arête sont fusionnés, ce qui entraîne la disparition de ses deux faces adjacentes (une seule dans le cas où l'arête appartient à un bord de la triangulation) qui dégénèrent en triangles aplatis. La manière de choisir la position du sommet résultant de la contraction varie suivant les critères d'optimisation des différentes méthodes fondées sur cette opération [20, 2, 28, 19]. La principale limitation de ces méthodes est de traiter exclusivement le cas des triangulations correspondant à une variété bidimensionnelle orientable.

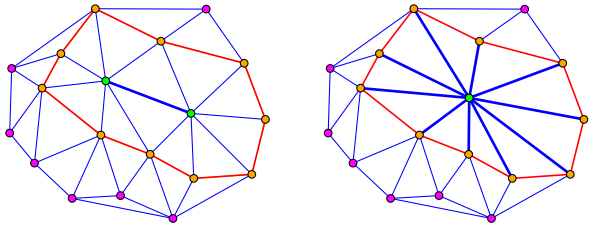

FIG. 4 – *La contraction d'arête*

– la demi-contraction d'arête: il s'agit d'un cas particulier de la contraction d'arête où la position du sommet résultant est contrainte à l'un des deux sommets de l'arête originale, de sorte que l'ensemble des sommets des versions simplifiées du maillage est un sous-ensemble des sommets du maillage original [25].
– la fusion de sommets (cf Figure 5): il existe une version généralisée de la contraction d'arête qui consiste à fusionner deux sommets non nécessairement incidents. Cet opérateur de simplification permet d'élargir le champ d'application de la contraction d'arêtes à une triangulation quelconque (i.e. qui ne soit

pas nécessairement une variété, éventuellement non orientable, et en dimension quelconque) [32, 15, 14, 12].

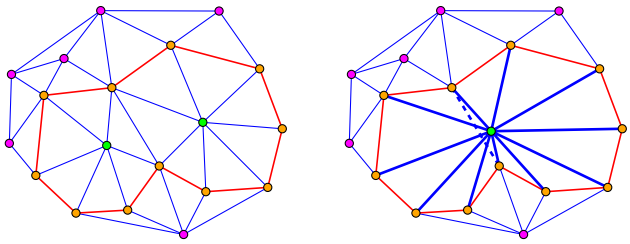

FIG. 5 – *La fusion de sommets*

– la contraction de face (cf Figure 6): les trois sommets du triangle sont fusionnés, ce qui entraîne la disparition par dégénérescence de 2 à 4 triangles (le triangle contracté et ses 3 voisins dans le cas général) [16].

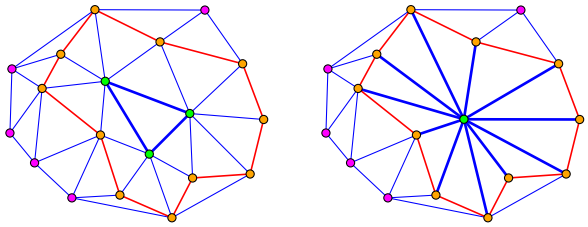

FIG. 6 – *La contraction de face*

Chaque primitive de simplification possède sa primitive inverse permettant, au cours du processus de décodage, de raffiner progressivement un maillage de départ grossier pour reconstruire progressivement le modèle original.

3.1.1 Principe général

La contrainte principale imposée à notre codeur topologique a été dictée par le codeur géométrique. Pour pouvoir compresser à la fois la connectivité et les positions d'un maillage triangulaire, il était nécessaire que le codeur topologique repose sur le schéma de subdivision successive de cellules qui gouverne la compression des positions. Pour décrire l'algorithme de simplification, nous nous plaçons donc dans la

situation où les cellules ont été suffisamment subdivisées par le codeur géométrique pour que tous les sommets soient isolés (chaque cellulle contient exactement un sommet). Dans cette situation, même si la position des sommets obtenus n'est pas connue avec toute la précision originale, on obtient un ensemble de points $P\prime$ de même cardinal que l'ensemble P initial et sur lequel ont peut donc *appliquer* la connectivité du modèle à comprimer. L'algorithme de compression topologique consiste ainsi à parcourir le chemin inverse de la subdivision des cellules en procédant par fusion de cellules. À chaque étape, l'information nécessaire à la restauration de la connectivité des cellules originales est codée.

À l'issue d'une fusion de cellules, les 2 cellules originales c_i et c_{i+1} sont remplacées par une cellule contenant un *super-sommet*, unique représentant des deux sommets contenus dans c_i et c_{i+1}. Une fusion de cellules est donc équivalente à une fusion de sommets et par conséquent, nous utiliserons dans la suite les outils et le vocabulaire propre à la simplification séquentielle de triangulation. Le codeur topologique utilise 2 des primitives de simplification précédemment énumérées:

- la contraction d'arête permet de fusionner deux cellules incidentes (cf Figure 4) (nous appellerons sa primitive inverse l'expansion d'arête),
- la fusion de sommets permet de fusionner deux cellules non incidentes (cf Figure 5) (nous appellerons sa primitive inverse la subdivision de sommet).

Dans les méthodes de simplification de maillages publiées jusqu'à maintenant, l'algorithme avait un degré de liberté pour le choix des éléments du maillage sur lesquels l'opérateur agissait. Ce degré de liberté a deux conséquences principales sur le fonctionnement de ces méthodes:

- la première est qu'il est possible d'optimiser la décimation pour approcher l'objet au plus près le plus longtemps possible. Ainsi, pour réduire la distorsion géométrique au minimum, une queue de priorité est maintenue dynamiquement sur les éléments du maillage auxquels la primitive est appliqué. À chaque étape, c'est l'élément qui minimise un critère de proximité au maillage original (ou parfois, au maillage à l'étape précédente) qui est supprimé,
- la seconde conséquence est que ce choix se faisant *avant* la simplification, le décodeur n'est pas en mesure d'identifier quels éléments ont été transformés. Il est donc nécessaire de lui indiquer explicitement par un code supplémentaire les indices de ces éléments.

Dans notre cas, c'est l'ordre de subdivision des cellules imposé par le codeur géométrique qui gouverne la fusion des sommets. Or, cet ordre étant canonique, aucun

code supplémentaire n'a besoin d'être transmis au décodeur. L'information topologique générée par le codeur à chaque fusion de cellule consiste donc en un symbole identifiant laquelle des 2 primitives a été utilisée, suivi des paramètres décrivant la manière dont cette primitive a agit sur la connectivité originale.

3.2 L'opérateur de contraction d'arête

La contraction d'arête est appliquée lorsque les deux cellules à fusionner c_i et c_{i+1} sont incidentes, et que le voisinage de l'arête est *2-manifold* et *orientable*. Dans ces conditions, la contraction d'arête provoque la disparition de ses 2 faces adjacentes par dégénérescence, et l'information nécessaire à l'opérateur inverse se limite aux indices de 2 arêtes parmi l'ensemble des arêtes incidentes à la cellule fusionnée résultante (cf Figure 7).

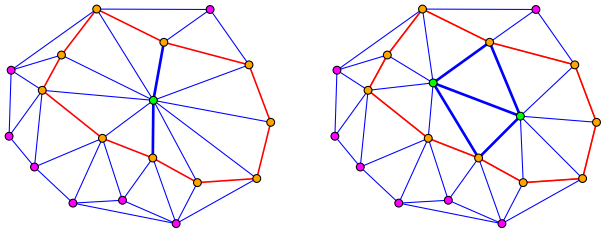

FIG. 7 – *L'expansion d'arête*

Si le sommet fusionné est de degré d, la longueur du code associé à cette opération est $\log_2 \binom{d}{2}$ bits. Pour un degré moyen de 6, 4 bits suffiront donc à coder une contraction d'arête. En pratique, l'utilisation de diverses techniques de prédiction permet d'atteindre des coûts inférieurs à 3 bits sur certains modèles.

3.3 L'opérateur de fusion de sommets

Lorsque les cellules à fusionner ne sont pas incidentes, ou que leur voisinage local possède une topologie complexe, un opérateur plus général, mais aussi plus coûteux, est utilisé. Il s'agit de la primitive de fusion de sommet, dont la primitive inverse a été décrite par Popović et Hoppe sous le nom de *generalized vertex split* (subdivision de sommet généralisée) dans l'article *Progressive Simplicial Complexes* [32]. L'utilisation de cette opération permet de simplifier (et, dans le cadre de ce rapport, de coder de manière efficace) n'importe quel complexe simplicial, ce qui

est beaucoup plus général que le cas d'une triangulation manifold, habituellement rencontré dans les travaux connexes en compression ou simplification de maillages.

Le principe général de la fusion de sommets revient à coder de manière exhaustive l'évolution des simplexes incidents à chacun des 2 sommets fusionnés. Plus précisément, lors de la fusion du sommet v_2 avec le sommet v_1, tout simplexe s incident à v_2 est remplacé par le simplexe $s\prime = (s \setminus v_2) \cup v_1$. Lorsque $s\prime$ existe déjà, s est supprimé.

La chaîne codante associée à une fusion de sommets doit permettre au décodeur de recréer à partir du sommet fusionné v_1v_2 l'ensemble des simplexes incidents à v_1 et v_2 avant la fusion. Pour cela, chaque simplexe s incident à v_1v_2 se voit affecter un code compris entre 1 et 4 décrivant son évolution au terme de la subdivision de v_1v_2:

- code 1: s est incident à v_1,
- code 2: s est incident à v_2,
- code 3: s est incident à v_1 et v_2,
- code 4: s est incident à v_1 et v_2, et il y a création d'un simplexe S de dimension $dim(s)+1$ incident à v_1 et v_2. Ce code correspond au cas de figure où la fusion de v_1 et v_2 a entrainé la disparition d'un simplexe (par exemple, l'arête v_1v_2).

Pour le détail du codage efficace de l'opérateur de fusion de sommets, on se reportera à l'article de Popović et Hoppe [32]. L'utilisation de contraintes limitant les possibilités d'apparition de certains codes suivant le contexte permet de réduire la taille du code associé à cet opérateur de 30 bits à 15 bits sur des objets 3D usuels. En outre, la distribution des codes n'étant pas équiprobable, un codage entropique additionnel conduit à un coût final d'environ 8 bits par fusion de sommet.

4 Statistiques et résultats expérimentaux

4.1 Statistiques sur l'utilisation des opérateurs

Nous avons vu que le codeur topologique utilisait deux opérateurs différents: la contraction d'arête, codée sur 4 bits en moyenne, et la fusion de sommets, codée sur 8 bits. L'efficacité de notre algorithme dépend donc dans une large mesure de la proportion de ces opérateurs dans le processus de fusion de cellules. La Figure 8 montre quelques statistiques sur cette proportion réalisées sur des modèles 3D usuels. La dernière colonne donne le coût de la séparation des opérateurs par codage arithmétique, c'est-à-dire le coût du code d'en-tête indiquant au décodeur le type de l'opération suivante.

modèles	nombre de sommets	contractions d'arêtes	fusions de sommets	coût de la séparation
triceratops	2832	76,4%	13,6%	0,79 bit
blob	8033	90,9%	9,1%	0,44 bit
fandisk	6475	96,0%	4,0%	0,24 bit
bunny	35947	93,0%	7,0%	0,37 bit
horse	19851	92,4%	7,6%	0,39 bit
moyenne	73138	92,2%	7,8%	0,39 bit

FIG. 8 – *pourcentage de contractions d'arêtes / fusions de sommets*

4.2 Résultats expérimentaux

Nous présentons dans la Figure 9 quelques résultats de notre algorithme comparés à ceux de Cohen-Or [38] et Pajarola [30]. Nous faisons également apparaître à titre indicatif les résultats obtenus en simple résolution (codage non progressif, donc plus compact) par Touma et Gotsman [39]. Pour chaque modèle et pour chaque algorithme testé, la première ligne donne le nombre de bits par sommets pour le codage de la topologie, et la seconde le nombre de bits par sommets pour le codage de la géométrie.

modèles	nombre de sommets	T & G 1998	C et al. 2000	P & R 2000	notre algo.
triceratops	2832	2,2 20,0	5,8 20,4	7,4 21,0	6,0 19,2
blob	8033	1,7 20,0	5,9 19,7	7,6 21,0	4,1 20,1
fandisk	6475	1,1 9,0	?	6,8 15,0	2,9 12,1
bunny	35947	?	?	7,0 16,0	3,2 14,7
horse	19851	2,3 17,0	5,7 15,4	?	3,9 16,4
moyenne	73138	2,0 16,5	5,8 17,0	7,1 16,9	3,6 15,7

FIG. 9 – *banc d'essai sur la compression de modèles 3D*

5 Généralisation aux dimensions supérieures

5.1 La fusion de sommets

La généralisation de l'opérateur de fusion de sommets est directe et possible en dimension quelconque. De ce fait, notre algorithme est facilement applicable à tout complexe simplicial en dimension d. Cependant, en l'absence d'un opérateur de coût réduit — l'équivalent en dimension d de la contraction d'arête — les performances en terme de taux de compression seront moins bonnes que dans le cas des surfaces triangulées. Le coût moyen du codage d'une fusion de sommets dans le cas de maillages tétraèdriques est de 120 bits par la méthode naïve, de 60 bits si l'on applique les règles de contraintes réduisant le nombre de codes possibles dans certains contextes, et de 40 bits en utilisant en outre un codeur entropique.

5.2 La contraction d'arête

Cet opérateur est plus délicat à généraliser. Nous proposons ici un opérateur équivalent adapté aux maillages tétraèdriques. Le principe général est de distinguer les fusions de sommets dont l'opération inverse remplit les conditions suivantes (cf Figure 10):

- le sommet à subdiviser est de code 4 (expansion d'arête),
- l'ensemble des faces de code 4 (ie les faces donnant naissance à un tétraèdre lors de la subdivision) forme une surface S 2-manifold,
- les arêtes restantes (ie en dehors de S) sont toutes de code 1 ou 2,
- toutes les arêtes de code 1 sont du même côté de S, et toutes les arêtes de code 2 sont de l'autre côté de S.

Lorsque toutes ces conditions sont réunies, la chaîne codante permettant de reconstituer l'état du voisinage du sommet avant la fusion est formée par:

- le nombre de faces de code 4,
- la liste des indices des faces de code 4,
- le code des arêtes situées du "premier" côté de S.

Là encore, la prédiction et le codage entropique permettent d'obtenir un code plus compact. Avec les techniques utilisées dans notre algorithme, une opération de contraction d'arête en 3D coûte environ 20 bits.

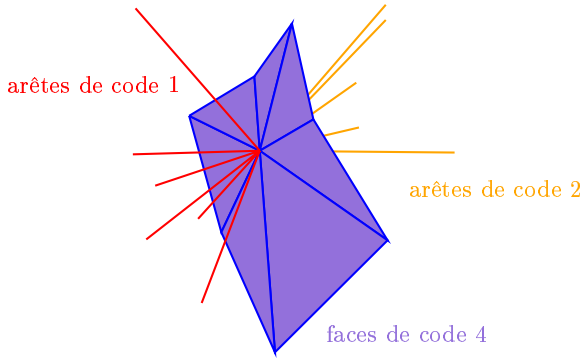

FIG. 10 – *Cas favorable à l'expansion d'arête 3D*

5.3 Résultats expérimentaux sur la compression tétraèdrique

Comme dans le cas des maillages surfaciques, les résultats de l'algorithme dépendent essentiellement de la fréquence d'apparition de l'opérateur de contraction d'arête. Sur une tétraèdrisation de Delaunay d'un nuage de 10000 points générés aléatoirement dans une sphère, ce pourcentage est d'environ 40 et conduit à un coût final inférieur à 35 bits par sommet pour le codage progressif de la topologie. Ces résultats sont à comparer avec ceux obtenus par Pajarola et al. [31], les seuls à notre connaissance à proposer un codeur topologique progressif pour les maillages tétraèdriques. Pour une tétraèdrisation de Delaunay aléatoire de 10000 sommets, le taux de compression obtenu est de l'ordre de 45 bits par sommet.

6 Conclusion

Nous avons présenté dans ce rapport un nouveau codeur topologique basé sur des techniques de simplification de maillage optimisées pour la compression. Ces travaux s'inscrivent dans la continuité de notre algorithme de compression géométrique [11] et permettent par conséquent un codage progressif conjoint de la géométrie et de la topologie du maillage. Outre les taux de compression obtenus, particulièrement compétitifs par rapport aux méthodes de codage progressif concurrentes, notre algorithme présente l'avantage de s'appliquer a tout complexe simplicial, ce qui inclut en particulier les triangulations non manifold ou les soupes de polygones. De plus, la méthode est généralisable en dimension quelconque, et offre pour le cas des maillages

tétraèdriques un gain d'efficacité d'environ 25% sur les précédents travaux dans le domaine.

Références

[1] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std* 754 – 1985. New York, NY, 1985. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

[2] M.-E. Algorri and F. Schmitt. Mesh simplification. *Computer Graphics Forum*, 1996.

[3] P. Alliez. *Étude de la Représentation Géométrique et Texturelle de Scènes 3D pour les Services de Visualisation dans un Contexte Télécommunicant.* PhD thesis, École Nationale Supèrieure des Télécommunications, 2000.

[4] P. Alliez and N. Laurent. Compression et représentation échelonnable de maillages triangulaires. In *AFIG - Journées de l'Association Française d'Informatique Graphique*, 1999.

[5] C. Bajaj, S. Cutchin, V. Pascucci, and G. Zhuang. Error resilient streaming of compressed vrml. Technical report, University of Texas, Austin, 1999.

[6] C. Bajaj, V. Pascucci, and G. Zhuang. Progressive compression and transmission of arbitrary triangular meshes. In *IEEE Visualization 99 Conference Proc.*, October 1999.

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

[8] D. Cohen-Or, D. Levin, and O. Remez. Progressive compression of arbitrary triangular meshes. In *IEEE Visualization 99 Conference Proc.*, pages 67–72, 1999.

[9] M. Deering. Geometry compression. In *SIGGRAPH 95 Conference Proc.*, pages 13–20, 1995.

[10] O. Devillers and P.-M. Gandoin. Compression géométrique pour une transmission progressive. In *AFIG 99*, 1999.

[11] O. Devillers and P.-M. Gandoin. Geometric compression for interactive transmission. In *IEEE Visualization 2000 Conference Proc.*, 2000.

[12] C. Erikson and D. Manocha. Gaps: General and automatic polygonal simplification. In *Symposium on Interactive 3D Graphics 99 Proc.*, 1999.

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

[14] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization 98 Conference Proc.*, 1998.

[15] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Conference Proc.*, 1997.

[16] T. S. Gieng, B. Hamann, K. I. Joy, G. L. Schussman, and I. J. Trotts. Smooth hierarchical surface triangulation. In *IEEE Visualization 97 Conference Proc.*, 1997.

[17] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48, March 1991.

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

[19] H. Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proc.*, 1996.

[20] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH 93 Conference Proc.*, 1993.

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

[22] M. Isenburg and J. Snoeyink. Mesh collapse compression. In *Symposium on Computational Geometry*, 1999.

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

[24] D. King and J. Rossignac. Optimal bit allocation in 3d compression. Technical report, Georgia Institute of Technology, 1999.

[25] L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. In *SIGGRAPH 98 Conference Proc.*, 1998.

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

[27] J. Li and C.-C. Jay Kuo. Progressive coding of 3d graphic models. *IEEE Computer Graphics*, 86, 1998.

[28] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization 98 Conference Proc.*, 1998.

[29] R. Pajarola and J. Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, January–March 2000.

[30] R. Pajarola and J. Rossignac. Squeeze: Fast and progressive decompression of triangle meshes. *CGI 2000 Proc.*, pages 173–182, 2000.

[31] R. Pajarola, J. Rossignac, and A. Szymczak. Implant sprays: Compression of progressive tetrahedral mesh connectivity. In *IEEE Visualization 99 Conference Proc.*, 1999.

[32] J. Popović and H. Hoppe. Progressive simplicial complexes. In *SIGGRAPH 97 Conference Proc.*, 1997.

[33] J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM J. Res. Develop.*, 23(2):149–162, 1979.

[34] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEE Transactions on Visualization and Computer Graphics*, pages 47–61, 1999.

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

[36] W. Schröder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. In *SIGGRAPH 92 Conference Proc.*, 1992.

[37] G. Taubin, A. Guéziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *SIGGRAPH 98 Conference Proc.*, pages 123–132, 1998.

[38] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2), 1998.

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

[40] I.H. Witten, R. Neal, and J.G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

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

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 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 - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

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