Compression progressive et sans perte de structures géométriques

Gandoin Pierre-Marie

THESE

Présentée à
L'UNIVERSITE DE NICE-SOPHIA ANTIPOLIS - UFR SCIENCES

pour obtenir le titre de
DOCTEUR EN SCIENCES
spécialité
INFORMATIQUE

par

Pierre-Marie GANDOIN

COMPRESSION PROGRESSIVE ET SANS PERTE DE STRUCTURES GEOMETRIQUES

Directeur : Olivier DEVILLERS

Soutenue publiquement le 13 Septembre 2001
devant le jury composé de :

Michel BARLAUD, Président
Ferran HURTADO, Rapporteur
Jarek ROSSIGNAC, Rapporteur
François SILLION, Rapporteur
Jean-Daniel BOISSONNAT, Examinateur
Olivier DEVILLERS, Examinateur
Jean-Luc DUGELAY, Examinateur

COMPRESSION PROGRESSIVE ET SANS PERTE DE STRUCTURES GEOMETRIQUES

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, la simulation chirurgicale, 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 fini de conférer à ce problème une place centrale dans la recherche en informatique.

Ainsi, depuis 1995, de nombreux algorithmes ont été proposés pour la compression de maillages triangulaires, en utilisant le plus souvent l'approche suivante : les sommets du maillage sont codés dans un ordre établi pour contenir partiellement la topologie (ou connectivité) du maillage. Parallèlement, quelques règles simples permettent de prédire la position du sommet courant à partir des positions de ses voisins qui ont déjà été codés. Dans ce mémoire, nous avons choisi de donner plutôt la priorité à la compression des positions des sommets. Nous décrivons un ensemble de méthodes de codage progressif, sans perte d'information, adaptées à une large classe de structures géométriques (non nécessairement triangulaires ni *manifold*, de genre quelconque), et généralisables à n'importe quelle dimension. Les taux de compression obtenus se positionnent avantageusement par rapport aux méthodes progressives actuelles les plus efficaces : par exemple, pour le cas particulier des 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 : compression, codage, géométrie, maillage, triangulation, tétraédrisation, transmission, visualisation, progressivité, niveaux de détail, sans perte.

PROGRESSIVE AND LOSSLESS GEOMETRIC COMPRESSION

For a few years, meshes have been conquering a predominant status amongst the different types of computer data structure for geometric objects. More precisely, the meshes composed of simplices — triangles for the modeling of surfaces embedded in 3D, tetrahedra for the modeling of volumes — seem to be the most widespread at the present time. The rapid growth of applications using these geometric structures in such various fields as finite elements computing, simulation of surgery, or video games has quickly raised the problem of an efficient coding, well suited for storage or visualization. The advent 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, increases this need.

Therefore, since 1995, a large number of algorithms have been proposed for the compression of triangular meshes, using for most of them the following approach: the vertices of the mesh are coded in an order such that it contains partially the topology of the mesh. In the same time, some simple rules attempt to predict the position of the current vertex from the positions of its neighbors that have been previously coded. In this thesis, we have chosen to give the priority to the compression of the vertex positions rather than their connectivity. We describe a set of coding methods which are progressive, lossless, adapted to arbitrary meshes (non necessarily triangular nor manifold, with arbitrary genus), and generalizable to any dimension. Our results are competitive with regards to the best current progressive methods: for instance, for the particular case of triangular surface meshes, about 3.6 bits per vertex are necessary on average to code the connectivity of usual models.

Keywords: compression, coding, geometry, mesh, triangulation, tetrahedralization, transmission, visualization, progressivity, levels of detail, lossless.

Thèse préparée à l'INRIA Sophia Antipolis dans le projet PRISME

Remerciements

En premier lieu, je voudrais remercier Olivier pour la qualité de son encadrement. Plus encore que son efficacité strictement professionnelle, je voudrais évoquer ici sa disponibilité constante et la confiance qu'il m'a témoignée tout au long de ces trois ans.

Pour des raisons évidentes de diplomatie élémentaire liée à la paix des ménages, je ne saurais remercier Olivier sans remercier également Monique. Un grand merci pour les conseils avisés, les relectures attentives (pour ne pas dire maniaques) et la (bonne) compagnie au quotidien.

Parmi les gens qui m'ont soutenu dans mon travail au cours de ces années, Agnès occupe sans aucun doute une place privilégiée. Je la remercie pour son extraordinaire générosité et sa bonne humeur permanente.

Je tiens aussi à dire un grand merci à Sylvain pour sa patience et son dévouement exemplaires, et pour le temps considérable sacrifié aux problèmes pratiques ou théoriques des uns et des autres (stagiaires, thésards, et chercheurs permanents confondus).

D'une manière générale, je remercie Jean-Daniel et tous les membres de l'équipe PRISME (Alexandra, Andreas, Darka, David, Dimitrii, Dominique, Frank, François A., François R., Frédéric, Hervé, Julia, Mariette, Philippe de M., Philippe G., Raphaëlle, Sophie, Stéphane, Tharavy) pour les conversations fructueuses, les intuitions géniales, les avis avertis, les orientations bénéfiques, les suggestions audacieuses et les critiques avisées.

Merci bien sûr à Ferran pour son accueil chaleureux, et à tous les chercheurs avec qui j'ai eu l'honneur et le plaisir de travailler pendant mon séjour à Barcelone (Vera, Pedro, Regina).

Enfin, outre Ferran, Jean-Daniel et Olivier, je remercie Michel Barlaud,

Jarek Rossignac, François Sillion et Jean-Luc Dugelay pour avoir pris le temps de se pencher sur mes travaux, et pour toutes les remarques qui ont permis d'améliorer ce mémoire.

S'il est possible d'affirmer que tous les gens cités jusqu'ici ont eu une influence indéniable sur la préparation de cette thèse, il me paraît en revanche tout à fait déraisonnable d'espérer pouvoir recenser toutes les personnes qui d'une manière ou d'une autre, de près ou de loin, directement ou indirectement, ont infléchi ma manière d'être et de penser tout au long de ces trois ans.

A ce titre, je remercie de tout mon cœur mon père, ma mère, frère Nicolas, frère Aurélien et frère Guillaume, ma grand-mère, et bien sûr Raphaëlle.

Mais je remercie aussi, dans le désordre, les membres de la famille que je vois moins souvent et qui comptent tous beaucoup pour moi (avec une mention spéciale pour la branche corse, un peu susceptible), les amis de longue date, les amis de fraîche date, et tous ceux qui à un moment ou à un autre de ces années, m'ont aidé à choisir mon chemin. Merci à Afifa, Agnès, Alex, Alexandra, Alison, André, Annabelle, Anne, Anne-Claire, Anne-Marie, Antoine, Audrey, Babette, Bernard, Bertille, Brice, Bruno, Camille, Caroline, Cécile, Célia, Céline, Charlotte, Christian F., Christian P., Claire, Clara, Claude, Clément, Clémentine, Colette, Corinne, Cyril, Cyrille, David, Dédé, Denise, Donatello, Doudou, Emilie, Emmanuelle, Florence, Francis, François, Fred, George-Marie, Gérard, Gérard C., Gilles, Guillaume, Hélène, Hugues, Imad, Inger, Jean, Jean-Daniel, Jean-Jacques, Jean-Louis, Jean-Marc, Jean-Philippe, Jean-Pierre, Jeff, Julia, Katia, Lætitia, Laura, Laurent B., Laurent C., Laurent D., Laurent P., Léon, Leslie, Louis, Luc, Manu, Marc, Marie, Marie-Ange, Marie-Claire, Marie-George, Marion, Mehdi, Mélaine, Michel, Michèle, Monique, Muriel, Nathalie I., Nathalie M., Nicolas M., Nicolas R., Nicole, Olivier, Pascale, Patrick, Paul, Philippe B., Philippe G., Pierre C., Pierre G., R. Ferri, Rémi, Robert, Rosette, Sam, Sophie B., Sophie G., Sophie R., Stein, Stéphane, Sylvain, Sylviane, Tatie, Véronique, Yan, Yvon, Zoubir, et aux autres...

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, la simulation chirurgicale, 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 fini de conférer à ce problème une place centrale dans la recherche en informatique.

Ainsi, depuis 1995, de nombreux algorithmes ont été proposés pour la compression de maillages triangulaires, en utilisant le plus souvent l'approche suivante : les sommets du maillage sont codés dans un ordre établi pour contenir partiellement la topologie (ou connectivité) du maillage. Parallèlement, quelques règles simples permettent de prédire la position du sommet courant à partir des positions de ses voisins qui ont déjà été codés. Dans ce mémoire, nous avons choisi de donner plutôt la priorité à la compression des positions des sommets. Nous décrivons un ensemble de méthodes de codage progressif, sans perte d'information, adaptées à une large classe de structures géométriques (non nécessairement triangulaires ni *manifold*, de genre quelconque), et généralisables à n'importe quelle dimension. Les taux de compression obtenus se positionnent avantageusement par rapport aux méthodes progressives actuelles les plus efficaces : par exemple, pour le cas particulier des 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.

Abstract

For a few years, meshes have been conquering a predominant status amongst the different types of computer data structure for geometric objects. More precisely, the meshes composed of simplices — triangles for the modeling of surfaces embedded in 3D, tetrahedra for the modeling of volumes — seem to be the most widespread at the present time. The rapid growth of applications using these geometric structures in such various fields as finite elements computing, simulation of surgery, or video games has quickly raised the problem of an efficient coding, well suited for storage or visualization. The advent 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, increases this need.

Therefore, since 1995, a large number of algorithms have been proposed for the compression of triangular meshes, using for most of them the following approach: the vertices of the mesh are coded in an order such that it contains partially the topology of the mesh. In the same time, some simple rules attempt to predict the position of the current vertex from the positions of its neighbors that have been previously coded. In this thesis, we have chosen to give the priority to the compression of the vertex positions rather than their connectivity. We describe a set of coding methods which are progressive, lossless, adapted to arbitrary meshes (non necessarily triangular nor manifold, with arbitrary genus), and generalizable to any dimension. Our results are competitive with regards to the best current progressive methods: for instance, for the particular case of triangular surface meshes, about 3.6 bits per vertex are necessary on average to code the connectivity of usual models.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Cadre et motivations

1.1.1 Structures géométriques

Si l'on se penche sur l'histoire de l'informatique, on peut observer, parallèlement aux différentes avancées technologiques et algorithmiques qui ont jalonné cette science, une évolution très nette de la nature de l'information manipulée. Les premiers programmes conçus pour le traitement automatique de l'information étaient tournés essentiellement vers les nombres (calculateurs scientifiques) ou le texte (gestionnaires de bases de données). Puis les techniques d'échantillonnage ont permis de modéliser et de rendre représentable en machine la réalité quotidienne sous forme de tableaux de nombres à une dimension (pour le cas de données sonores) et à deux dimensions (pour le cas des images fixes dites *bitmap*). Enfin, le passage à la vidéo s'est fait naturellement lorsque les capacités des ordinateurs grand public ont permis de stocker et d'afficher à une cadence soutenue des images *bitmap* fixes.

Ces différents types de données sont aujourd'hui couramment utilisés, et ont donné naissance, dans le cadre de l'informatique théorique, à un grand nombre de recherches conduisant à des structures de données et des algorithmes adaptés. En informatique appliquée, l'ensemble de ces informations constitue ce que l'on appelle communément le *multimédia*.

Cependant, avec le développement récent de l'infographie et de la vision

3D, de nouvelles techniques ont vu le jour pour la modélisation *tridimensionnelle* du monde qui nous entoure. Le principe général commun à ces méthodes revient à modéliser un objet (ou une scène complexe formée de plusieurs objets) par une description de type *géométrique*, c'est-à-dire basée sur l'utilisation de primitives géométriques comme des points, des polygones, des sphères, ..., positionnées dans l'espace tridimensionnel euclidien. Ce sont ces structures géométriques qui sont placées au centre de notre étude : elles peuvent consister en ensembles de points dans l'espace, en maillages surfaciques (des surfaces formées de polygones et plongées dans l'espace), ou en maillages volumiques (des volumes subdivisés en polyèdres).

Actuellement, ce type de données géométriques est en train de supplanter le *multimédia* traditionnel, comme le montre l'extraordinaire diversité des applications concernées. On peut citer entre autres :

- les systèmes d'information géographique (*SIG*, ou *GIS* en anglais), incluant la cartographie, la topographie, la géologie ou l'astronomie. Des relevés de terrain réalisés par imagerie traditionnelle, satellite ou radar, offrent la possibilité de modéliser avec une extrême précision la surface de la Terre ou des planètes accessibles aux instruments de mesure actuels (cf figure 1.1) ;

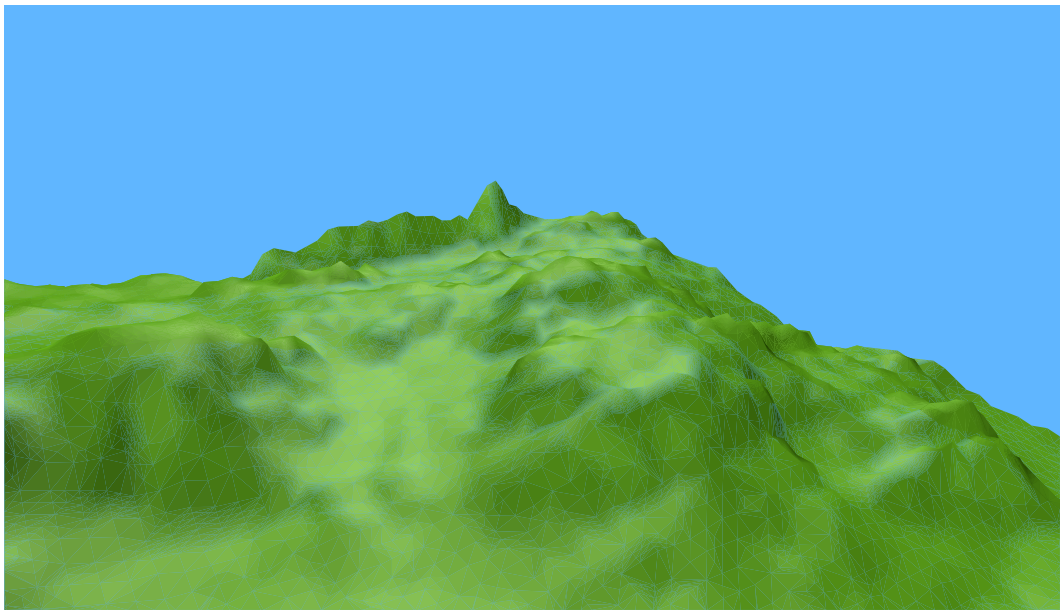
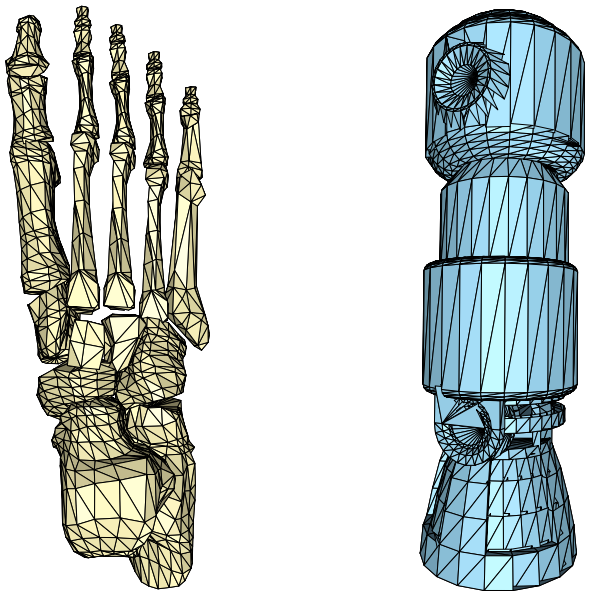


FIG. 1.1 – *Modèle de terrain*

- la médecine, avec en particulier l'aide au diagnostic, la chirurgie assistée par ordinateur ou la simulation. Des modélisations précises des organes humains, à la fois géométriques et physiologiques ouvrent la voie à des techniques aussi révolutionnaires que l'intervention chirurgicale à distance ou l'apprentissage par chirurgie virtuelle (cf figure 1.2) ;
- la conception assistée par ordinateur, utilisée dans les industries automobile et aéronautique, en architecture ou en *design*. Elle regroupe des techniques permettant de créer des objets à partir de volumes simples et d'optimiser leur usinage (cf figure 1.2) ;



(a) (b)

FIG. 1.2 – *Applications à la médecine (a) et à la C.A.O. (b)*

- le calcul par éléments finis. Dans de nombreuses branches de la physique et de l'ingénierie, les surfaces et les volumes sont subdivisés en mailles régulières afin de faciliter les calculs d'écoulement de fluides ou de propagation de chaleur, d'ondes électro-magnétiques, etc ;
- la simulation : à l'aide d'équations permettant de modéliser un phénomène physique de manière informatique, et d'objets géométriques recréant les objets réels dont on cherche à étudier le comportement, il est possible de réaliser des expériences virtuelles à moindre coût, avec une précision accrue sur les résultats, et un aspect parfaitement reproduc-

tible ;
- la réalité augmentée, qui permet de mélanger des modèles géométriques tridimensionnels à la vision naturelle. En particulier, il est possible pour un architecte de visualiser un projet sur son site de construction afin de se rendre compte de l'incidence de la construction sur le site naturel ;
- les visites virtuelles, qui peuvent jouer un rôle éducatif et culturel (musées), ou commercial (boutiques) (cf figure 1.3).

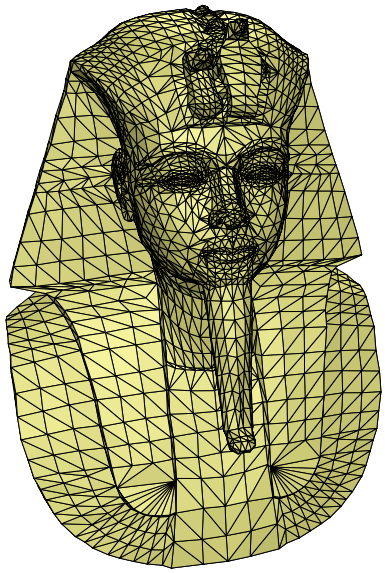


FIG. 1.3 – *Visite virtuelle de musée*

- les jeux vidéo, à l'origine d'une forte innovation en matière de modélisation tridimensionnelle par la recherche d'un réalisme sans cesse croissant.

En conséquence, il ne fait aucun doute que la compréhension et la maîtrise de l'information géométrique est un enjeu central dans la recherche en informatique théorique et appliquée pour les années à venir.

1.1.2 Compression

La compression de données est un domaine de l'informatique qui est né avec les tout premiers ordinateurs. En effet, le besoin de coder l'information de manière optimale s'est fait sentir très tôt, pour des raisons d'ordre

pratique : taille limitée des premières mémoires, nécessité de transmettre de l'information par le réseau téléphonique. Dès 1948, Claude Shannon publie ses premières idées sur ce qui restera connu sous le nom de *théorie de l'information* [94]. Il définit la notion d'entropie d'un message, qui correspond intuitivement à l'information *non redondante* portée par celui-ci, et propose le premier codeur entropique permettant d'attribuer à chaque symbole du message un code de longueur inversement proportionnel à sa probabilité d'apparition. Depuis cette date, les algorithmes se sont succédé en s'adaptant aux types de données à comprimer, et occupent encore aujourd'hui une place prépondérante au coeur de la recherche en informatique.

On peut se demander pourquoi une telle place est accordée à la compression de données alors que les mémoires des ordinateurs ont atteint actuellement une taille confortable, et continuent d'augmenter suivant la loi de Moore (i.e. d'environ 100% tous les 18 mois). Nous voyons principalement trois raisons à cela :

- il semblerait que le volume des informations à traiter augmente au moins aussi vite que la capacité des ordinateurs. Cela est dû d'une part à l'expansion permanente des différents domaines de la connaissance et à la richesse de notre environnement (bases de données géographiques, astronomiques, biologiques, etc.), et d'autre part à une numérisation systématique des documents (notamment dans le domaine des loisirs) ;
- la loi de Moore ne s'applique pas aux réseaux de manière homogène : aujourd'hui, des réseaux à haut débit cohabitent avec le réseau téléphonique standard utilisé par les modems grand public à débit limité. Par conséquent, la transmission efficace d'information continue de nécessiter une phase préalable de compression ;
- tous les types de mémoires informatiques ne sont pas équivalents en terme de capacité et d'efficacité : la taille de la mémoire cache est de l'ordre du mégaoctet (pour un temps d'accès de quelques nanosecondes), tandis que les disques durs actuels avoisinent le terraoctet (avec des temps d'accès de quelques millisecondes). La compression apparaît donc comme un outil permettant de réorganiser les données de manière optimale afin de tirer le meilleur parti des différentes mémoires.

Ainsi, en ce qui concerne le *multimédia* traditionnel (texte, son, image et vidéo), un niveau d'expertise très élevé semble avoir été atteint au cours des dernières décennies. On fait généralement la distinction entre :

- la compression *sans perte* d'information, qui fait appel au codage en-

tropique et à des techniques de modélisation statistiques et de prédiction. Après compression et décompression, l'information originale est reconstruite de manière exacte. Outre les codeurs de Huffman [52] et arithmétique [88, 87, 110] exploitant les idées de Shannon, citons les méthodes de codage par substitution (ou par dictionnaire) initiées par Ziv et Lempel en 1977 [112, 113]. Ces méthodes s'appliquent au texte, aux exécutables, et aux données sensibles (les images médicales par exemple) ;

– la compression *avec perte*, qui concerne principalement le son et l'image, et ou l'on s'autorise une légère perte d'information par rapport au document original, à condition que cette perte soit bien contrôlée et peu détectable par l'utilisateur. Parmi les méthodes les plus efficaces, on peut citer la transformée en cosinus discret (proche de la transformée de Fourier), la transformée en ondelettes, et la compression fractale. L'idée générale de ces méthodes est de tirer parti des faiblesses du système perceptuel humain : des outils de traitement du signal sont utilisés pour porter l'information du domaine spatial vers le domaine fréquentiel afin de simplifier le signal dans les fréquences moins bien perçues par l'homme.

Malheureusement, les structures géométriques, par leur complexité, ne se prêtent pas à une telle analyse du signal. En effet, contrairement au son et aux images, une structure géométrique usuelle n'est pas régulièrement échantillonnée, c'est-à-dire que l'ensemble des points qui composent un modèle tridimensionnel ne sont pas répartis sur une grille. En outre, il n'existe pas de fonction permettant de décrire globalement un objet géométrique, mais uniquement des paramétrisations locales. Enfin, l'ensemble des points contenus dans l'objet ne suffit généralement pas à définir celui-ci de manière unique ; un deuxième type d'information est nécessaire, composé des arêtes et des faces joignant les points (la *topologie*, ou *connectivité*).

Cette complexité rend l'information géométrique particulièrement lourde à coder. Par exemple, un maillage triangulaire sera formé par une liste de sommets (la *géométrie*) repérés par leurs coordonnées (3 nombres flottants, soit 96 bits), suivie d'une liste de faces (la *topologie*) décrites par les indices de leurs sommets dans la liste précédente (3 indices entiers, soit 96 bits). Etant donné que le degré moyen d'un sommet dans une triangulation est 6, l'indice d'un sommet apparaîtra 6 fois en moyenne dans la liste des faces, ce qui rend la description de la topologie très redondante et beaucoup plus coûteuse que la géométrie en terme de codage. La nécessité de comprimer

l'information géométrique est donc évidente.

Remarque 1.1.1 *Nous attirons l'attention du lecteur sur l'ambiguïté du terme* géométrie *et de ses dérivés dans ce mémoire (et plus généralement, dans la littérature consacrée à la compression géométrique). Une structure dite géométrique est constituée d'un ensemble de points et d'un ensemble de relations d'incidence entre ces points. Traditionnellement, en compression géométrique, on se réfère au premier ensemble par le terme* géométrie *(ou* positions*), et au second par le terme* topologie *ou* connectivité*). Par conséquent, pour respecter cette terminologie, une structure géométrique devrait être appelée structure géométrique* et topologique*; de même, l'expression* compression géométrique *est incorrecte dans la mesure où elle désigne généralement la compression de la géométrie* et de la topologie *d'une structure géométrique. Dans la suite de ce mémoire, nous nous sommes efforcés d'employer les termes* géométrie *ou* géométrique *dans des contextes suffisamment précis pour ne laisser place à aucune ambiguïté.*

1.1.3 Codage progressif et sans perte

Un code de taille minimale n'est pas le seul objectif visé dans la recherche d'une représentation optimale de l'information géométrique. Le développement actuel des réseaux pose le problème de la représentation *progressive* (ou *échelonnable*, ou encore *multirésolution*) des données. Dans le contexte de l'imagerie et de la vision pour une application réseau, un serveur doit transmettre des données à un client. Lorsqu'il s'agit d'un modèle géométrique volumineux, et que la bande passante du réseau est limitée, il est important que le client puisse avoir très tôt un aperçu, même relativement grossier, de l'objet transmis. Cela est rendu possible par un codage échelonnable de l'information : une première version du modèle, rudimentaire et peu coûteuse, est d'abord transmise, puis les données supplémentaires parvenant au client permettent de raffiner progressivement cette version simplifiée jusqu'à obtenir l'objet original. L'avènement du World Wide Web dans le domaine de la réalité virtuelle, et le caractère très hétérogène des technologies utilisées (tant au niveau des réseaux que des terminaux) confèrent aux méthodes de compression progressive un avantage décisif sur les techniques non progressives (compression en *simple résolution*, ou *monorésolution*).

D'autre part, il nous paraît important qu'au terme de la transmission (ou plus généralement, de la décompression), l'objet géométrique soit reconstruit

de manière exacte, en particulier avec la même précision sur les coordonnées de ses sommets que le modèle original. En effet, pour certaines applications professionnelles comme le calcul scientifique sur des objets de haute précision, la perte d'information ne peut être tolérée. En outre, la compression avec perte est interdite sur certaines données sensibles comme les informations médicales pour ne pas risquer de fausser un diagnostic.

Cependant, pour d'autres applications, une erreur bornée est admissible (réalité virtuelle grand public, jeux vidéo, etc.). Un atout précieux des méthodes *progressives sans perte* est donc de pouvoir s'adapter au type d'application, aux technologies mises en oeuvre, et à l'utilisateur final. C'est à ce dernier qu'appartiendra la décision de continuer la transmission pour augmenter la précision de son modèle, ou de l'arrêter avant terme.

1.1.4 Positionnement de la thèse

Si le problème de la compression de son, d'images bitmap ou de vidéo a été largement étudié dans le passé, la compression géométrique, située entre la géométrie algorithmique et la compression de données, est un domaine de recherche beaucoup plus récent. C'est le développement conjoint des applications de la synthèse d'image et des réseaux, qui a rendu nécessaires la manipulation et l'échange de données géométriques de manière rapide et économique. Depuis 1995, de nombreux articles se sont succédé à un rythme soutenu. Parmi les différentes contributions, on distingue les méthodes multirésolution des méthodes monorésolution, les méthodes avec perte des méthodes sans perte, celles qui traitent uniquement la compression de la topologie (ou *connectivité*), de celles qui s'intéressent aussi au codage efficace de la géométrie.

D'une manière générale, la tendance dominante de ces travaux est de donner la priorité au codage de la topologie, et pour certains, de s'appuyer sur la connaissance de l'information topologique pour essayer d'optimiser le codage de la géométrie. Dans ce mémoire, nous abordons le problème de la compression de structures géométriques de manière fondamentalement différente, en nous efforçant d'optimiser prioritairement la description de la *géométrie* du modèle, c'est-à-dire le codage des positions de ses sommets. L'idée sous-jacente est que très souvent, la topologie peut être reconstruite automatiquement à partir de la géométrie. Cependant, pour les cas où cette reconstruction est impossible, nous proposons également des algorithmes per-

mettant de coder explicitement la connectivité d'un modèle en plus de sa géométrie. Dans tous les cas, nous nous sommes fixé comme contrainte de comprimer de manière *progressive* et *sans perte* d'information une classe de structures géométriques la plus large possible (incluant notamment les surfaces *non manifold*).

1.2 Présentation des structures géométriques

1.2.1 Géométrie, topologie et attributs

Il nous faut tout d'abord définir ce que nous regroupons dans ce mémoire 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 dimension supérieure 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 dimension 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

Les définitions qui suivent concernent le cas particulier de structures géométriques décrivant une surface plongée dans l'espace euclidien à 3 dimensions. Nous nous sommes limités à quelques notions fondamentales utiles à la compréhension de la suite du mémoire. Pour plus de détails, le lecteur est invité à se reporter par exemple à l'ouvrage *Differential Geometry of Curves and Surfaces* de Do Carmo [19].

Définition 1.2.1 *Une structure géométrique constituée d'un graphe 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 1.2.2 *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.*

Définition 1.2.3 *Etant donnée une surface triangulée S constituée de s sommets, a arêtes et f faces, on appelle caractéristique d'Euler-Poincaré de S le nombre $\chi(S) = s - a + f$.*

Définition 1.2.4 *On appelle genre d'une surface S le nombre $g = \frac{2-\chi(S)}{2}$. Ce nombre est indépendant de la manière dont S est triangulée, et correspond au nombre d'anses de S (0 pour une surface homéomorphe à une sphère, 1 pour une surface homéomorphe à un tore, ...).*

Remarque 1.2.5 *Cette dernière définition est souvent formulée comme suit : toute triangulation d'une surface S de genre g vérifie la relation d'Euler :*

$$s - a + f = 2 - 2\,g$$

Pour la preuve de cette relation, le lecteur pourra aussi se reporter aux travaux de Boissonnat et Yvinec [16, 17].

Définition 1.2.6 *Un maillage M est dit* manifold *(ou* variété*) 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 1.2.7 *Un maillage est dit* manifold avec bords *s'il vérifie les propriétés d'un maillage* manifold *partout sauf pour un ou plusieurs cycles disjoints d'arêtes incidentes à une seule face chacune. De telles arêtes sont appelées arêtes de bord. Les sommets de bord (i.e. les extrémités des arêtes de bord) ont un voisinage homéomorphe à un demi-disque.*

Définition 1.2.8 *Un maillage est dit* simple *s'il est* manifold *et de genre nul, c'est-à-dire topologiquement équivalent à une sphère.*

1.2.3 Structures géométriques étudiées

Les structures géométriques les plus fréquemment rencontrées en pratique, et que nous étudierons dans la suite de ce mémoire, sont les suivantes :

Les ensembles de points

Il s'agit des structures les plus simples puisqu'elles ne contiennent aucune information topologique. On manipule ces ensembles en modélisation tridimensionnelle et en reconstruction de formes (les nuages de points non organisés sont obtenus par un scanner laser qui relève des points sur la surface de l'objet). Les ensembles de points bruts sont également utilisés dans des domaines aussi divers que la géographie, la géologie ou l'imagerie médicale.

Les triangulations bidimensionnelles

Ce type de structure est obtenu lorsque tous les points d'un maillage sont coplanaires, et que toutes ses faces sont triangulaires. Les triangulations bidimensionnelles sont utilisées en particulier pour la modélisation des terrains, le codage d'images bitmap, et plus généralement pour la modélisation d'objets bidimensionnels ou projetables en 2 dimensions (cf chapitre 4).

Les maillages surfaciques

Pour coder informatiquement les objets volumiques qui nous entourent, on se limite le plus souvent à leur surface. En effet, lorsque l'application

finale est liée à la visualisation, il est inutile de considérer l'intérieur d'un objet opaque. Ainsi, on assimilera un objet tridimensionnel à un ensemble de variétés différentielles bidimensionnelles plongées dans l'espace euclidien à 3 dimensions. Comme pour tout objet destiné à être manipulé par un ordinateur, une phase de discrétisation est nécessaire. Les maillages surfaciques sont donc obtenus par un découpage de ces surfaces en mailles élémentaires. En fonction de la nature de ces mailles, on distinguera les maillages polygonaux quelconques (cf section 5.1) des maillages triangulaires (cf section 5.2). Ces derniers se sont rapidement imposés dans la pratique par leur simplicité d'utilisation (coplanarité, interpolation linéaire directe, etc.) et leur adéquation au matériel informatique (les cartes graphiques accélératrices actuelles ne gèrent que des triangles).

Les maillages tétraédriques

Lorsque l'information volumique de l'objet tridimensionnel à modéliser est pertinente, les maillages surfaciques ne sont plus suffisants, et on les généralise en 3 dimensions aux maillages volumiques. Cette fois, l'objet est découpé en éléments de volume (polyèdres). Le maillage tétraédrique correspond au cas particulier où tous les éléments de volume sont des tétraèdres. Ce sont les maillages volumiques les plus utilisés actuellement en visualisation de volumes par des grilles irrégulières ou dans les applications du calcul scientifique sur les éléments finis (cf chapitre 6).

1.2.4 Le codage naïf

La manière la plus directe de coder les positions d'une structure géométrique tridimensionnelle consiste à représenter chaque point par ses 3 coordonnées cartésiennes. Le code associé à la partie géométrique P de la structure sera donc de la forme :

$$\begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \vdots & & \\ x_{n-1} & y_{n-1} & z_{n-1} \end{bmatrix}$$

Pour le codage de la topologie, on décrit les polytopes composant la structure géométrique par un code identifiant la nature du polytope et une liste

d'indices de points de P correspondant aux sommets du polytope (lorsque tous les éléments de la structure sont de même nature, le code identifiant n'est pas répété). Par exemple, le code associé à la connectivité d'un maillage surfacique triangulaire contenant m triangles se présentera sous la forme suivante (les t_i^j étant compris entre 0 et $n-1$) :

$$
\begin{array}{l}
3 \\
\left[\begin{array}{ccc}
t_0^0 & t_0^1 & t_0^2 \\
t_1^0 & t_1^1 & t_1^2 \\
\vdots & & \\
t_{m-1}^0 & t_{m-1}^1 & t_{m-1}^2
\end{array}\right]
\end{array}
$$

Ce type de codage constitue la base du standard VRML [47, 18], capable de gérer également les attributs additionnels associés aux points ou aux faces du maillage. Ce format présente l'avantage d'être lisible et très général, mais il est aussi très redondant puisque chaque point de P est référencé autant de fois qu'il a de polytopes incidents. En outre, le format VRML est un format texte (ou *ASCII*), donc plus volumineux qu'un format brut où les coordonnées des sommets sont généralement codées par des nombres flottants sur 32 bits, et les indices de sommets (pour la description de la connectivité) par des entiers sur 32 bits. L'objet de la compression géométrique sans perte est de réduire la redondance présente dans un tel codage en organisant l'information de façon optimale.

1.3 Codage entropique

La compression de données standard propose de nombreuses méthodes pour réduire la redondance présente dans un ensemble E de données informatiques arbitraires. Ces méthodes permettent de s'approcher de la limite minimale théorique — au sens de la théorie de l'information (cf section 1.3.2, paragraphe *Prédiction*) — du nombre d'octets nécessaires au codage de l'information exacte (codage sans perte) contenue dans E. Fournissant des résultats différents suivant le type de données auxquelles ils sont appliqués, ces algorithmes sont utilisées conjointement par la plupart des codeurs entropiques standards, comme *zip*, *gzip* ou *compress*.

1.3.1 Méthodes de substitution

Il existe deux principaux algorithmes de compression par substitution. Le premier, développé par Ziv et Lempel en 1977 [112], gère un dictionnaire adaptatif implicite en cherchant dans un tampon contenant les derniers symboles codés des correspondances exactes avec le symbole courant. Ces correspondances sont ensuite codées par leur position dans le tampon et leur longueur.

Le second, développé par les mêmes auteurs en 1978 [113], construit un dictionnaire explicite de manière dynamique. L'algorithme maintient un préfixe (initialement vide) auquel on ajoute le symbole courant tant que le mot formé du préfixe et du symbole courant est présent dans le dictionnaire. Lorsque cette condition n'est plus vérifiée, le nouveau mot (préfixe courant + symbole courant) est inséré dans le dictionnaire, et le préfixe courant est codé par une référence au dictionnaire. Cette méthode atteint des taux de compression équivalents a ceux de la méthode implicite, tout en remplaçant la coûteuse recherche de motif dans le tampon par une recherche efficace dans une structure de dictionnaire adaptée.

Ces algorithmes, destinés originellement à des données textuelles, se prêtent moins bien à l'analyse et la compression de données multidimensionnelles.

1.3.2 Méthodes statistiques

Codage de Huffman

Le codage de Huffman, qui date des années 50 [52], repose sur une analyse statistique préalable des données à comprimer. A l'issue de cette analyse, un arbre est construit qui permet d'attribuer à chaque symbole un code dont le nombre de bits est inversement proportionnel à la probabilité d'apparition du symbole. Ainsi, les symboles les plus fréquents se voient affecter des codes plus courts, et les codes les plus longs sont attribués aux symboles rares. En outre, ces codes sont séparables, c'est-à-dire qu'un code donné ne peut pas être le préfixe d'un autre code. Bien sûr, pour permettre au décodeur de reconnaître les symboles, il est nécessaire de lui transmettre le dictionnaire obtenu après la phase d'analyse statistique des données.

Cette méthode a ensuite été raffinée pour permettre une analyse statistique *dynamique* des données, qui évite la transmission du dictionnaire ainsi que l'analyse préalable des données, au prix d'une augmentation de la complexité provenant de la reconstruction de l'arbre de Huffman après chaque mise à jour du modèle statistique.

Codage arithmétique

La méthode de codage entropique que nous avons choisi d'utiliser ici est le codage arithmétique. Développé dans les années 80 [88, 87, 110], 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 considérable sur la méthode 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. Comme pour le codage de Huffman, cette modélisation peut être statique ou dynamique, mais dans le cas d'une modélisation dynamique, la complexité du codage arithmétique est meilleure puisqu'il n'y a pas d'arbre à reconstruire après la mise à jour du modèle statistique.

La figure 1.4 illustre le fonctionnement du codage arithmétique sur un exemple. Soit à coder le message : "UN EXEMPLE". L'algorithme subdivise l'intervalle [0,1[en fonction de la probabilité de chaque symbole de l'alphabet (un ordre arbitraire sur les symboles est fixé) : (espace,[0;0,1[), (E,[0,1;0,4[), (L,[0,4;0,5[), (M,[0,5;0,6[), (N,[0,6;0,7[), (P,[0,7;0,8[), (U,[0,8;0,9[), (X,[0,9;1[). Dans cet exemple, la modélisation est statique (la probabilité des symboles est déterminée par une passe sur les données avant le codage et doit donc être transmise au décodeur ; le coût supplémentaire engendré dépend de la taille de l'alphabet).

C'est l'utilisation de cette méthode de codage qui nous permet, dans la suite de ce mémoire, de nous affranchir des parties entières supérieures appliquées aux logarithmes dans les analyses de coût théorique. Pour alléger

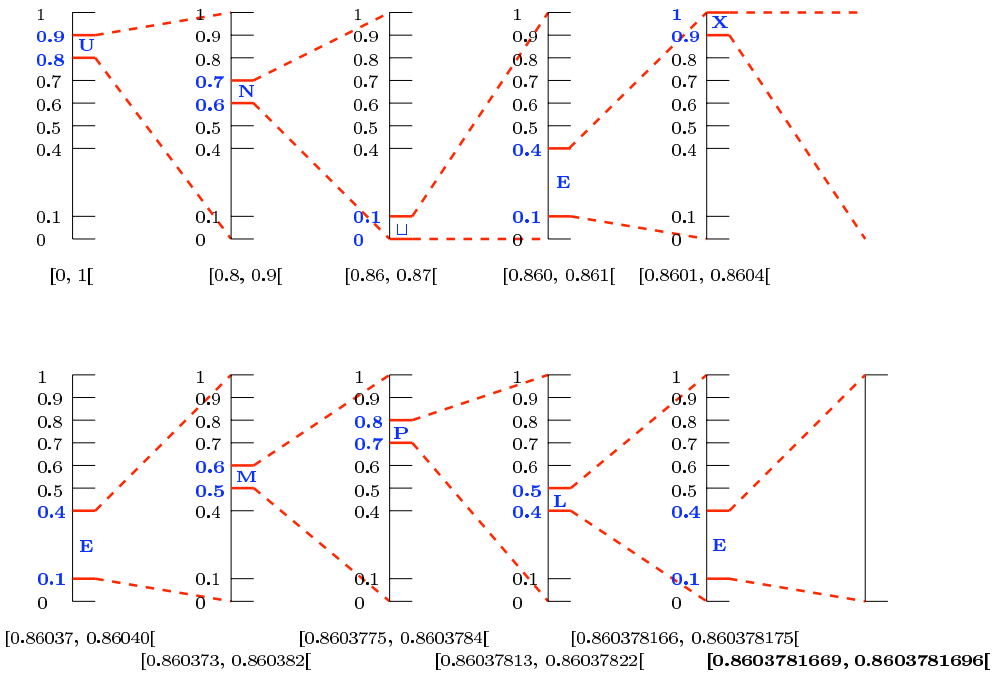


FIG. 1.4 – *Exemple de codage arithmétique*

les notations, nous avons choisi de ne pas faire figurer non plus les quantités ϵ induites par le codeur arithmétique et négligeables devant $\log_2 \frac{1}{P}$.

Prédiction

Nous avons écrit plus haut que ces méthodes de codage permettaient de s'approcher de la limite théorique du volume de données nécessaire au codage sans perte de l'information contenue dans un fichier initial donné. Cependant, cette limite théorique, appelée entropie en théorie de l'information [94], dépend d'un paramètre d'ordre. A l'ordre 0, l'entropie e d'un ensemble E de n symboles s_i de probabilité p_i ($i \in \{0,...,n-1\}$) est :

$$\sum_{i=0}^{n-1} p_i \log_2 \frac{1}{p_i}$$

L'ordre 0 signifie que les probabilités sont obtenues sans tenir compte de la corrélation entre les symboles : la probabilité d'apparition du $i$ème symbole de la chaîne à coder est considérée indépendante de la valeur des symboles précédents.

Dans la pratique, il existe au contraire des corrélations plus ou moins fortes entre un symbole donné et son voisinage. C'est en tenant compte de ces corrélations que l'on parvient à réduire la taille du code final. C'est pourquoi la plupart des algorithmes de compression reposent sur le principe de prédiction. Au moment de coder le symbole s courant, l'algorithme analyse son voisinage (seul le voisinage causal est examiné, i.e. les symboles qui précèdent s, puisque le décodeur doit pouvoir faire la même analyse), et estime les probabilités $p(s = v_i)\,(i \in \{0,...,m-1\})$ pour chacune des m valeurs de s possibles. La taille du voisinage analysé correspond à l'ordre du modeleur statistique. Ainsi, un codeur arithmétique adaptatif d'ordre 3 calculera les probabilités d'occurrence des différentes valeurs possibles pour le symbole courant à partir de l'analyse des 3 derniers symboles de la séquence codante.

1.4 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 32 bits (dont 24 bits pour la mantisse en notation normalisée [41, 1]) pour chaque coordonnée. A 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, 24 bits s'avèrent généralement suffisants). Cependant, les méthodes décrites dans le chapitre 2 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 polyédrique 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 le chapitre 2.

1.5 Résidu 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 après quantification et V_c le volume des données comprimées, on peut choisir d'utiliser un facteur de compression, qui exprime combien de fois les données comprimées 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 mémoire, nous utiliserons le résidu de compression

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

qui exprime le volume des données comprimées en pourcentage du volume des données initiales. *Un algorithme sera donc d'autant plus efficace que le résidu de compression associé sera 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 triangulaires, la relation

d'Euler (cf section 1.2.2) permet de passer facilement d'une unité à l'autre : 1 bps $\approx 1/2$ bpf $\approx 1/3$ bpa. Dans ce mémoire, nous exprimerons la taille d'un objet géométrique en bits par sommet.

1.6 Présentation du document

Nous rappelons que ce travail est consacré à l'étude de la *compression* de structures géométriques de manière *progressive* — c'est-à-dire la représentation *multirésolution* et *compacte* de ces structures — et *sans perte* d'information : au niveau de détail le plus fin, le décodeur doit reproduire le modèle original.

Dans la suite de ce mémoire, nous abordons ce problème de manière différente de ce qui a été fait jusqu'à maintenant. Nous utilisons le fait que dans de nombreux cas, les objets tridimensionnels sont construits automatiquement à partir d'échantillons de points. Ainsi, la connectivité d'un maillage pourra souvent être *reconstruite* à partir de ses sommets.

Par conséquent, contrairement au principe général qui gouverne l'ensemble des méthodes de compression précédemment publiées et qui font l'objet du chapitre 2, l'algorithme principal que nous proposons au chapitre 3 donne la priorité à la compression de la géométrie du modèle. Il exploite l'ordre de transmission des sommets pour coder uniquement leur position. Pour un ensemble de n points, un gain minimal de $n \log n$ bits est démontré, et des résidus de compression équivalents (ou meilleurs, lorsque la distribution de points est non uniforme) à ceux obtenus pour les positions par les méthodes actuelles les plus efficaces, sont atteints *sans aucune connaissance topologique*. La méthode est destinée a priori aux structures géométriques à base de points non connectés en *dimension quelconque*, ou bien aux objets dont la topologie peut être reconstruite canoniquement, étudiés au début du chapitre 4.

Cependant, la nécessité d'un codeur topologique explicite se fait très vite ressentir, au moins pour aider l'algorithme de reconstruction automatique à restaurer la connectivité originale sans perte, en lui transmettant un *sous-ensemble* des données topologiques. Après une étude théorique sur l'optimalité d'un tel sous-ensemble dans le cadre de la triangulation de Delaunay contrainte, nous proposons au chapitre 4 un codeur topologique progressif adapté à l'algorithme principal de compression des positions.

Au chapitre 5, un second codeur topologique est introduit. Basé sur deux opérations classiques de simplification de maillage (la contraction d'arête et la fusion de sommet), il est adapté aux maillages surfaciques triangulaires, et plus généralement à tout complexe simplicial bidimensionnel. Couplé au codeur de positions décrit dans le chapitre 3, il permet d'atteindre des résidus de compression très compétitifs par rapport aux précédentes méthodes progressives, sur des modèles dont la *complexité topologique* est *arbitraire*. Ce codeur est généralisé aux maillages volumiques de type tétraédrique dans le chapitre 6 ; là encore, les résidus obtenus améliorent les résultats des méthodes précédemment publiées en compression progressive.

Pour conclure, nous présentons un bref tour d'horizon des contributions apportées par ce travail dans le domaine de la compression géométrique, et proposons quelques pistes pour de futurs développements.

Chapitre 2

Etat de l'art

Nous nous intéressons dans ce mémoire à la compression de structures géométriques de manière *progressive* et *sans perte* d'information. Ce domaine de l'informatique théorique situé entre la géométrie algorithmique et la compression de données standard est en plein essor depuis environ cinq ans. (Même si le problème théorique du codage compact de graphes planaires, où la compression géométrique trouve aussi ses racines, a été étudié de manière approfondie depuis plus longtemps [108, 109, 57, 107, 79, 59].)

Dans ce chapitre, nous proposons une classification des différents résultats obtenus dans le domaine depuis 1995. Afin de donner au lecteur une vision d'ensemble de ces travaux, nous ne décrivons pas le détail des algorithmes cités. En revanche, dans les sections 4.1, 5.1.1, 5.2.1, 6.1, nous reviendrons sur les principales méthodes évoquées ici, en détaillant davantage les algorithmes utilisés, leurs avantages et leurs limites.

En toute logique, les travaux sur la compression *non progressive* de structures géométriques (maillages surfaciques triangulaires ou polygonaux, maillages tétraédriques) auraient dû trouver leur place dans ce mémoire à la section 2.5 concernant les travaux connexes. Cependant, leur parenté directe avec la plupart des méthodes progressives (en particulier, de nombreuses méthodes de compression progressive constituent une extension de méthodes non progressives), nous a conduit à leur accorder une place importante dans cet état de l'art.

Dans notre revue 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 1.4). 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 aux chapitres 5 et 6.

Enfin, à propos des résultats affichés par les différents algorithmes de compression géométrique, une dernière remarque générale doit être faite. Puisque l'on distingue, dans une structure géométrique, la topologie (ou connectivité), de la géométrie proprement dite (c'est-à-dire la position des sommets), il est d'usage de présenter les résultats d'un algorithme en séparant le coût de la topologie du coût de la géométrie (lorsque la méthode proposée comprime les deux types de données). Cependant, il est important de comprendre que ces deux coûts sont presque toujours indissociables, dans la mesure où le codage de la géométrie est largement dépendant du codage de la topologie.

2.1 Triangulations bidimensionnelles

Le problème de la visualisation d'un modèle de terrain triangulé (ou *TIN*, pour *Triangulated Irregular Network*, dans la littérature anglo-saxonne) à différents niveaux de détails a d'abord été étudié hors du cadre de la compression [37]. On notera en particulier l'algorithme de de Berg et Dobrindt qui utilise les résultats de Kirkpatrick [66] pour construire une hiérarchie où les différents niveaux de détail, obtenus par la triangulation de Delaunay (cf section 4.2), peuvent être combinés [25].

Un autre problème classique des *SIG* (Systèmes d'Information Géographique) est la transmission efficace de triangulations de Delaunay d'un ensemble de points bidimensionnel. Cependant, les travaux de Snoeyink et van Kreveld [95], et plus récemment de Sohler [96], ont un objectif différent du nôtre : l'ordre de transmission sur les points est utilisé pour accélérer la reconstruction de la triangulation (un temps linéaire est obtenu à la place de $O(n \log n)$). Accessoirement, un gain est réalisé en codant les coordonnées des points de manière différentielle avec des codes de longueur variable. Toutefois, cela ne constitue pas le principal intérêt de ces méthodes, les facteurs de compression obtenus restant relativement faibles (cf section 4.1).

Kim, Park, Jung et Cho ont développé un algorithme de compression adapté aux modèles de terrain qui repose sur le codage des arêtes du modèle qui ne sont pas de Delaunay (cf section 4.3), et utilise la triangulation de Delaunay contrainte incrémentale [62] pour reconstruire la topologie originale complète. Cependant, si la méthode conduit à des résultats pratiques spectaculaires pour le codage de la connectivité, elle ne propose aucun schéma de compression pour la géométrie du maillage (cf section 4.1).

Or, l'intérêt d'une méthode de codage de la connectivité d'une triangulation sans compression sur ses positions est assez limité. En effet, Denny et Sohler ont montré qu'il était possible de coder une triangulation planaire par une simple permutation de son ensemble de points support [28]. Autrement dit, on sait coder la connectivité de toute triangulation bidimensionnelle avec un coût *nul*, en utilisant uniquement l'ordre d'énumération de ses sommets. Par conséquent, l'enjeu principal de la compression de triangulation planaire (et, comme nous le verrons dans la suite, de toute structure géométrique d'une manière générale) est de combiner un codage efficace de la connectivité et de la géométrie (cf section 4.1).

2.2 Maillages surfaciques triangulaires

Ce cas particulier des maillages surfaciques polygonaux plongés en dimension 3 a été beaucoup plus étudié que le cas général, exposé dans la section suivante. On distingue les méthodes de compression monorésolution, des méthodes progressives (ou multirésolution) permettant de visualiser l'objet dans des versions préliminaires de plus en plus raffinées au fur et à mesure de la décompression.

2.2.1 Compression monorésolution

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. A 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 et de coder uniquement le résidu (i.e. l'erreur entre la position prédite et la position réelle) par codage entropique. Le plus souvent, le prédicteur P utilisé s'exprime comme une combinaison linéaire des n sommets précédemment codés :

$$P(\mathbf{C},s_{i-1},...,s_{i-n}) = \sum_{k=1}^{n} C_k\, s_{i-k}, \quad avec\ \mathbf{C} = (C_1,...,C_n)$$

Les trois principaux prédicteurs utilisés sont le codage différentiel ($n = 1$, $\mathbf{C} = (1)$), le prédicteur linéaire direct ($n = 2$, $\mathbf{C} = (2, -1)$) et le prédicteur de type parallélogramme ($n = 3$, $\mathbf{C} = (1,1, -1)$). La figure 2.1 illustre ces techniques de prédiction géométrique.

Un premier groupe de méthodes consacrées à la compression de maillages triangulaires s'est formé autour du principe de décomposition du maillage en bandes de triangles [26, 35, 12, 22, 27, 111]. Les bandes de triangles sont des représentations largement utilisées par les bibliothèques graphiques comme *OpenGL* [80, 93], et prises en charge par la plupart des accélérateurs matériels. Elles permettent de réduire le nombre de références à un même sommet, en connectant systématiquement le sommet courant aux deux sommets précédents, temporairement stockés dans un tampon. La bande est ainsi construite en zigzag : chaque nouveau sommet crée un triangle adjacent au triangle précédent, alternativement par la droite et par la gauche. Une version généralisée des bandes de triangles permet de lever cette contrainte d'alternance en précisant par un bit supplémentaire si le triangle créé par le sommet courant est adjacent au triangle précédent par la droite ou par la gauche (cf figure

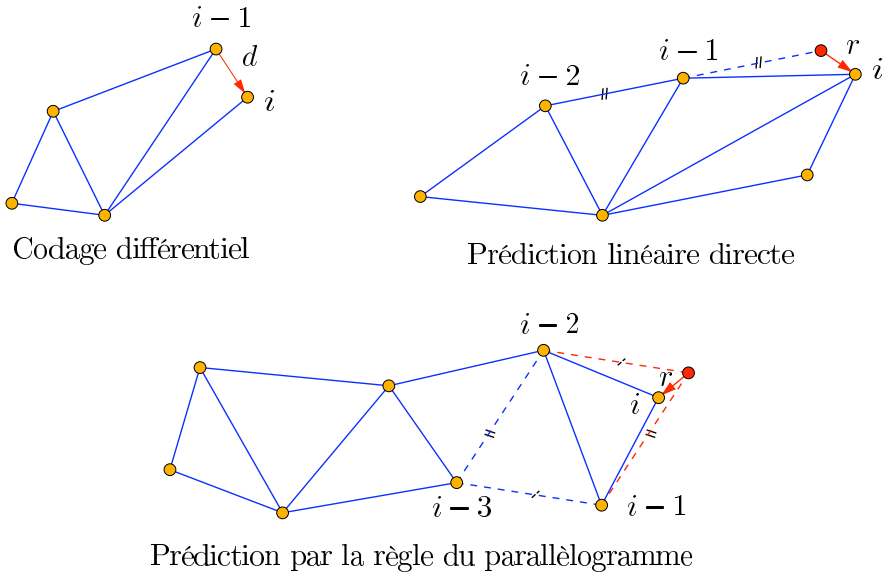


FIG. 2.1 – *Techniques classiques pour le codage des positions*

2.2). Le problème de ces méthodes est que plusieurs bandes sont nécessaires pour coder un maillage complet, et les sommets partagés par deux bandes doivent être dupliqués. De plus, Evans, Skiena et Varshney ont montré que le problème du partitionnement d'un maillage triangulaire en un nombre minimum de bandes est NP-complet [35]. La conséquence est qu'en moyenne, chaque sommet est référencé deux fois dans la séquence finale codant la topologie du maillage. Pour référencer les sommets déjà codés de manière plus économique, un tampon stockant les k derniers sommets codés est maintenu. Bar-Yehuda et Gotsman ont montré que pour un maillage contenant n sommets, la taille optimale de ce tampon est $12{,}72\sqrt{n}$ [12]. Pour la compression de la géométrie, ces méthodes utilisent une description différentielle des positions, et des codes de longueur variable pour le codage entropique. Concernant le codage de la connectivité, c'est l'algorithme de Deering [26] (cf section 5.2.1), exploité aussi par Chow [22], qui fournit la représentation la plus compacte, avec $7{,}5 + \log_2 n/8$ bits par sommet.

La deuxième famille de méthodes de compression de maillages triangulaires regroupe des algorithmes présentant de fortes analogies dans leur manière de construire la séquence de sommets décrivant la connectivité du maillage. Le parcours entrelacé des deux arbres couvrants (sommets et triangles) commence par une face ou une arête, et procède par conquête. La position de chaque sommet de la séquence est accompagnée d'une informa-

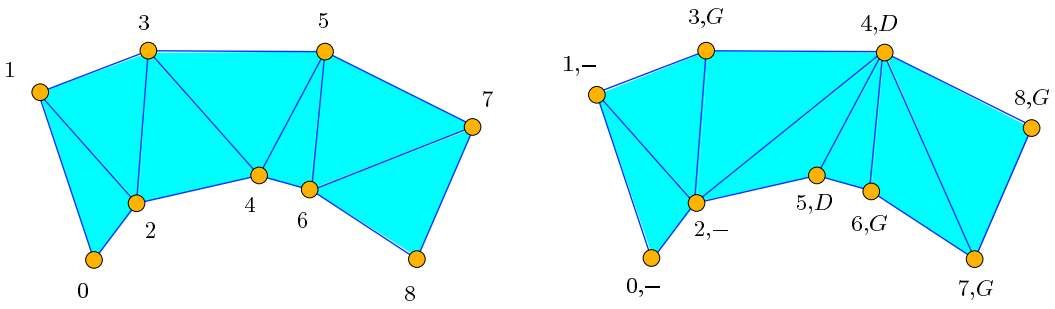


FIG. 2.2 – *Bandes de triangles standard et généralisée*

tion supplémentaire permettant de reconstruire la topologie au voisinage de ce sommet. Cette information consiste généralement en une étiquette (ou commande) indiquant au décodeur comment connecter le sommet à ses voisins. Ainsi, les algorithmes de Taubin et Rossignac [104] (cf section 5.2.1), Gumhold et Strasser [45], Touma et Gotsman [106], De Floriani, Magillo et Puppo [36], Rossignac, Szymczak et King [89, 91, 63, 98], Isenburg et Snoeyink [54, 53], parcourent les sommets du maillage suivant des arbres couvrants spiralés très proches les uns des autres. La méthode de Touma et Gotsman se distingue des autres par la nature du code associé à la position de chaque sommet dans la séquence : le codage des *degrés* des sommets permet de limiter à 2 le nombre de cas (et donc le nombre de commandes), et d'obtenir ainsi d'excellents résidus de compression pour la connectivité des maillages très réguliers (1,4 bits par sommet en moyenne sur des modèles usuels) (cf section 5.2.1). En revanche, ces résultats se dégradent pour des maillages irréguliers, et ne sont pas bornés dans le cas le pire. Seuls King et Rossignac [63] proposent un algorithme dont les performances sont comparables en moyenne, et garantissant de surcroît une borne théorique de 3,67 bits par sommets dans le cas le pire. Ce faisant, ils améliorent les résultats obtenus dans le domaine de la compression de graphe par Keeler et Westbrook (4,6 bits par sommet) [59] et se rapprochent de la borne inférieure théorique de 3,24 bits par sommets, obtenue par Tutte [108] en énumérant les différents maillages constructibles à partir d'un ensemble de points (cf section 5.2.1). La compression de la géométrie, qui n'est d'ailleurs traitée que par une partie de ces méthodes, utilise le codage différentiel, la prédiction (linéaire ou d'ordre élevé), et l'estimation du *pli* entre deux triangles adjacents.

L'originalité de l'approche de Li et Kuo [72] consiste à coder la topologie par le parcours du graphe *dual* de la triangulation (cf figure 2.3). Ce graphe

est décrit de manière compacte (3 bits par sommet sur des maillages usuels) par un arbre binaire dégradé.

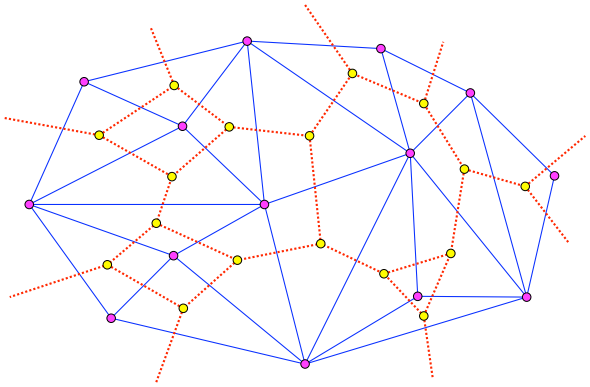


FIG. 2.3 – *Une triangulation et son graphe dual*

Enfin, Bajaj, Cutchin, Pascucci et Zhuang [8, 11] se démarquent des méthodes précédentes en proposant un partitionnement du maillage en couches de sommets et de triangles. Grâce à un codage indépendant des différentes couches, leur technique s'avère particulièrement utile pour la transmission robuste de l'objet à travers un réseau. En outre, l'algorithme s'adapte relativement bien au cas de maillages non *manifold*, ce qui n'est pas le cas des méthodes précédentes : conçus à l'origine pour des maillages *simples* (cf section 1.2.2), leurs performances se dégradent considérablement lorsqu'il s'agit de coder des maillages présentant une topologie plus complexe.

Récemment, Alliez et Desbrun [5] ont proposé un ensemble d'heuristiques de conquête permettant de raffiner l'algorithme de Touma et Gotsman [106]. Grâce à ces optimisations, les auteurs améliorent non seulement les résultats pratiques sur les maillages usuels, mais prouvent également une borne supérieure de 3,5 bits par sommet pour des maillages arbitraires, se positionnant ainsi de manière très favorable au sein des méthodes publiées jusqu'alors (cf 5.2.1).

2.2.2 Compression progressive

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 données à tout moment pour créer la meilleure approximation possible du modèle original. La difficulté est que l'organisation

hiérarchique de la structure géométrique, indispensable pour obtenir 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 nécessitaient un nombre important de bits par sommet (ces algorithmes sont évoqués dans la section 2.5).

Par la suite, plusieurs méthodes de compression monorésolution ont pu être étendues à la compression progressive. Les techniques les plus récentes, en regroupant les opérations de raffinement en paquets, permettent d'obtenir des résidus de compression proches de ceux des techniques non progressives (la méthode de Pajarola et Rossignac [82] atteint moins de 8 bits par sommet en moyenne (cf section 5.2.1)). Ainsi, Taubin, Guéziec, Horn et Lazarus ont proposé une extension progressive [102] (cf section 5.2.1) aux travaux originaux de Taubin et Rossignac [104], et Bajaj, Pascucci et Zhuang ont adapté leur algorithme de décomposition en couches [11] à la transmission progressive [10] (cf section 5.2.1).

Les travaux de Cohen-Or, Levin et Remez [23] combinent des techniques de simplification séquentielle par suppression de sommet pour la connectivité (cf section 2.5.1) 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 d'excellents résidus de compression pour le codage progressif de la connectivité (environ 6 bits par sommet en moyenne) (cf section 5.2.1).

La méthode décrite par Li et Kuo [73] 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 (cf section 5.2.1). 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 [64].

Alliez et Laurent [6, 3] proposent une méthode hybride qui comprime la topologie de l'objet en monorésolution, 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 [106] 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

plans de bits à des nombres flottants pour obtenir l'échelonnabilité : le bit de poids le plus fort est transmis *pour chaque point*, puis le bit suivant, et ainsi de suite jusqu'au bit de poids le plus faible (cf section 5.2.1).

Tout récemment, Alliez et Desbrun [4] ont développé un algorithme dont les performances semblent être actuellement les plus compétitives (moins de 3,7 bits par sommet en moyenne). Comme Cohen-Or, Levin et Remez [23], les auteurs procèdent par vagues de suppressions de sommets indépendants. Les trous polygonaux ainsi engendrés sont retriangulés de manière déterministe à l'aide de marqueurs positionnés sur les sommets du bord. Ces phases de décimation et retriangulation, auxquelles s'ajoute une phase de régularisation, sont gouvernées par la contrainte de maintenir les degrés des sommets du nouveau maillage autour de la valeur moyenne 6. Cela garantit la régularité des maillages intermédiaires, minimise les changements de topologie, et réduit l'entropie du signal. En ce qui concerne la compression des positions, la décomposition de l'information en composantes normale et tangentielle améliore sensiblement les résultats obtenus par les approches précédentes (cf section 5.2.1).

Remarque 2.2.1 *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 résidus de compression sont atteints pour des maillages très réguliers, qui optimisent la prédiction géométrique et favorisent le codage entropique de la description topologique.*

2.3 Maillages surfaciques polygonaux

Plusieurs méthodes de compression topologique de maillages triangulaires ont pu être étendues au cas plus général des maillages polygonaux arbitraires, au prix d'une triangulation préalable du maillage, d'un code additionnel pour le marquage des arêtes intérieures ajoutées, et de la suppression finale de ces arêtes après la décompression [9, 103, 43].

A notre connaissance, il existe peu de travaux directement consacrés à la compression topologique de maillages polygonaux non triangulaires.

King, Szymczak et Rossignac ont montré les premiers que la connectivité des maillages à base de quadrilatères pouvait être comprimée plus efficacement que celle de leurs équivalents triangulés. Ils prouvent une borne théorique de 3 bits par sommet dans le cas le pire, pour les maillages simples et composés exclusivement de quadrilatères [65] (cf section 5.1.1). Kronrod et Gotsman obtiennent des résultats similaires (environ 3,5 bits par sommet) par une méthode analogue [69]. Bien qu'une généralisation à des polygones de taille supérieure soit évoquée dans ces deux papiers, seul le cas des triangles et des quadrilatères y est effectivement traité. Ce sont les récents travaux de Isenburg et Snoeyink qui élargissent finalement le domaine de la compression topologique à des maillages polygonaux complètement arbitraires [55] (cf section 5.1.1).

Ces trois méthodes s'inspirent plus ou moins directement du principe de l'algorithme *edgebreaker* de Rossignac initialement développé pour les maillages triangulaires [89]. Leur principe général reste donc de décrire l'arbre couvrant des sommets du maillage et celui de ses faces en énumérant les sommets (accompagnés de codes d'assemblage compacts) dans un ordre déterminé. Ces méthodes ne proposent pas de schéma explicite de compression de la géométrie des modèles. Les techniques de prédiction habituellement utilisées sur les maillages triangulaires sont évoquées, mais aucun résultat expérimental n'est proposé. On notera en outre que ces algorithmes opèrent une compression monorésolution des objets géométriques, et donc ne permettent pas leur visualisation progressive au cours de la décompression. Enfin, comme pour le cas particulier des faces triangulaires, les méthodes citées sont d'autant plus efficaces que le maillage original est régulier, et voient leurs performances réduites de manière sensible sur des maillages non *manifold* (cf remarque 2.2.1).

2.4 Maillages tétraédriques

2.4.1 Compression monorésolution

Comme pour le cas des maillages polygonaux, les méthodes de compression de maillages tétraédriques dérivent généralement de méthodes initialement conçues pour les maillages triangulaires surfaciques.

L'algorithme *grow&fold* de Szymczak et Rossignac [99, 100] peut se dé-

composer en deux phases : la conquête, à partir d'un tétraèdre arbitraire, de l'ensemble des tétraèdres du maillage sous forme d'arbre couvrant, puis une suite de pliages et de collages pour reconstruire les relations d'adjacence entre les tétraèdres de l'arbre. La méthode produit des résultats variant autour de 44 bits par sommet en moyenne (cf section 6.1).

Gumhold, Guthe et Strasser ont proposé quant à eux une généralisation directe de leur algorithme de codage de maillages triangulaires surfaciques [45] au cas des maillages tétraédriques [44]. La méthode de conquête maintient une surface triangulée entre l'ensemble des tétraèdres déjà codés et le reste du maillage. Les tétraèdres extérieurs adjacents à la frontière sont ajoutés à la partie conquise par des opérations élémentaires de connexion. Cet algorithme réalise une nette progression en terme de résidus de compression puisqu'il permet d'obtenir des coûts inférieurs à 15 bits par sommet (cf section 6.1).

2.4.2 Compression progressive

A notre connaissance, le seul papier traitant de compression progressive de maillages tétraédriques est dû à Pajarola, Rossignac et Szymczak [84]. La méthode est fondée sur l'utilisation d'un opérateur de simplification en 3D équivalent à la contraction d'arête de Hoppe. Le maillage original est simplifié par vagues successives de contraction d'arêtes, et des résidus voisins de 45 bits par sommet sont obtenus pour des tétraédrisations de Delaunay de points aléatoires (cf section 6.1).

2.5 Travaux connexes

Outre les travaux cités dans la section précédente et qui rentrent dans le cadre que nous nous sommes fixé dans ce mémoire, la manipulation efficace de maillages surfaciques a donné lieu à de nombreuses recherches dans des domaines connexes. Ces recherches peuvent être regroupées sous le terme de *simplification* de surfaces polygonales. La motivation initiale était de réduire la complexité des modèles issus des procédés d'acquisition tridimensionnels. En effet, les triangulations ainsi obtenues forment une surface ayant une topologie arbitraire (composantes connexes, bords, genre) et une connectivité généralement irrégulière. La simplification a donc pour but de rendre ces

maillages complexes et volumineux plus faciles à stocker, à transmettre, et à visualiser (en particulier grâce à des niveaux de détails variables suivant la position de l'objet par rapport à la caméra).

L'aspect progressif de ces algorithmes de simplification les rattachent naturellement à notre champ d'étude. En revanche, il existe deux différences fondamentales qui les en écartent. La première est que les travaux de simplification de maillages ne débouchent 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). La seconde différence essentielle concerne plus particulièrement les méthodes reposant sur l'analyse multirésolution, dont l'une des caractéristiques est de paramétriser la surface représentée par le maillage original, ce qui implique la *perte* d'une partie de l'information de départ : ce n'est pas le maillage qui est codé, mais plutôt la surface qu'il représente.

Nous nous limiterons donc ici à un survol des principales méthodes publiées. Pour un état de l'art plus détaillé sur les méthodes de simplification et d'approximation géométrique, le lecteur pourra se reporter aux travaux de Heckbert, Garland, Rossignac, Hoppe, Schroeder, Soucy et Varshney [48, 49] ou de Luebke [76].

2.5.1 Simplification séquentielle

La technique commune aux différents algorithmes de simplification séquentielle de maillages est la *décimation*. On part du maillage original et on lui applique itérativement une opération élémentaire 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. 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 2.4) : 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 [92].

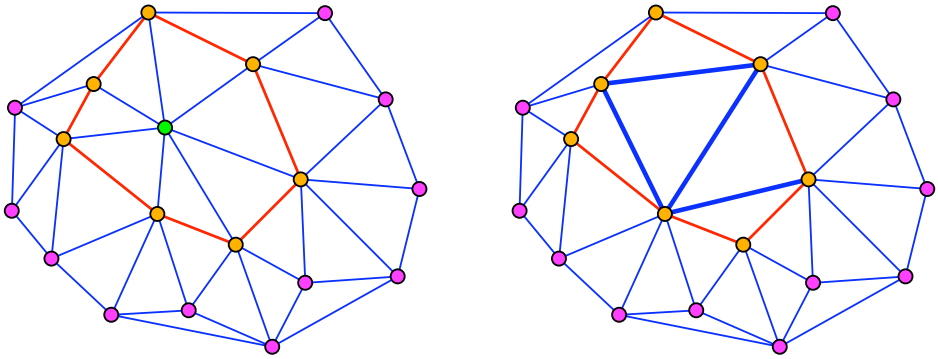


FIG. 2.4 – *La suppression de sommet*

– la contraction d'arête (cf figure 2.5) : 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 [51, 2, 74, 50]. La principale limite de ces méthodes est de traiter exclusivement le cas des triangulations bidimensionnelles *manifold* et orientables.

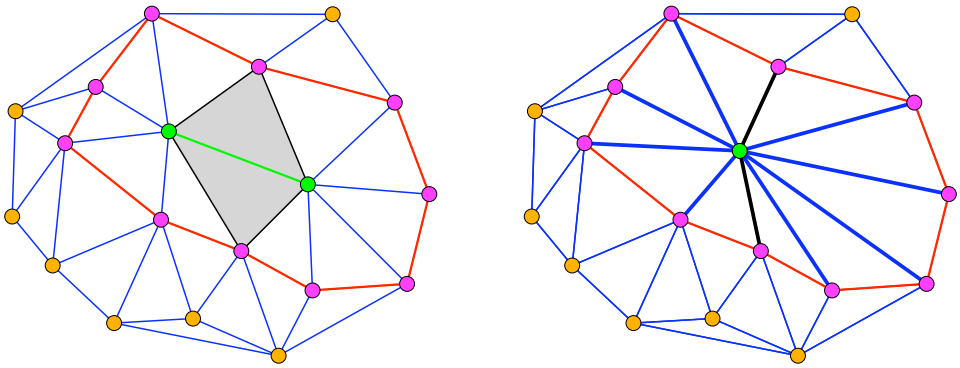


FIG. 2.5 – *La contraction d'arête*

– 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 [67].

– la fusion de sommets (cf figure 2.6) : 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 *manifold*, éventuellement non orientable), et même à tout complexe simplicial en dimension quelconque [85, 39, 38, 34].

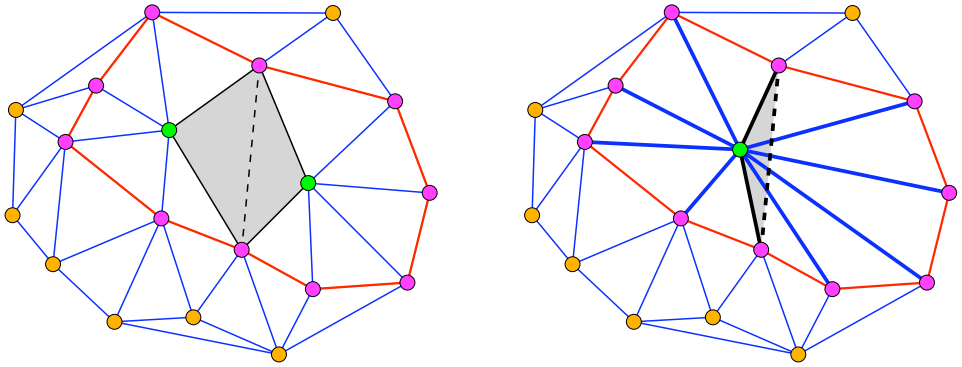


FIG. 2.6 – *La fusion de sommets*

– la contraction de face (cf figure 2.7) : 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) [40].

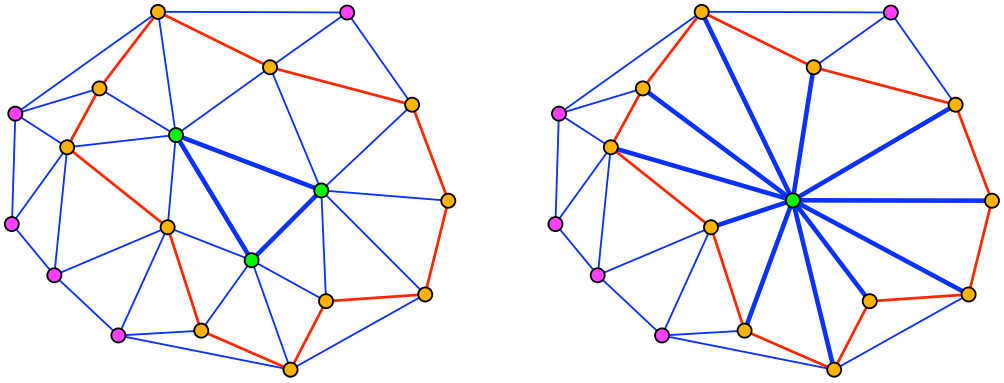


FIG. 2.7 – *La contraction de face*

Outre le type d'opération élémentaire utilisé, les méthodes de simplification séquentielles se distinguent entre elles par le choix du critère d'optimisation pour l'approximation de l'objet original. En effet, pour réduire la distorsion géométrique au minimum, une queue de priorité est maintenue dynamiquement sur les éléments du maillage auxquels l'opérateur est appliqué. Ainsi, 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. Les composantes des méthodes de simplification séquentielle se résument donc à un opérateur élémentaire, une métrique liée à la distorsion géométrique, et un critère de validité.

2.5.2 Analyse multirésolution

L'alternative classique à la simplification séquentielle de maillage est la paramétrisation de surface couplée à l'analyse multirésolution, dont l'outil privilégié est la décomposition en ondelettes. Le principe général des ondelettes est de redistribuer l'énergie du signal en le décomposant en deux types de fonctions de base : les fonctions dites *d'échelle*, obtenues par moyennage, et les fonctions *de détail*, dont les coefficients (décroissant au fur et à mesure de la décomposition) affinent progressivement le signal. Très utilisées en compression d'images bitmap, les ondelettes sont particulièrement efficaces pour décorréler les données originales, facilitant de ce fait un éventuel codage entropique [31, 24]. L'idée intuitive est que les niveaux de données grossiers fournissent d'excellents prédicteurs pour les niveaux plus fins, réduisant ainsi considérablement les résidus (i.e. les corrections de prédiction) à coder.

Le principe de la décomposition en ondelettes peut être appliqué de manière directe à des maillages surfaciques réguliers, c'est-à-dire issus de modèles échantillonnés uniformément [31, 42, 97], mais cette régularité est très contraignante en pratique. Ce sont les travaux de Lounsbery, DeRose et Warren [75] qui ont permis de traiter le cas plus général d'une topologie arbitraire. Les résultats obtenus ont ensuite été appliqués à l'approximation progressive de surfaces [20].

Une autre classe importante de méthodes d'analyse multirésolution est basée sur le principe de subdivision [114]. La principale limite de ces méthodes est d'imposer que le maillage original soit semi-régulier (i.e. tous les sommets du maillage sont de même degré), ce qui a conduit au développement de nombreux algorithmes de remaillage [33, 70, 68].

Guskov, Vidimce, Sweldens et Schroeder [46] introduisent la notion de *maillage normal*. Il s'agit d'une représentation multirésolution où chaque niveau de détail est déduit du niveau précédent par l'application d'une correction (ou raffinement) *scalaire*, localisée dans la direction normale à la surface. Par conséquent, le maillage est codé par un seul nombre décimal par sommet au lieu des 3 coordonnées spatiales habituelles. Ainsi, les maillages normaux ont une application directe à la compression géométrique progressive, à condition que l'application finale tolère des pertes d'information sur la connectivité du maillage original [61, 60].

Enfin, Karni et Gotsman [58] utilisent l'approche spectrale propre à l'analyse multirésolution à des fins de compression géométrique avec perte. Leur algorithme partitionne le maillage original en sous-maillages locaux afin de réduire la complexité algorithmique. Puis chaque sous-maillage S est représenté par une combinaison linéaire de fonctions de base orthogonales définies par les vecteurs propres du laplacien topologique de S.

Chapitre 3

Présentation de l'algorithme principal

L'algorithme présenté dans cette section est au centre de ce mémoire. Il permet de coder de manière compacte, progressive, et sans perte d'information un nuage de points en dimension quelconque, avec une borne inférieure sur le gain généré. Après une description détaillée de la méthode (section 3.1), nous proposons une analyse théorique de ses performances dans le cas le pire (section 3.2). Nous revenons ensuite sur ses principales caractéristiques (section 3.3) et montrons comment ses performances peuvent être optimisées par une modélisation statistique et du codage entropique (section 3.4).

3.1 Description de l'algorithme

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 à coordonnées entières 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, puisque ce nombre peut prendre les $p+1$ valeurs 0,1,...,p. (Nous avons vu dans la section 1.3.2 comment le codage arithmétique permet, en affectant un seul nombre réel à une séquence de symboles, de coder un symbole sur un nombre de bits non entier.)

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 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. $\quad\mathcal{S} \leftarrow$extraire le premier segment de $\mathcal{L}$
6. $\quad n \leftarrow$nombre de points situés sur $\mathcal{S}$
7. $\quad\mathcal{S}_1 \leftarrow$moitié gauche de $\mathcal{S}$
8. $\quad n_1 \leftarrow$nombre de points situés sur $\mathcal{S}_1$
9. $\quad\mathcal{S}_2 \leftarrow$moitié droite de $\mathcal{S}$
10. $\quad n_2 \leftarrow$nombre de points situés sur $\mathcal{S}_2$
11. $\quad$écrire n_1 sur $\log_2(n+1)$ bits
12. $\quad$**si** $n_1 > 0$
13. $\quad\quad$**alors** ajouter $\mathcal{S}_1$ à la fin de $\mathcal{L}$
14. $\quad$**si** $n_2 > 0$
15. $\quad\quad$**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 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 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 leur coordonnée 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, ou encore de générer un point au milieu du segment lorsque n_i est non nul (cf 3.3.3).

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 3.1 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. Le code correspondant à cet exemple est présenté au-dessous.

3.2 Analyse théorique

3.2.1 Facteur de compression

Pour réaliser une analyse théorique de l'algorithme, nous allons supposer que les n points sont uniformément distribués dans un hypercube de dimension d. Soit 2^{b_i} (pour $i = 1..d$) les longueurs des côtés de l'hypercube (la cellule originale de l'algorithme). Dans la suite, Q désignera le nombre de bits nécessaires pour coder la position d'un point : $Q = \sum_{i=1}^{d} b_i$.

Séparons l'algorithme en deux phases successives :

- séparation des points : les cellules sont subdivisées récursivement jusqu'à ce que chaque cellule de la liste contienne exactement 1 point,

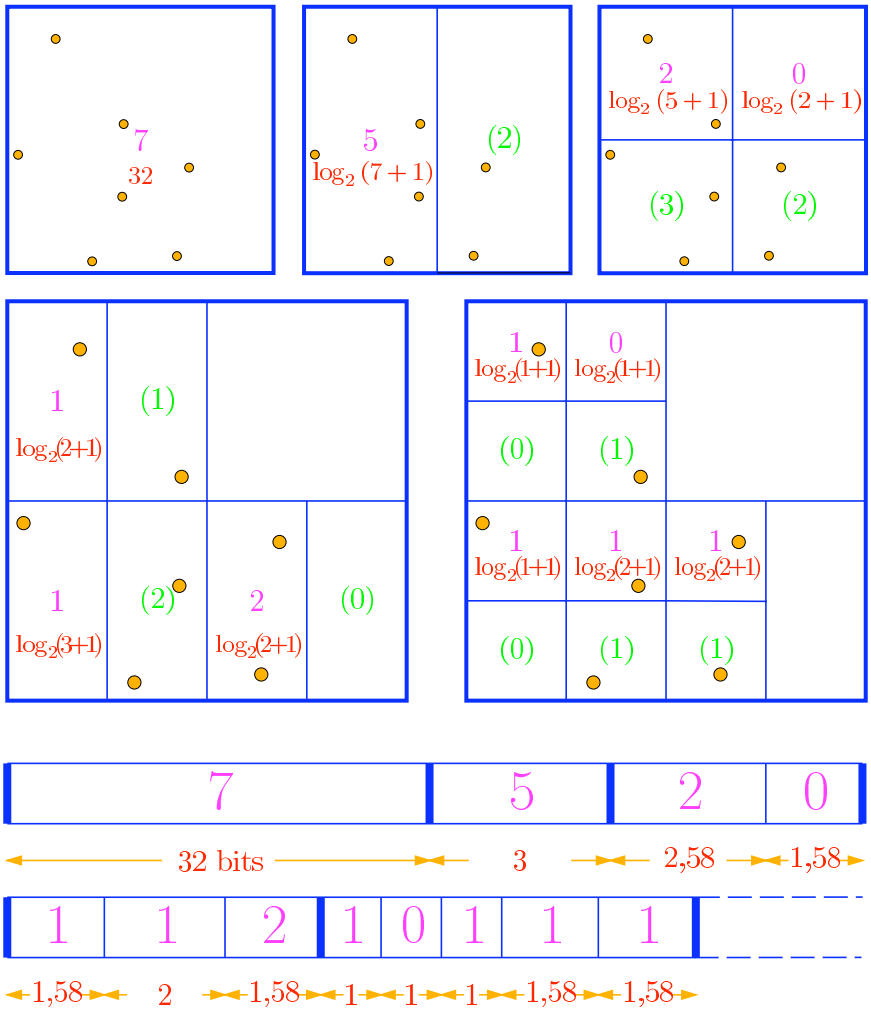


FIG. 3.1 – *L'algorithme de codage sur un exemple bidimensionnel*

– localisation finale : chaque cellule (contenant 1 point seulement) est subdivisée jusqu'à ce qu'elle atteigne la taille unitaire.

Calculons le nombre de bits utilisés pour séparer les points. Avec l'hypothèse d'uniformité, la dichotomie d'une cellule contenant c points génère deux cellules contenant chacune $c/2$ points. Par conséquent, pour séparer n points par cette méthode, $\log_2 n$ subdivisions sont nécessaires. Si l'on décompose l'algorithme en phases définies par la taille des cellules dans la liste courante, le nombre de cellules double et le nombre de points dans chaque cellule est réduit de moitié d'une phase à la suivante. Or, le nombre de bits utilisés

lors de la subdivision d'une cellule contenant c points est égal à $\log_2(c+1)$. Donc finalement, le nombre total de bits utilisés pour coder la séparation des points est donné par :

$$\begin{aligned}
\sum_{i=0}^{\log_2 n-1} 2^i \log_2\left(\frac{n}{2^i}+1\right) &= -\sum_{i=0}^{\log_2 n-1} i\,2^i + \sum_{i=0}^{\log_2 n-1} \log_2(n+2^i)\,2^i \\
&\leq -(n\log_2 n - 2n + 2) \\
&\quad + \frac{n}{2}\log_2\frac{3n}{2} + \frac{n}{4}\log_2\frac{5n}{4} + \frac{n}{8}\log_2\frac{9n}{8} + \ldots \\
&\leq -n\log_2 n + 2n \\
&\quad + n\log_2 n\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\ldots\right) \\
&\quad + n\left(\frac{1}{2}\log_2\frac{3}{2} + \frac{1}{4}\log_2\frac{5}{4} + \frac{1}{8}\log_2\frac{9}{8} + \ldots\right)
\end{aligned}$$

Finalement, les calculs de sommes montrent que le nombre de bits utilisés à l'issue de la phase de séparation des points est inférieur à :

$$N_1 = 2{,}402\,n$$

Une fois qu'une cellule ne contient plus qu'un point, elle doit être subdivisée jusqu'à ce que le point soit complètement localisé. Comme $\log_2 n$ subdivisions ont été réalisées au cours de la phase de séparation, il reste à subdiviser chaque cellule $Q - \log_2 n$ fois. Pendant cette phase, une subdivision coûte 1 bit : le point appartient soit à la première demi-cellule, soit à la seconde. Le nombre de bits utilisés pour coder la localisation finale des points est donc :

$$N_2 = n\,(Q - \log_2 n)$$

Par conséquent, le nombre total de bits utilisés par l'algorithme pour coder les coordonnées des points est :

$$N = n\,(Q - \log_2 n + 2{,}402)$$

Si l'on compare N à nQ (le nombre de bits nécessaires pour coder les points sans compression), on remarque que le gain est $\log_2 n - 2{,}402$ par point, et pour l'ensemble, si l'on néglige la constante additive, il est de $n\log_2 n$, ce qui 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. Cette analyse correspond au comportement de l'algorithme en pratique : il prend en entrée une *liste* de points, et à l'issue d'un cycle de codage/décodage, l'ordre des points dans cette liste est perdu. A la sortie de l'algorithme, l'ordre des points est canonique : il s'agit de l'ordre lexicographique, qui correspond à un parcours en profondeur de l'arbre des cellules. Cet ordre n'est pas porteur d'information (on dit que son entropie est nulle) car il peut être obtenu à partir de l'ensemble de points sans information supplémentaire (par tri suivant les dimensions successives).

Une idée intuitive qui permet d'appréhender le fonctionnement de l'algorithme est donnée par la notion de *partage de l'information* entre les points. Lorsqu'une cellule est scindée en deux sous-cellules de même taille suivant l'axe (Ox), répartir les points qu'elle contient dans les deux sous-cellules revient exactement à augmenter d'un bit la précision sur la coordonnée en x de *chacun* de ces points. Or, pour une cellule contenant k points, le codage de cette information coûtera $\log_2(k+1)$ bits au lieu de k bits (1 bit de précision par point). Ainsi, on peut dire que les k points de la cellule partagent l'information portée par le nombre $\log_2(k+1)$ qui est ajouté à la séquence codante. Le gain généré par la méthode peut donc être exprimé par le rapport $k/\log_2(k+1)$, qui vaut 1 lorsque $k = 1$, et croît strictement lorsque k augmente : le gain croît avec le nombre de points contenus dans la cellule subdivisée. En particulier, cela montre qu'aucun gain n'est généré durant la phase de localisation finale, puisque les cellules ne contiennent plus qu'un point.

Le raisonnement qui précède permet de montrer que le gain théorique établi dans cette section est une borne inférieure, dans la mesure où la distribution uniforme des points constitue le cas le pire pour l'algorithme. En effet, soit c une cellule contenant k points et c_0 et c_1 les deux sous-cellules issues de la subdivision, contenant respectivement k_0 et $k - k_0$ points. Le coût des deux subdivisions à l'étape suivante de l'algorithme est donné par $c(k_0) = \log_2(k_0+1)+\log_2(k-k_0+1)$ pour $k_0 = 1..k-1$, et $c(0) = c(k) = \log_2(k+1)$. Or, il est facile de montrer que c atteint son maximum pour $k_0 = k/2$. La méthode est donc d'autant plus efficace que la distribution est structurée, ce qui la rend cohérente avec la théorie de l'information.

Remarque 3.2.1 *Il est intéressant de noter que ce gain théorique minimal correspond au gain estimé résultant d'un codage différentiel des positions. L'idée intuitive d'un tel codage est de définir un ordre sur les points mi-*

nimisant la distance entre 2 points successifs, de coder de manière absolue le premier point de la séquence, puis de coder le vecteur différence entre le point courant et le point suivant dans la séquence. Sous l'hypothèse d'une distribution uniforme de n points dans un espace de dimension d, il y a en moyenne $\sqrt[d]{n}$ *points par axe de coordonnée. La distance entre deux points voisins suivant chaque axe est donc* $t/\sqrt[d]{n}$*, où t est la taille de la boîte englobante contenant les points. Par conséquent, chaque composante du vecteur différence sera codée sur* $\log_2(t/\sqrt[d]{n}) = \log_2 t - \frac{1}{d}\log_2 n$*. Le gain généré pour chaque point est donc de* $\frac{1}{d}\log_2 n$ *bits par coordonnée, soit au total* $\log_2 n$ *bits. La différence fondamentale est que pour le codeur différentiel, la distribution uniforme des points constitue le cas optimal; le gain estimé de* $\log_2 n$ *bits est donc une borne supérieure. En outre, si le modèle de distribution uniforme paraît raisonnable dans le cadre des triangulations en 2D ou des tétraédrisations en 3D, il est en revanche peu adapté au cas des maillages surfaciques.*

Remarque 3.2.2 *Pour cette analyse, nous avons distingué la phase de séparation de la phase de localisation finale. En pratique, pour des distributions arbitraires de points, ces deux phases sont réalisées simultanément.*

Outre la notion intuitive de partage de l'information évoquée précédemment, l'efficacité de l'algorithme s'explique par la méthode utilisée pour le codage des nombres de points contenus dans les sous-cellules. Contrairement aux approches à base d'arbres quaternaires (*quadtree*) et octaux (*octree*) utilisées en particulier pour la localisation de sommets dans le cadre de la simplification de maillages [90, 76, 77], l'algorithme repose sur une *dichotomie* des cellules. En 2 dimensions (resp. 3 dimensions), la subdivision d'une cellule n'engendre pas directement 4 (resp. 8) sous-cellules, mais 2, puis 2×2, (puis 2×4 en 3 dimensions) sous-cellules, structurant ainsi les cellules selon un arbre binaire. Du fait que la taille des codes est ajustée en fonction du contenu de la cellule mère, cette subdivision progressive des cellules permet d'optimiser le codage des nombres de points, comme le montre l'exemple bidimensionnel de la figure 3.2. De plus, une économie supplémentaire est réalisée par rapport au découpage direct chaque fois que la dichotomie détecte une cellule vide c_i, qui n'engendre plus aucun coût ; en effet, dans le cas du découpage direct, les sous-cellules (vides) de c_i seraient codées (cf figure 3.3).

Une autre différence fondamentale avec les approches classiques pour la localisation de points dans l'espace à l'aide de *kd-trees* [86] est que notre

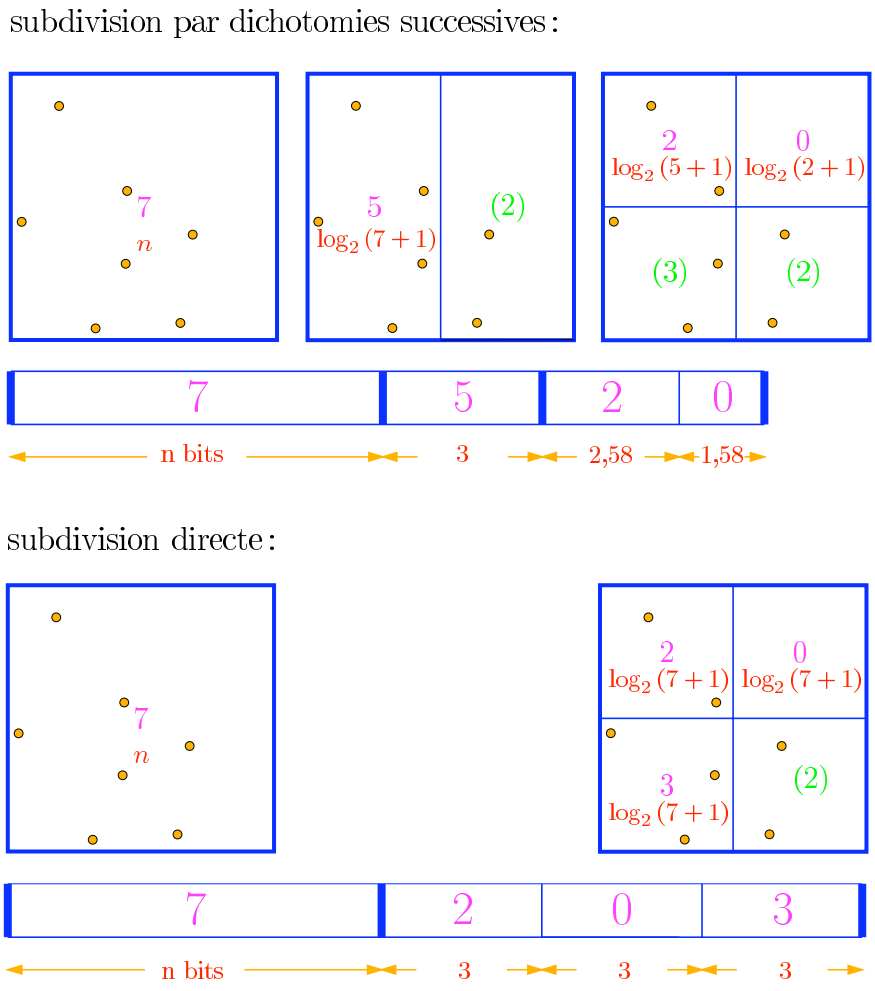


FIG. 3.2 – *Comparaison de l'efficacité du codage adaptatif*

algorithme code le nombre de points contenus dans chaque cellule, et non pas seulement la présence ou l'absence de points dans la cellule. Contrairement à ce qu'on pourrait penser intuitivement, cela *diminue* la taille du code pour deux raisons : i) la connaissance du nombre de points de la première sous-cellule permet de déduire celui de la seconde (une telle déduction est impossible avec l'information binaire de présence ou d'absence) ; ii) ainsi, le seul moyen de ne pas transmettre deux fois plus de codes consiste à subdiviser les cellules par découpage direct ; or, comme nous l'avons vu, cela engendre un coût supplémentaire à cause de la gestion moins efficace des cellules vides (cf figure 3.3).

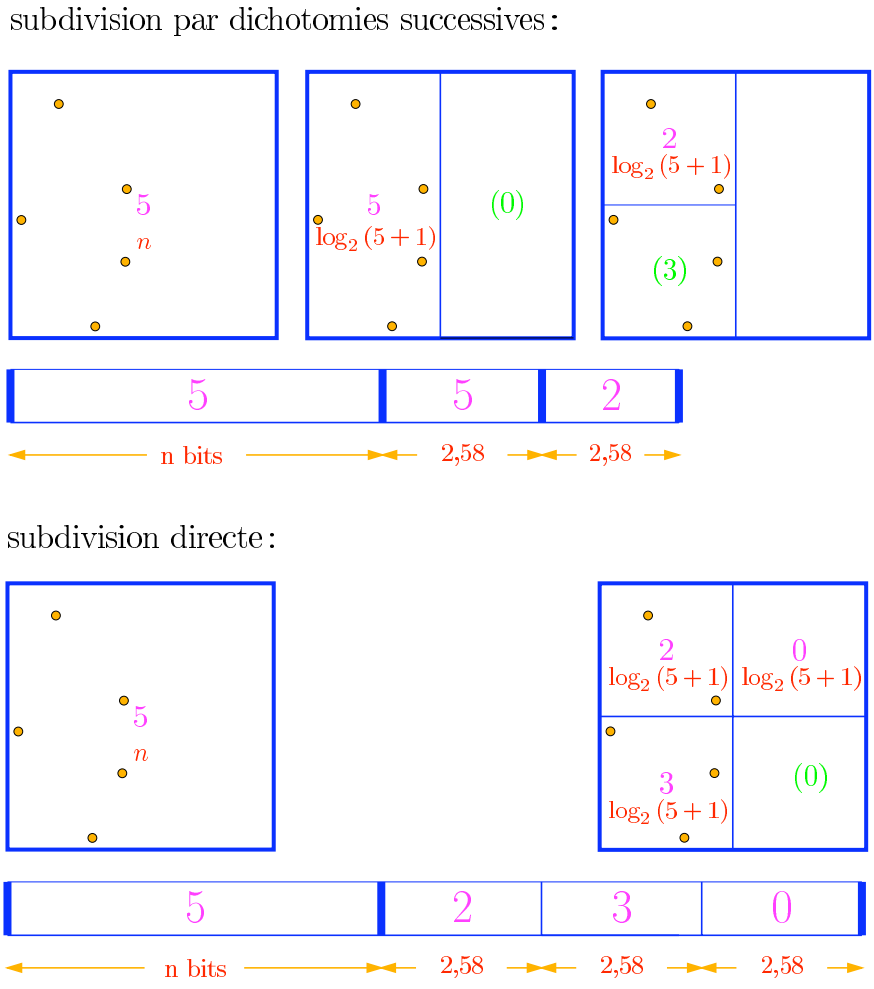


FIG. 3.3 – *Comparaison de la gestion des cases vides*

3.2.2 Complexité

L'algorithme (compression et décompression) est linéaire en temps et en espace dans le nombre de points n de l'objet à coder. Cependant, la constante de temps de la décompression est sensiblement inférieure à celle de la compression.

L'opération dominante pour la compression est la séparation/localisation des points. A chaque subdivision de cellule, un test est effectué sur chaque point pour déterminer s'il appartient à la première sous-cellule ou à la seconde. Or, une subdivision de cellule permet de localiser un point avec 1 bit

de précision supplémentaire. Donc si Q est le nombre de bits codant chaque point, l'algorithme effectue exactement Q tests par point. La complexité globale de la compression est donc $Q\,n$.

En ce qui concerne la décompression, l'opération dominante est la lecture des codes qui permettent de localiser les points. Lors de la phase de séparation, sous l'hypothèse d'une distribution uniforme des points, le nombre de codes lus est $1 + 2 + 4 + 8 + ... + n/2 = n - 1$. La séparation des points nécessitant $\log_2 n$ subdivisions (là encore, à condition que les points soient uniformément distribués), il reste $Q - \log_2 n$ codes à lire pour finir de localiser chaque point. Il en résulte que la complexité globale de la décompression est $(Q - \log_2 n + 1)\,n$.

La complexité en espace est la même pour la compression et la décompression, et dépend dans une large mesure des structures de données utilisées. Nous n'avons pas cherché à optimiser notre code de ce point de vue : dans l'implantation actuelle, nous utilisons une liste de cellules, chacune contenant une liste de points. Par conséquent, lorsque tous les points sont isolés, n pointeurs sont utilisés pour les cellules, plus n pointeurs pour les points (chaque point est stocké dans une liste à 1 élément). Cela conduit à un encombrement mémoire de $(Q + 64)\,n$ bits.

3.3 Caractéristiques

3.3.1 Progressivité

La caractéristique la plus intéressante de l'algorithme est la possibilité de coder (et décoder) les scènes géométriques de manière progressive. Nous avons vu dans la section 3.1 que les seules sorties du codeur étaient les nombres de points contenus dans les cellules successives, et que les tailles et les positions de ces cellules étaient implicitement codées dans l'ordre des sorties. Le choix de cet ordre, c'est-à-dire de la façon de subdiviser l'ensemble de points original, peut être optimisé pour privilégier le codage progressif de la scène. Puisque l'algorithme structure les cellules en arbre binaire, deux parcours sont possibles. Le premier est un parcours en profondeur : chaque point est complètement localisé avant de passer au point suivant. Dans le second (parcours en largeur), toutes les cellules de la même taille sont traitées, générant des cellules deux fois plus petites qui seront traitées ensemble

à l'étape suivante de l'algorithme. Ainsi, après le décodage d'une "vague" entière de cellules, il est possible de reconstruire une version intermédiaire de l'ensemble de points telle que la précision est la même sur chaque point. Un moyen naturel de réaliser cette reconstruction est de générer n_i points uniformément ou régulièrement répartis dans la cellule c_i (cf section 3.3.3. Bien sûr, si la précision uniforme n'est pas nécessaire, la scène peut être visualisée à n'importe quel stade de la décompression, et même en temps réel. Ainsi, pour les applications réseau (en particulier, le survol de données), il est possible de comprimer un ensemble de points sans quantification préalable (compression sans perte), et d'envoyer les versions successivement raffinées de la scène 3D à l'utilisateur final jusqu'à ce qu'il considère que la précision suffit à ses besoins.

3.3.2 Interactivité

En fait, l'algorithme permet d'aller plus loin dans l'interactivité avec l'utilisateur. Puisque les cellules sont structurées en arbre binaire, il est possible, pendant le décodage, de sélectionner un ou plusieurs sous-ensembles des données et de raffiner seulement ces parties. De cette manière, une navigation interactive à travers une scène 3D peut être optimisée du point de vue de la quantité d'information transmise.

3.3.3 Choix de la distribution

Pour obtenir les versions intermédiaires de la scène décodée, on doit générer des points dans l'espace d-dimensionnel à partir d'un nombre de points n_i et d'une cellule c_i. Une façon naturelle de procéder est d'injecter n_i points uniformément répartis dans la boîte englobante c_i, mais ce n'est pas la seule. Une analyse préalable de la scène peut indiquer que les points suivent localement une autre loi de probabilité, ou sont fortement structurés. Par exemple, les modèles de terrain sont souvent construits à partir d'un maillage triangulaire 2D régulier. Il suffit donc d'ajouter en en-tête des données comprimées la méthode de reconstruction la plus adaptée à la scène.

En pratique, pour la plupart des modèles, les meilleurs résultats visuels sont obtenus en générant au centre de chaque cellule c_i non vide ($n_i > 0$) un unique représentant p_i pour l'ensemble des points de c_i. Cependant, si les contraintes matérielles tolèrent un post-traitement, la distribution uniforme

des n_i points dans la cellule suivie d'une phase de lissage [101, 29, 81] peut également fournir une bonne approximation du modèle original.

3.3.4 Dimension

Une autre caractéristique importante de l'algorithme est qu'il peut être appliqué directement à des données en dimension quelconque. Au delà de l'espace tridimensionnel, cela peut être utile pour des données de réalité virtuelle. En effet, le format VRML, largement répandu, associe souvent des données additionnelles aux sommets, telles que normales, surfaces, couleurs ou radiosité. Ces données peuvent être traitées comme des dimensions supplémentaires et donc comprimées de la même manière que les coordonnées. Cependant, il faut se rappeler que l'ordre de grandeur du gain induit par l'algorithme est $n \log_2 n$ (où n est le nombre de points de la scène), à comparer à nQ, la taille des données non comprimées. Ainsi, pour être efficace du point de vue du facteur de compression, l'algorithme doit s'appliquer à des données telles que le quotient $Q/\log_2 n$ ne soit pas trop grand.

En outre, dans le cas de dimensions élevées, le choix de l'ordre de subdivision peut avoir des conséquences importantes sur le facteur de compression. Si la priorité est d'obtenir des représentations intermédiaires fidèles à la scène originale, l'ordre idéal est celui du parcours en largeur, qui consiste à subdiviser toutes les cellules d'une vague suivant la première dimension, puis subdiviser les cellules ainsi obtenues suivant la seconde dimension, ..., jusqu'à la dimension d, et recommencer le même processus jusqu'à la localisation complète des points. En revanche, du point de vue de l'efficacité de la compression, le découpage optimal doit créer des cellules vides en priorité, et donc, suivant la distribution des données, il peut être plus rentable de subdiviser les cellules plusieurs fois suivant la même direction.

3.4 Codage entropique et prédiction

L'algorithme que nous avons décrit jusqu'ici n'est pas une méthode de compression dans le sens classique de la théorie de l'information. Habituellement, une méthode de compression fournit une manière d'extraire l'information canonique des données (canonique signifiant ici non redondante) et de la coder. Ce que nous faisons ici est une réorganisation des données dans

le but d'abandonner une partie de l'information qui ne nous intéresse pas (l'ordre sur les points). Par conséquent, il est naturel de penser qu'il reste une part de redondance dans la partie de l'information que l'on conserve (les coordonnées des points).

3.4.1 Codage arithmétique

Le premier intérêt du codage arithmétique (cf section 1.3.2) pour notre algorithme est de coder les nombres de points des cellules sur un nombre de bits optimal, y compris quand ce nombre n'est pas entier. En effet, nous avons vu dans la description de l'algorithme que pour une cellule contenant p points, le nombre de points dans la première demi-cellule issue de la subdivision était codé sur $\log_2(p+1)$ bits. En fait, ce n'est possible que grâce au principe du codage arithmétique : sans une méthode adaptée, ce nombre serait codé sur $\lceil \log_2(p+1) \rceil$ bits. Par suite, le gain pour chaque point serait de $\log_2 n - 3$ au lieu de $\log_2 n - 2{,}402$, comme le montre le calcul suivant :

$$
\begin{aligned}
S &= \sum_{i=0}^{\log_2 n-1} 2^i \left\lceil \log_2 \left(\frac{n}{2^i} + 1 \right) \right\rceil = - \sum_{i=0}^{\log_2 n-1} i\, 2^i + \sum_{i=0}^{\log_2 n-1} \lceil \log_2 (n + 2^i) \rceil\, 2^i \\
&\leq -(n \log_2 n - 2\,n + 2) \\
&\quad + \frac{n}{2} \left\lceil \log_2 \frac{3\,n}{2} \right\rceil + \frac{n}{4} \left\lceil \log_2 \frac{5\,n}{4} \right\rceil + \frac{n}{8} \left\lceil \log_2 \frac{9\,n}{8} \right\rceil + ... \\
&\leq -n \log_2 n + 2\,n \\
&\quad + n \log_2 n \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... \right) \\
&\quad + n \left(\frac{1}{2} \left\lceil \log_2 \frac{3}{2} \right\rceil + \frac{1}{4} \left\lceil \log_2 \frac{5}{4} \right\rceil + \frac{1}{8} \left\lceil \log_2 \frac{9}{8} \right\rceil + ... \right) \\
&\leq 3\,n
\end{aligned}
$$

3.4.2 Méthodes de prédiction

En codant le nombre de points dans la première demi-cellule issue de la subdivision sur $\log_2(p+1)$ bits (où p est le nombre de points dans la cellule mère), on suppose que chaque valeur entière comprise entre 0 et p est équiprobable, de probabilité $1/(p+1)$. Pour améliorer les performances de

l'algorithme, on peut chercher à estimer plus précisément la probabilité de chacune de ces valeurs. Pour cela, on étudie les densités locales des points au voisinage de la cellule à subdiviser.

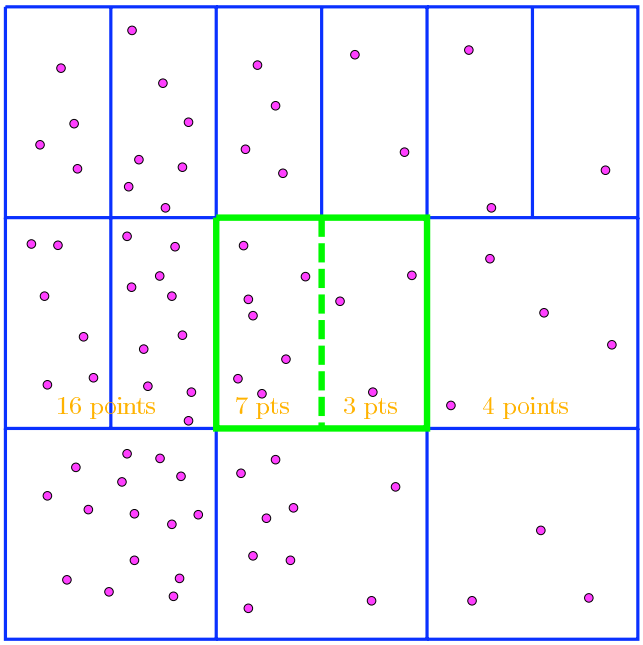


FIG. 3.4 – *Voisinage d'une cellule bidimensionnelle*

La technique de prédiction utilisée repose sur l'hypothèse que les densités locales de points dans la cellule courante sont corrélées avec les densités locales dans son voisinage. Ainsi, l'algorithme analyse le contexte en prenant en compte toute l'information disponible à ce moment précis du codage ou du décodage. Donnons un exemple pour illustrer le principe de la méthode. Supposons que l'on doive subdiviser verticalement la cellule centrale de la figure 3.4. La figure montre l'état de l'arbre de cellules à ce stade. Un moyen très simple de déterminer la répartition de points la plus probable dans les deux demi-cellules est de calculer le pourcentage de points dans la cellule voisine gauche par rapport au nombre total de points voisins (dans les cellules voisines gauche et droite), puis de supposer que les demi-cellules respectent ce pourcentage. Dans l'exemple, on compte 16 voisins à gauche sur un total de 20 voisins, ce qui conduit à prédire 8 points dans la moitié gauche de la cellule courante et 2 points dans sa moitié droite. De là, une méthode simple consiste à estimer les probabilités des 11 valeurs possibles pour la demi-cellule gauche par une loi gaussienne discrète centrée en 8. Ainsi, la valeur effective de la demi-cellule gauche (7 points) aura une probabilité estimée forte, et

sera donc codée sur un petit nombre de bits.

Dans cet exemple, la prédiction utilise seulement le voisinage au premier ordre, mais la technique peut être étendue aux ordres supérieurs, en donnant plus de poids aux cellules voisines les plus proches. En fait, l'ordre du contexte analysé peut être optimisé pour atteindre un compromis satisfaisant entre la précision de la prédiction et la complexité de l'algorithme.

Avec notre réglage des paramètres, cette méthode de prédiction apporte un gain supplémentaire d'environ 5% en moyenne, les meilleurs résultats étant obtenus pour les modèles 3D dont les densités locales sont le plus varié. Il est à noter que la simple liste de cellules utilisée dans la section 3.1 n'est plus suffisante, puisque la prédiction nécessite une structure de données pour un accès rapide au voisinage d'une cellule.

Chapitre 4

Triangulations bidimensionnelles et modèles de terrain

Dans ce chapitre, nous allons voir de quelle manière l'algorithme décrit au chapitre 3 pour comprimer un ensemble de sommets non structuré peut être appliqué à un cas particulier d'objets géométriques structurés : les triangulations bidimensionnelles.

Nous avons vu que le principe de notre algorithme le destinait essentiellement aux structures géométriques dont la topologie peut être reconstruite de manière canonique à partir de la géométrie. Un tel exemple de construction automatique de maillage à partir de ses sommets est donné par la triangulation de Delaunay. En 2 dimensions, il s'agit d'une structure de triangulation canonique, calculable en temps $O(n \log n)$ (où n est le nombre de sommets), et qui offre certaines propriétés de régularité. En particulier, la triangulation de Delaunay maximise son plus petit angle, c'est-à-dire crée des triangles équilibrés (cf section 4.2).

Cette structure trouve des applications dans le domaine des scènes 3D avec les modèles de terrain, dont la topologie est généralement obtenue en triangulant les points sans leur coordonnée en z. En effet, un terrain T est une surface triangulée plongée en 3 dimensions, qui présente la propriété remarquable d'être projetable dans un plan. Autrement dit, il existe un plan dans lequel la projection de T est une subdivision planaire dont les faces sont des triangles.

En l'absence d'information de nature topologique (nuage de points), on

utilise la triangulation de Delaunay pour ses propriétés de régularité. Dans certains cas cependant, on peut vouloir imposer la présence de quelques arêtes dans la triangulation (par exemple pour suivre le contour d'un objet), tout en garantissant les propriétés remarquables de Delaunay pour le reste de la topologie du modèle. On utilise alors la triangulation de Delaunay contrainte, qui permet de construire en temps $O(n \log n)$, à partir d'un ensemble d'arêtes contraintes, une triangulation optimale du point de vue de la régularité de ses triangles (cf section 4.3).

4.1 Précédents travaux

Linear-Time Reconstruction of Delaunay Triangulations with Applications, de Snoeyink et van Kreveld [95] présente un algorithme qui, étant donnée la triangulation de Delaunay TD d'un ensemble P de n points du plan, calcule en temps $O(n)$ une permutation P' de P permettant de reconstruire progressivement la topologie de TD en temps $O(n)$ randomisé sans information supplémentaire. Cette permutation est construite en $O(\log n)$ phases. Au cours d'une phase, un ensemble de sommets indépendants est supprimé, et chaque sommet se voit affecter un pointeur sur la face qui le contient après retriangulation du trou polygonal. L'ordre des points est déterminé par la place de leur face associée dans le parcours de la triangulation courante (ce parcours est défini canoniquement par la géométrie). La position des sommets est comprimée au moyen d'un codage différentiel, qui consiste à coder le vecteur de déplacement entre deux points successifs dans la séquence ordonnée P'. L'efficacité du codage géométrique est donc dépendante de l'ordre des points fixé par le codeur topologique. Le résidu de compression ainsi obtenu est supérieur à 84%, l'objectif principal de cet article n'étant pas de comprimer les données, mais d'obtenir une complexité meilleure que $O(n \log n)$ pour la triangulation de Delaunay.

Fast Reconstruction of Delaunay Triangulations, de Sohler [96], apporte quelques améliorations à la méthode précédente. Tout d'abord, les sommets supprimés au cours d'une phase donnée de l'algorithme de construction de P' ne doivent pas nécessairement constituer un ensemble de sommets indépendants. S'il est vrai que ce changement empêche l'insertion parallèle de sommets pendant la reconstruction incrémentale de TD, il présente en revanche l'avantage de porter la limite sur le nombre de sommets supprimés au cours d'une phase de 1/10 à 1/3. D'autre part, l'algorithme garantit la proximité géométrique de deux points successifs dans P', ce qui permet d'améliorer

l'efficacité de la compression (le résidu de compression obtenu atteint 71%). Enfin, les constantes des algorithmes de calcul d'ordre et de reconstruction sont réduites d'environ 15%.

An Improved TIN Compression Using Delaunay Triangulation, de Kim, Park, Jung et Cho [62], part de l'observation que 90% des arêtes composant un terrain appartiennent à la triangulation de Delaunay TD de l'ensemble de points sous-jacent. Un algorithme est donc proposé pour coder la fraction ΔE de la connectivité qui n'est pas contenue dans TD. Ce codage est réalisé en décomposant ΔE en arêtes simples (arêtes isolées), arbres (chemins d'arêtes) et graphes (chemins cycliques). C'est l'ordre de description des sommets (plus quelques codes d'en-tête) qui permet alors de coder chacune des composantes de ΔE. Lorsque tous les sommets $V(\Delta E)$ décrivant les arêtes contraintes sont transmis, les sommets restant sont envoyés dans un ordre d'innovation décroissante : à chaque étape, c'est le sommet dont la distance à son plan moyen est maximale (c'est-à-dire le sommet le moins "prévisible") qui est transmis [92]. L'utilisation de la triangulation de Delaunay contrainte incrémentale permet ainsi une reconstruction progressive du modèle par le client. Malgré un coût global moyen de 0,23 bits par sommet pour la connectivité, l'intérêt de la méthode est limité par l'absence de compression sur les positions.

Encoding a Triangulation as a Permutation of its Point Set, de Denny et Sohler [28], met en évidence cette limite en décrivant deux algorithmes — l'un, théorique, de complexité $O(n)$ et l'autre, plus facilement applicable, en $O(n \log n)$ — permettant de coder une triangulation T d'un ensemble P de points du plan par une permutation P' des éléments de P. Le coût de la connectivité est ainsi ramené à 0 bits par sommet pour une triangulation bidimensionnelle arbitraire. Mais là encore, l'ordre des points dans P' n'est pas favorable au codage des positions par prédiction (dans la mesure où il ne privilégie pas la proximité géométrique de deux points successifs). Par conséquent, le coût total du codage (topologie et géométrie) reste élevé.

4.2 Triangulation de Delaunay

4.2.1 Définitions et propriétés

Définition 4.2.1 (Triangulation de Delaunay) *Soit S un ensemble de sites de l'espace E^d. On appelle simplexe de Delaunay tout simplexe de dimension n p_0, ..., p_n ($p_i \in S$) tel qu'il existe une boule passant par p_0, ..., p_n ne contenant pas de points de S en son intérieur. On appelle triangulation de Delaunay le complexe composé des simplexes de Delaunay de S.*

Remarque 4.2.2 *La triangulation de Delaunay de S n'est unique que si les sites sont en position générale, i.e. s'il n'existe pas $p+2$ points cosphériques dans S. Dans la suite, pour tenir compte des cas dégénérés où de telles cosphéricités existent, on parlera* des *triangulations de Delaunay de S.*

Remarque 4.2.3 *La triangulation de Delaunay existe en dimension quelconque. Dans ce mémoire, nous nous intéresserons à la triangulation de Delaunay bidimensionnelle (pour le présent chapitre) et à la triangulation de Delaunay tridimensionnelle appelée aussi tétraédrisation de Delaunay (au chapitre 6).*

La triangulation de Delaunay possède de nombreuses propriétés qui expliquent son utilisation fréquente en modélisation géométrique.

Définition 4.2.4 (Granularité) *Soit une triangulation T d'un ensemble de points P du plan. A chaque triangle t de T, on associe le rayon r_t du plus petit cercle contenant t, et on appelle grain de T la quantité*

$$G(T) = \max_{t \in T} r_t$$

Théorème 4.2.5 *Parmi toutes les triangulations T d'un ensemble de points P du plan, les triangulations de Delaunay de P sont celles qui ont le grain le plus fin.*

Définition 4.2.6 (Finesse) *Etant donné une triangulation T d'un ensemble P de n points du plan, on définit la finesse de T comme le vecteur $Q(T) = (\alpha_1,...,\alpha_{3t})$, où les α_i sont les angles des p triangles de T classés par ordre croissant.*

Remarque 4.2.7 *Une triangulation qui maximise la finesse selon l'ordre lexicographique maximise notamment le plus petit des angles de ses triangles.*

Théorème 4.2.8 *Etant donné un ensemble de points P du plan, la triangulation qui maximise la finesse selon l'ordre lexicographique est une triangulation de Delaunay de P.*

Pour la preuve des théorèmes 4.2.5 et 4.2.8, le lecteur pourra se reporter aux travaux de Boissonnat et Yvinec [16, 17]. Au nombre des propriétés utiles de la triangulation de Delaunay, on notera que ses faces extérieures forment la frontière de l'enveloppe convexe de l'ensemble de points triangulé. La figure 4.1 permet de juger visuellement de la régularité de la triangulation de Delaunay en comparaison d'une triangulation quelconque du même ensemble de points.

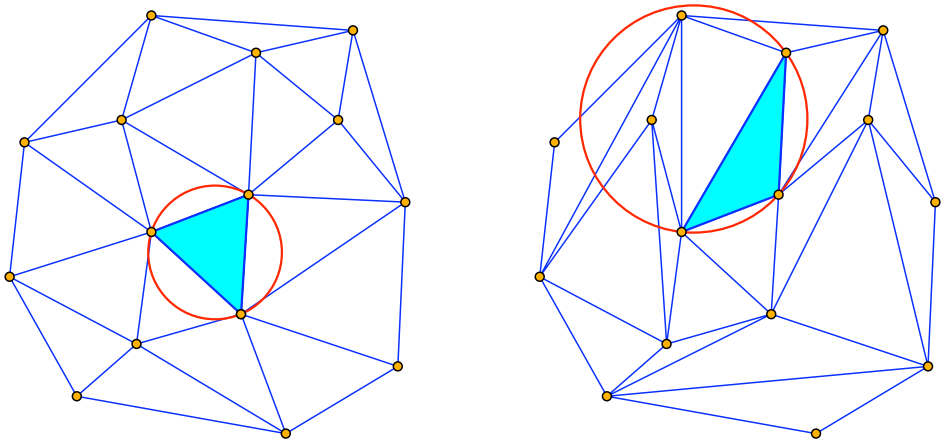


FIG. 4.1 – *La triangulation de Delaunay et une triangulation quelconque d'un même ensemble de points du plan*

4.2.2 Application

Lorsque la topologie du modèle de terrain est intégralement obtenue par la triangulation de Delaunay du projeté de l'ensemble de sommets initial, le problème est ramené à la compression d'un ensemble de points non connectés, et l'algorithme présenté au chapitre 3 est donc directement applicable.

Cependant, il faut noter que la reconstruction n'est pas unique, puisque dans le cas de points cocycliques, le choix de l'arête de Delaunay est arbitraire (cf figure 4.2).

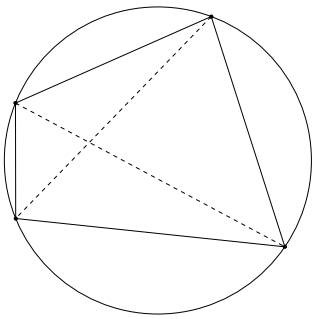


FIG. 4.2 – *Points cocycliques*

En outre, l'économie du codage de la topologie induit une contrepartie importante en terme de complexité, puisque l'on passe d'une reconstruction en temps linéaire dans le cas du codage explicite de la connectivité, à une reconstruction en temps $O(n \log n)$. L'incidence de ce surcoût sur l'application finale est fonction des contraintes matérielles. Dans le cadre d'une transmission réseau du modèle, le facteur limitant est généralement le débit du réseau, et par conséquent, le temps de reconstruction n'intervient pas de manière sensible au niveau utilisateur. En outre, les algorithmes les plus efficaces actuellement pour construire la triangulation de Delaunay sont suffisamment rapides [30] pour que l'on puisse généralement négliger le surcoût engendré.

4.3 Triangulation de Delaunay contrainte

Dans certains cas, plutôt que d'être implicitement définie par un algorithme de reconstruction comme la triangulation de Delaunay de l'ensemble de points projeté en 2 dimensions, la topologie du modèle de terrain est explicitement déterminée. Cela arrive en particulier lorsque l'on impose à la connectivité de respecter certains contours du modèle. Dans le cas des terrains, il peut s'agir par exemple de suivre une rivière ou une route. Cependant, très souvent, la plupart des arêtes de ces modèles vérifient le critère de Delaunay, et sont donc automatiquement reconstructibles. En pratique, moins de 5% des arêtes ne sont pas localement de Delaunay, comme le montrent les statistiques effectuées sur des modèles de terrain usuels (cf tableau 4.2).

Par conséquent, de telles triangulations bidimensionnelles peuvent être reconstruites à partir de leurs sommets et d'un sous-ensemble de leurs arêtes en utilisant la triangulation de Delaunay contrainte.

Après quelques définitions sur la triangulation de Delaunay contrainte, nous allons nous intéresser aux deux problèmes théoriques suivants :

- étant donnée une triangulation bidimensionnelle T, trouver l'ensemble minimal d'arêtes E tel que T soit la triangulation de Delaunay contrainte de E,
- trouver la borne supérieure pour cet ensemble minimal.

Nous verrons ensuite, dans la section 4.4, comment coder cet ensemble d'arêtes minimal.

4.3.1 Définitions

Définition 4.3.1 (Critère de Delaunay) *Soit $p_1\,p_2$ une arête d'une triangulation bidimensionnelle T. $p_1\,p_2$ est appelée arête de Delaunay s'il existe un cercle passant par p_1 et p_2 ne contenant aucun point de T (cf figure 4.3).*

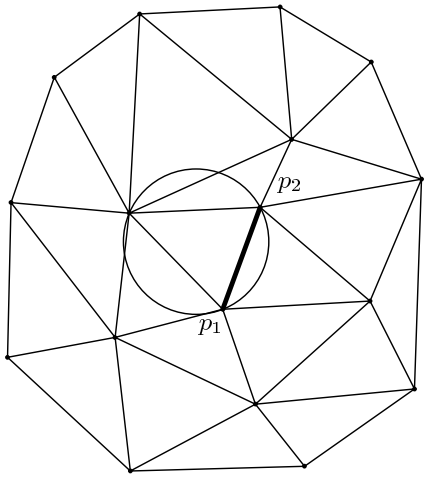


FIG. 4.3 – *Critère de Delaunay*

Définition 4.3.2 (Critère de Delaunay local) *Soit $p_1\,p_3$ une arête de T, et soient $\{p_1\,p_2\,p_3\}$ et $\{p_1\,p_3\,p_4\}$ les triangles adjacents à $p_1\,p_3$. $p_1\,p_3$ est dite*

localement de Delaunay si le cercle $(p_1\,p_2\,p_3)$ ne contient pas p_4, ou, de manière équivalente, si le cercle $(p_1\,p_3\,p_4)$ ne contient pas p_2 (cf figure 4.4).

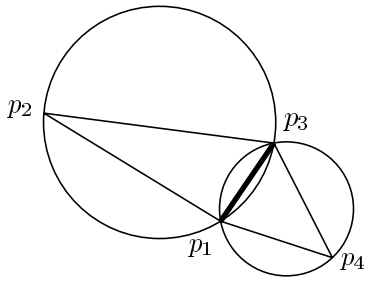


FIG. 4.4 – *Critère de Delaunay local*

Remarque 4.3.3 *En particulier, si le quadrilatère $\{p_1\,p_2\,p_3\,p_4\}$ est non convexe avec son angle concave en p_1 ou p_3, alors $p_1\,p_3$ est localement de Delaunay (cf figure 4.5).*

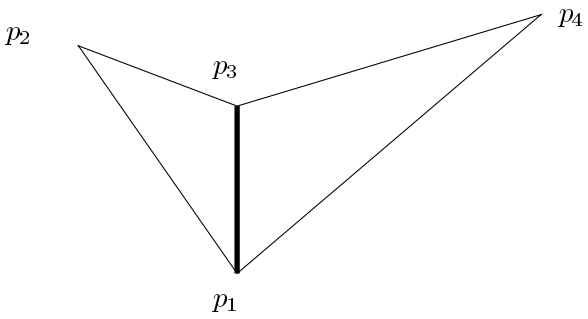


FIG. 4.5 – *Quadrilatère non convexe*

Définition 4.3.4 (Bascule d'arête) *Soit $p_1\,p_3$ une arête de T, et soient p_2 et p_4 les sommets de ses triangles adjacents. Basculer l'arête $p_1\,p_3$ revient à la remplacer par $p_2\,p_4$ dans T. Ce n'est possible que si $\{p_1,p_2,p_3,p_4\}$ est un quadrilatère convexe (cf figure 4.6).*

Définition 4.3.5 (Triangulation de Delaunay contrainte) *Etant donnés un ensemble de points P et un ensemble d'arêtes A dans le plan, la triangulation de Delaunay contrainte $CD(P,A)$ est l'unique triangulation telle que toute arête qui n'est pas localement de Delaunay appartient à A.*

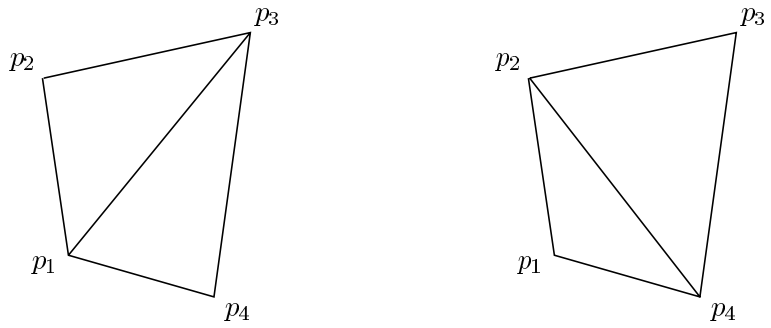


FIG. 4.6 – *Bascule d'arête*

Remarque 4.3.6 *Cette définition est équivalente à la définition classique utilisée par exemple par Chew [21] : $CD(P,A)$ est l'unique triangulation contenant E et telle que pour chaque arête restante a of $CD(P,A)$, il existe un cercle c ayant les propriétés suivantes :*

(1) *Les extrémités de a sont sur c*
(2) *Tout sommet s de A à l'intérieur de c ne peut pas être "vu" depuis au moins une des extrémités de a.*

Cette équivalence est démontrée en particulier dans un article de Lee et Lin [71].

4.3.2 Ensemble de contraintes minimal

Théorème 4.3.7 *Soit T une triangulation bidimensionnelle, P_T l'ensemble de ses sommets, et NLD_T l'ensemble de ses arêtes qui ne sont pas localement de Delaunay. Alors NLD_T est l'ensemble de contraintes minimal de T.*

Démonstration : Il est facile de voir que $CD(P_T,NLD_T) = T$. En effet, $NLD_T \subset T$, donc il est possible de compléter NLD_T pour obtenir T. Ce faisant, on ajoute $T \setminus NLD_T$, qui, par définition, est constitué exclusivement d'arêtes localement de Delaunay. Par conséquent, aucune de ces arêtes n'est basculable, et la triangulation de Delaunay contrainte est terminée. NLD_T est donc un ensemble de contraintes suffisant.

Réciproquement, montrons que NLD_T est nécessaire. Soit W_T un sous-ensemble des arêtes de T tel que $CD(P_T,W_T) = T$. Soit a_1a_3 une arête de

$NLD_T \setminus W_T$, et $\{a_1,a_2,a_3,a_4\}$ le quadrilatère associé (par la remarque 4.3.3, ce quadrilatère est convexe). Puisque $CD(P_T,W_T) = T$, alors $CD(P_T,W_T \cup \{a_1a_2,a_2a_3,a_3a_4,a_4a_1\}) = T$. Cela implique en particulier $CD(\{a_1,a_2,a_3,a_4\}, \{a_1a_2,a_2a_3,a_3a_4,a_4a_1\}) \subset T$, ce qui est faux puisque a_1a_3 n'est pas localement de Delaunay. Il en découle qu'une telle arête a_1a_3 ne peut pas exister, et par suite NLD_T est minimal. □

Remarque 4.3.8 *On obtient ainsi de manière directe un algorithme linéaire pour calculer l'ensemble de contraintes minimal d'une triangulation bidimensionnelle.*

4.3.3 Etude du cas le pire

Dans cette section, nous allons montrer que, étant donnée une triangulation de n points dans le plan, le nombre maximal d'arêtes non localement de Delaunay est la moitié du nombre total d'arêtes, et que cette borne est atteinte.

Lemme 4.3.9 *Etant donnée une triangulation T, tout demi-plan ouvert D dont la frontière contient un point intérieur p de T, contient au moins une arête localement de Delaunay incidente à p.*

Démonstration : Considérons pp_1, la première arête (en parcourant les arêtes dans le sens trigonométrique, depuis le bord d de D) incidente à p dans D (une telle arête existe puisque p est un point intérieur de T). Si pp_1 est la seule arête incidente à p dans D, elle est nécessairement localement de Delaunay (c'est clair par la remarque 4.3.3). Sinon, soit pp_2 la seconde arête incidente à p dans D. Supposons qu'aucune des arêtes pp_1 et pp_2 n'est localement de Delaunay, et considérons le cercle C défini par p, p_1 et p_2. Par construction, l'arc A de C borné par p et p_2 ne contenant pas p_1 est dans D. Puisque pp_2 n'est pas localement de Delaunay, il existe un point p_3 situé dans la région bornée par A et $[p,p_2]$, qui est contenue dans D (cf figure 4.7). En itérant le processus, il est clair que si $[p,p_i]$ n'est pas localement de Delaunay, il existe un point $p_{i+1} \in D$ incident à p. Par conséquent, si aucune des arêtes incidentes à p et contenues dans D n'est localement de Delaunay, il n'y a aucun point incident à p dans l'autre demi-plan, ce qui est impossible puisque p est un point intérieur de T. □

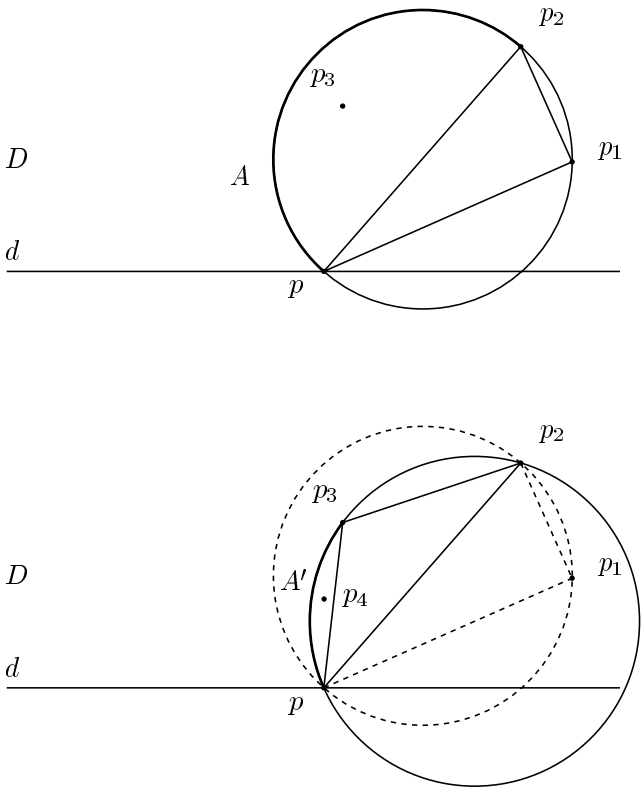


FIG. 4.7 – *Lemme*

Théorème 4.3.10 *Soit P un ensemble de n points dans le plan, dont un nombre i d'entre eux sont intérieurs. Toute triangulation de P contient au moins $n+i/2$ arêtes localement de Delaunay. De plus, cette borne est serrée, à une constante additive près.*

Démonstration : Le lemme 4.3.9 implique clairement que pour tout point intérieur p de T, il existe au moins 3 arêtes incidentes à p dans T qui sont localement de Delaunay. En ajoutant les arêtes de l'enveloppe convexe, on obtient $3\,i/2 + (n-i) = n + i/2$ arêtes localement de Delaunay dans T.

En ce qui concerne le caractère serré de cette borne, puisqu'une triangulation compte $2\,n-3+i$ arêtes, le théorème 4.3.10 peut se formuler en disant que toute triangulation contient au plus $2\,n - 3 + i - (n + i/2) \approx n + i/2$ arêtes non localement de Delaunay. La figure 4.8 montre une triangulation contenant $n + i/2 - 5$ arêtes non localement de Delaunay. □

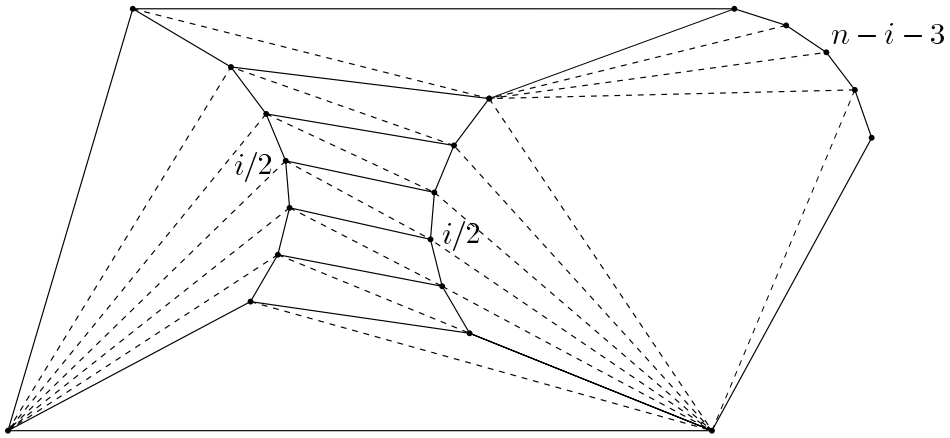


FIG. 4.8 – *Les $n+\frac{i}{2}$ arêtes non localement de Delaunay sont représentées en pointillés*

4.4 Codeur topologique

Pour pouvoir appliquer les résultats obtenus à la section précédente, nous avons maintenant besoin d'un algorithme de codage d'un ensemble d'arêtes entre des points du plan.

Une manière naturelle d'opérer ce codage est d'attribuer un indice à chacun des sommets de l'ensemble initial, et de coder une arête par une paire de pointeurs sur des sommets. Le coût d'un tel codage s'élève à $2\,n_a\,\log_2 n_s$ où n_s désigne le nombre total de sommets et n_a le nombre d'arêtes à coder.

Nous décrivons ici un algorithme plus efficace pour coder un ensemble de relations d'incidence entre points en dimension n. Reposant sur le même principe de subdivision de cellules que l'algorithme principal de compression des points (cf chapitre 3), il permet comme lui un codage progressif des données originales. Il est ainsi possible d'utiliser simultanément les deux algorithmes pour obtenir un codage compact et progressif d'un objet géométrique constitué de sommets et d'arêtes.

4.4.1 Description de l'algorithme

On se replace dans le contexte de l'algorithme de compression de points décrit à la section 3.1.

Définition 4.4.1 (Relation d'incidence entre cellules) *Deux cellules c_1 et c_2 sont dites incidentes s'il existe une relation d'incidence entre un point de c_1 et un point de c_2. On dit encore que les cellules c_1 et c_2 sont voisines.*

Définition 4.4.2 (Super-arête) *On utilisera le terme de super-arête pour représenter une relation d'incidence entre cellules. Une super-arête entre deux cellules c_1 et c_2 peut être vue comme le représentant d'une ou plusieurs arêtes entre des points de c_1 et des points de c_2.*

L'algorithme code ces super-arêtes de la façon suivante. La structure de donnée *cellule* est augmentée d'un champ contenant une liste de pointeurs sur les cellules voisines. Au début du codage, la liste de voisines de la première cellule — qui est la boîte englobante de l'objet — est vide. A chaque subdivision, l'algorithme code la manière dont les super-arêtes incidentes à la cellule originale c sont connectées à ses 2 sous-cellules c_0 et c_1. Plus précisément, il y a 3 possibilités pour chaque super-arête e incidente à c :

- e est incidente à c_0,
- e est incidente à c_1,
- e est incidente à c_0 et c_1.

Par conséquent, pour chaque subdivision de cellule, un code de $\log_2(3)$ bits est associé à chacune de ses super-arêtes. En outre, un bit supplémentaire est nécessaire pour coder l'apparition éventuelle d'une nouvelle super-arête entre les sous-cellules c_0 et c_1. Bien sûr, lorsqu'une subdivision de cellules engendre une sous-cellule vide, aucune information n'est transmise puisqu'il n'y a pas de changement de connectivité : la sous-cellule non vide hérite de toutes les super-arêtes de sa cellule parent.

En ce qui concerne la visualisation des versions intermédiaires du modèle original, il peut être nécessaire d'envisager un travail de régularisation après le décodage, dans la mesure où les premières étapes du codage peuvent engendrer des croisements de super-arêtes (cf figure 4.9).

La figure 4.9 montre un exemple bidimensionnel de codage d'arêtes. Le code représenté sur cette figure correspond à une description des super-arêtes dans le sens horaire, en commençant par la cellule voisine la plus en bas à gauche. A l'issue de la subdivision, la super-arête CV_1 évolue en C_0V_1, donnant lieu à un code c_0. Il en est de même de la super-arête CV_2. La super-arête CV_3 se dédouble en C_0V_3 et C_1V_3 ; elle est donc codée par le symbole c_0c_1. De même pour CV_4, dont l'évolution en C_0V_4 et C_1V_4 crée un croisement d'arêtes. Puis CV_5 et CV_6 évoluent respectivement en C_1V_5 et C_1V_6, engendrant chacune un code c_1. Enfin, le code c_0c_1 décrit la scission de CV_7 en C_0V_7 et C_1V_7. Le symbole *oui* final informe le décodeur de l'existence d'une super-arête entre C_0 et C_1.

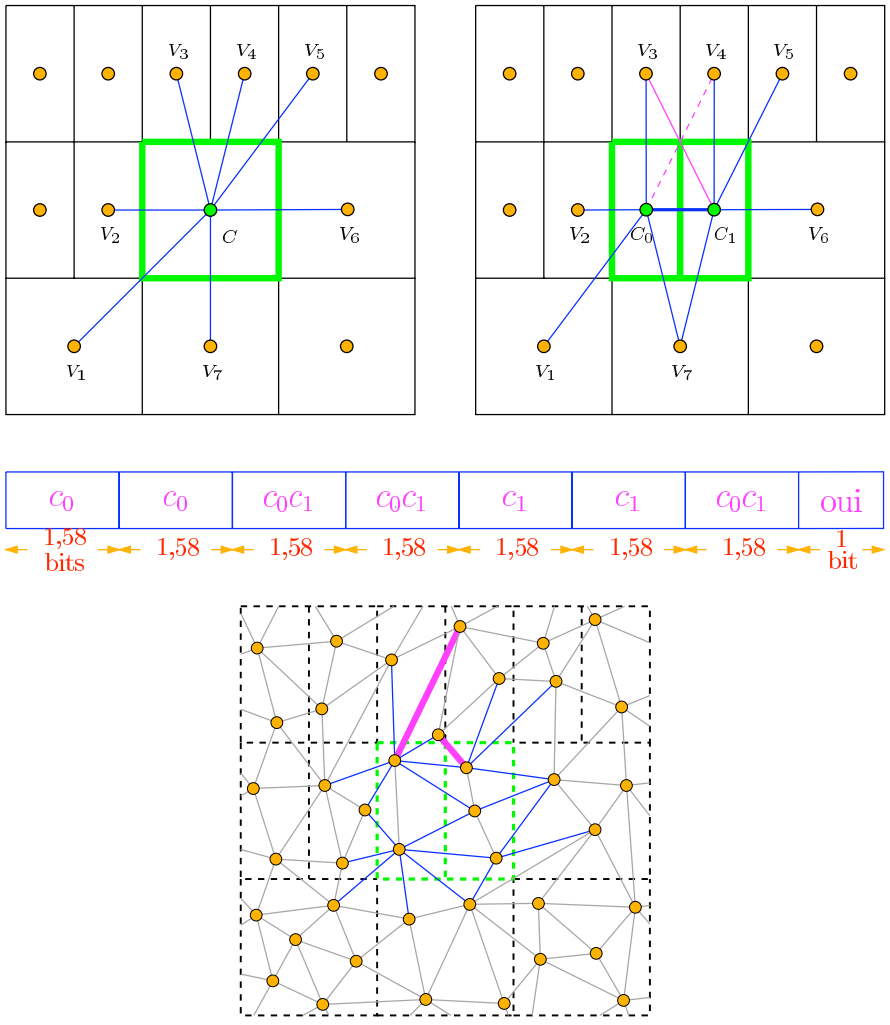


FIG. 4.9 – *Codeur topologique sur un exemple bidimensionnel*

4.5 Résultats expérimentaux

Le tableau 4.1 regroupe les résultats de la méthode décrite à la section 4.2.2 (lorsque la topologie est intégralement reconstruite par triangulation de Delaunay), appliquée à quelques modèles de terrain. Les colonnes 3 et 4 donnent la taille des données en octets et en bits par sommet. Les deux premières lignes proviennent d'une base de données de SIG couvrant la région de Vancouver, tandis que la troisième correspond à un modèle de terrain simple de 14641 points avec des coordonnées sur 12/12/10 bits. Cet exemple permet de visualiser la progression du décodage sur les figures 4.10 à 4.19 : pour chaque résolution, la triangulation de Delaunay est recalculée à partir de l'ensemble de points courant. La figure 4.20 montre la courbe débit/distorsion correspondante, où le débit est exprimé en pourcentage de du fichier non comprimé, et la distorsion est mesurée par l'erreur maximale en tout point exprimée en pourcentage de la diagonale de la boîte englobante.

	nombre de sommets	données originales	données comp.	résidu de comp.	résidu théorique
rivers	120998	650365 43	341365 22,6	52%	66%
vancouver	908907	4885376 43	2169750 19,1	44%	60%
terrain	14641	62225 34	33776 18,5	54%	66%

TAB. 4.1 – *Résultats de la compression sur des modèles de terrain*

Les deux dernières colonnes du tableau 4.1 montrent que la méthode tire parti des répartitions de points non uniformes. Le résidu de compression théorique, calculé à la section 3.2.1 pour une distribution de points uniforme (le cas le pire pour l'algorithme), est réduit d'environ 25% sur les exemples proposés.

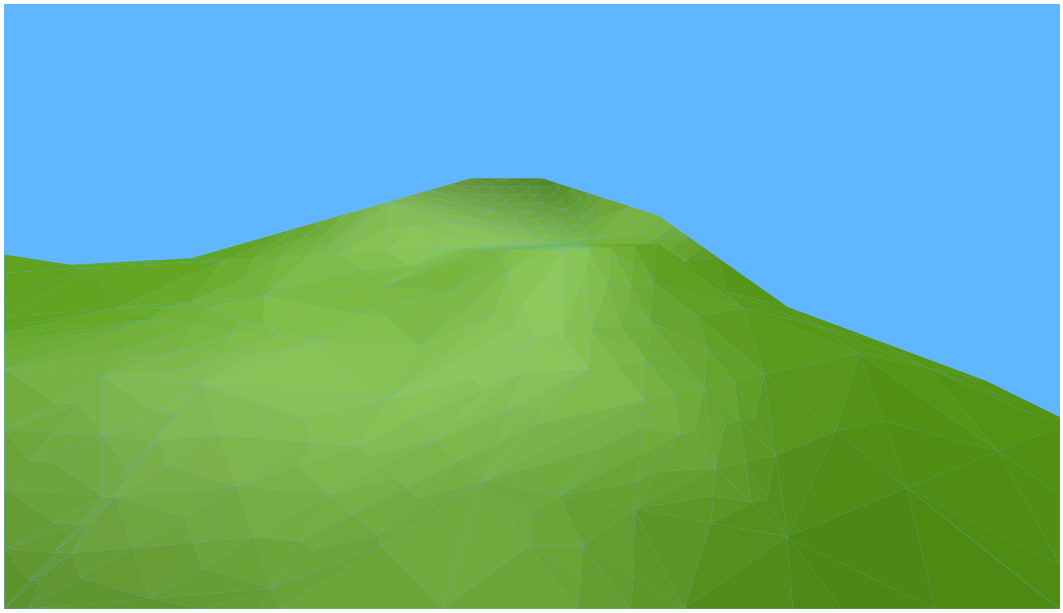


FIG. 4.10 – *Précision* = 3 *bits, Résidu de compression* = 0,4%

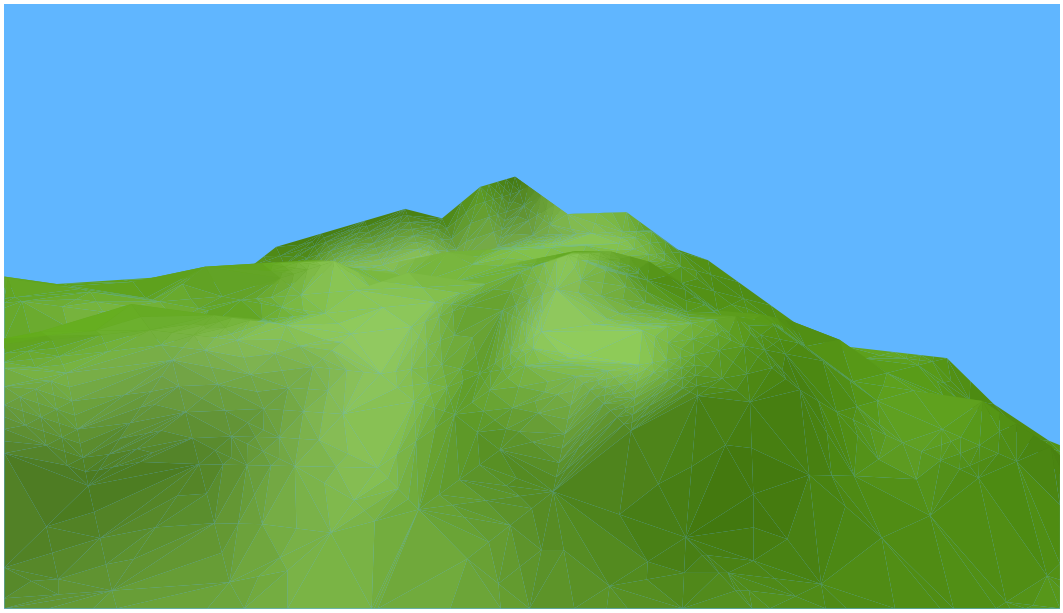


FIG. 4.11 – *Précision* = 4 *bits, Résidu de compression* = 1,3%

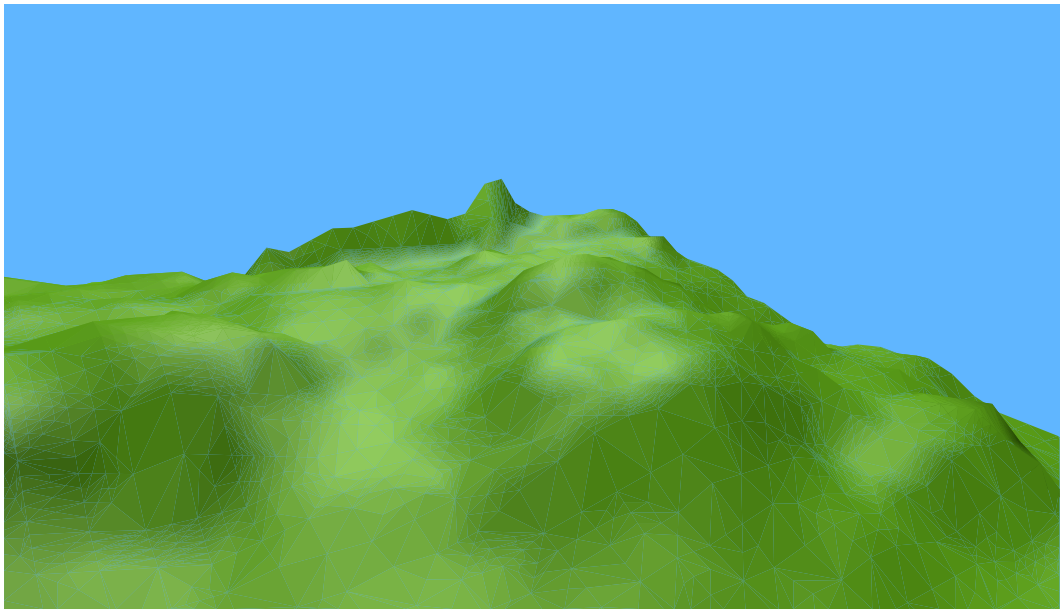


FIG. 4.12 – *Précision* = 5 *bits, Résidu de compression* = 3,6%

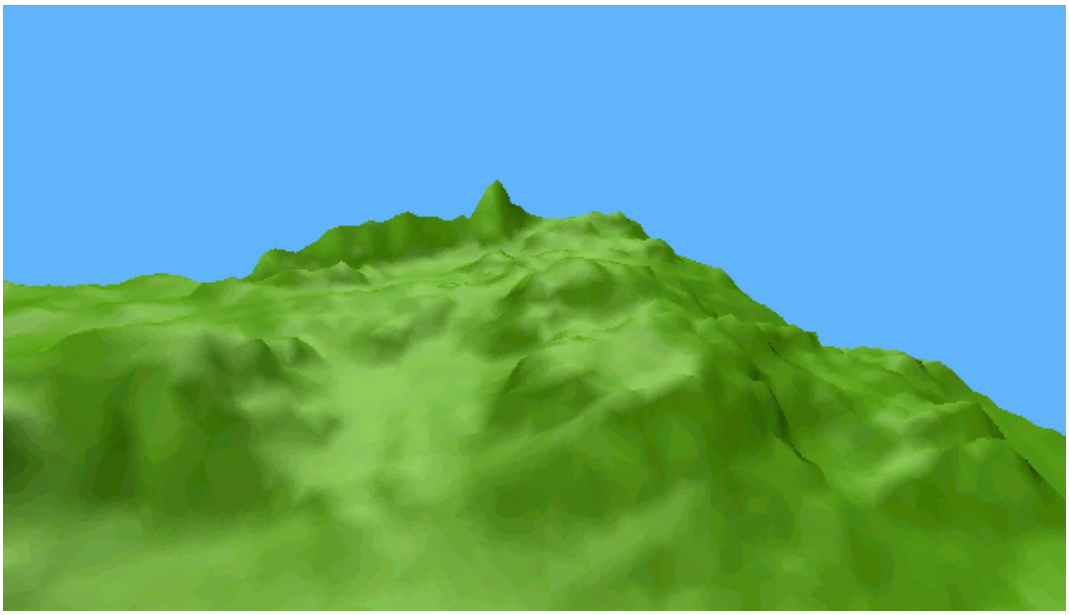


FIG. 4.13 – *Précision* = 6 *bits, Résidu de compression* = 8,6%

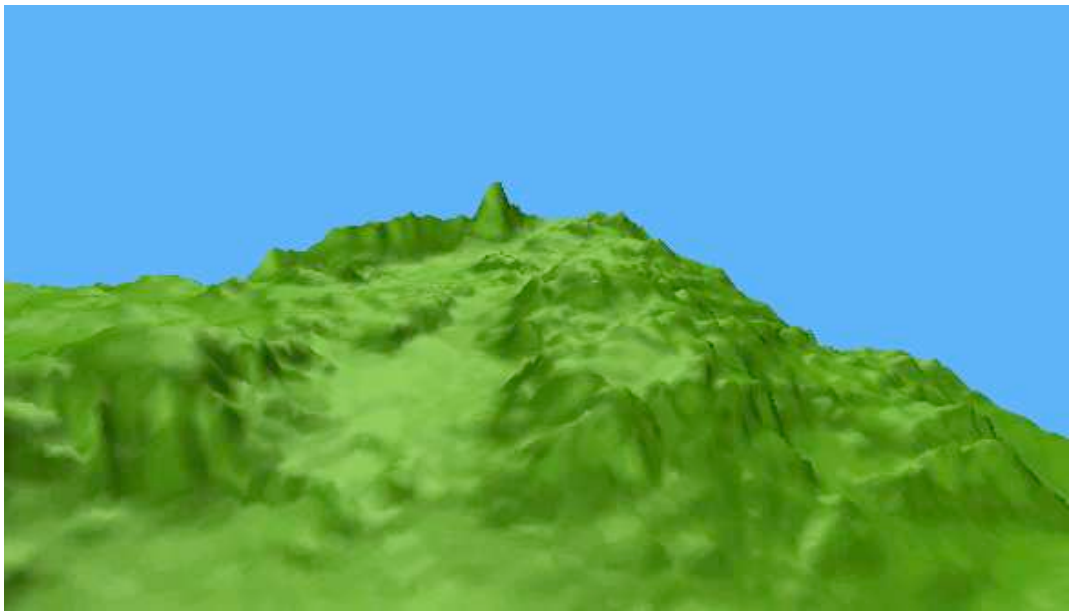


FIG. 4.14 – *Précision* = 7 *bits, Résidu de compression* = 16,4%

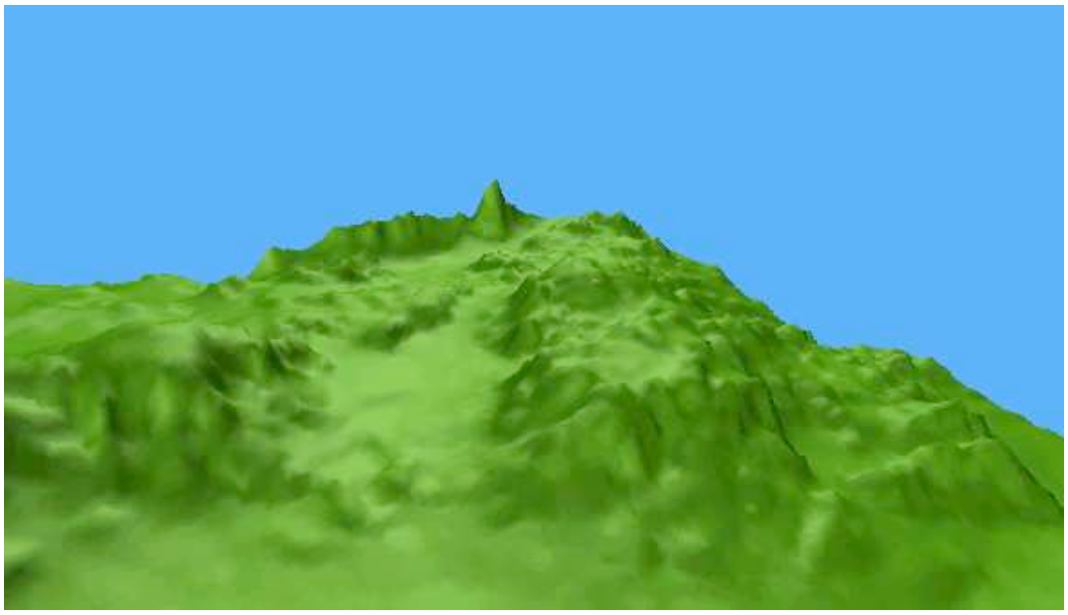


FIG. 4.15 – *Précision* = 8 *bits, Résidu de compression* = 25,2%

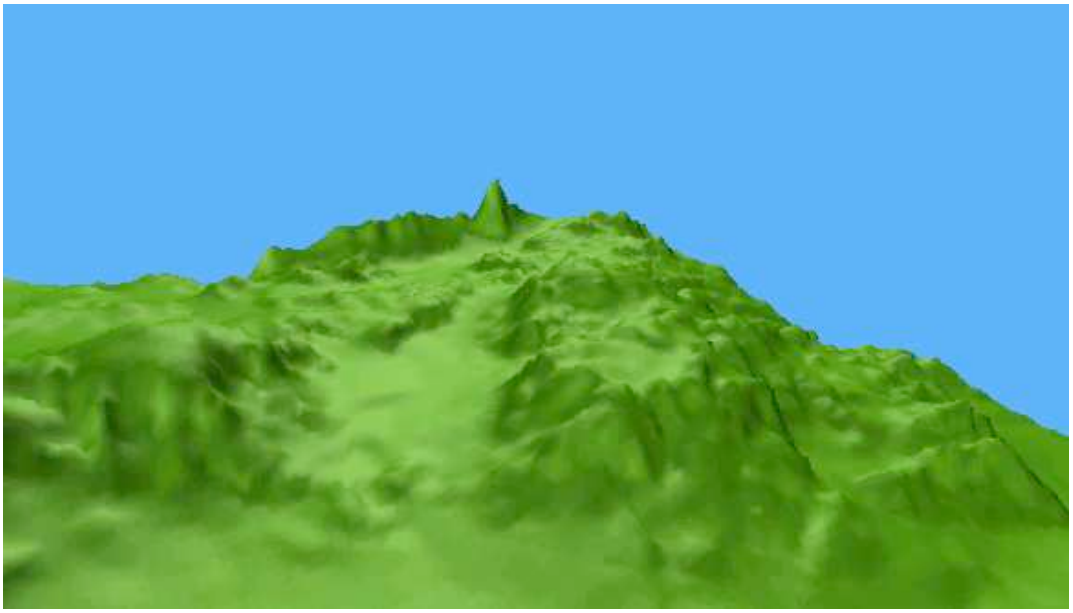


FIG. 4.16 – *Précision* = 9 *bits, Résidu de compression* = 34,0%

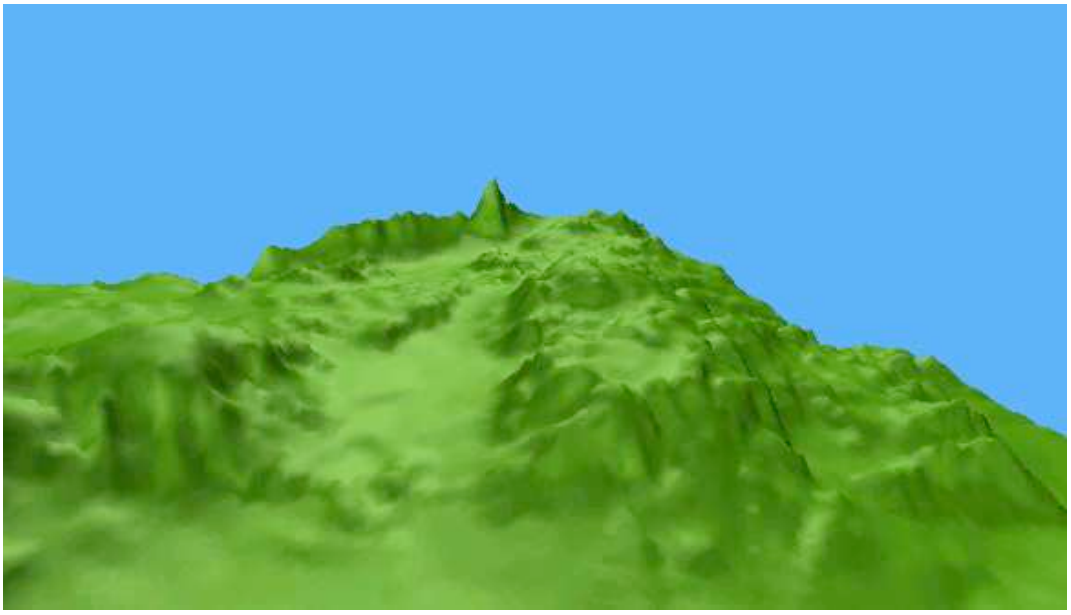


FIG. 4.17 – *Précision* = 10 *bits, Résidu de compression* = 42,8%

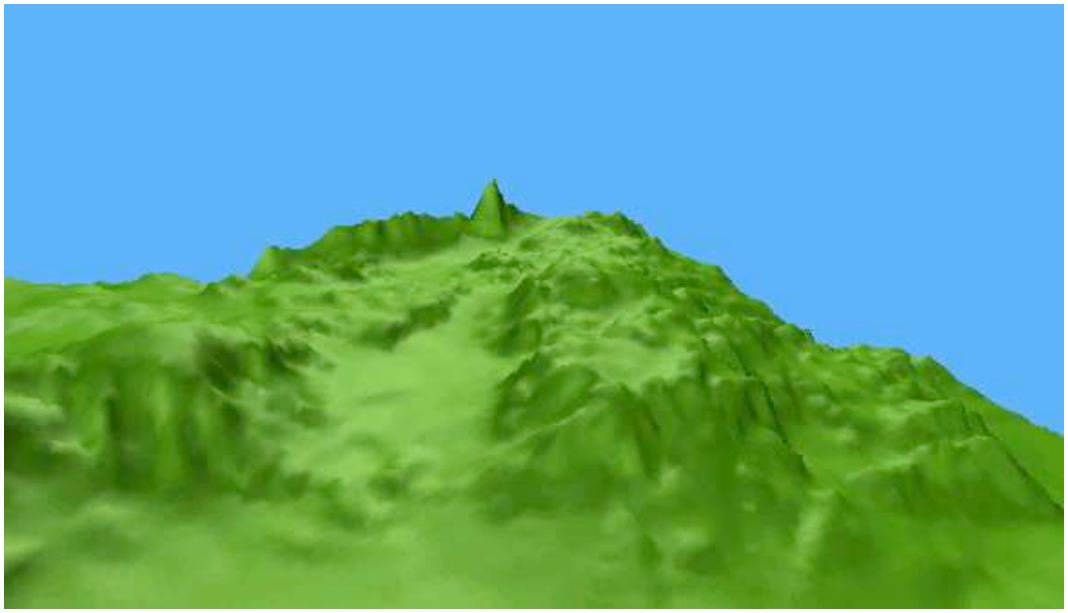


FIG. 4.18 – *Précision* = 11 *bits, Résidu de compression* = 48,7%

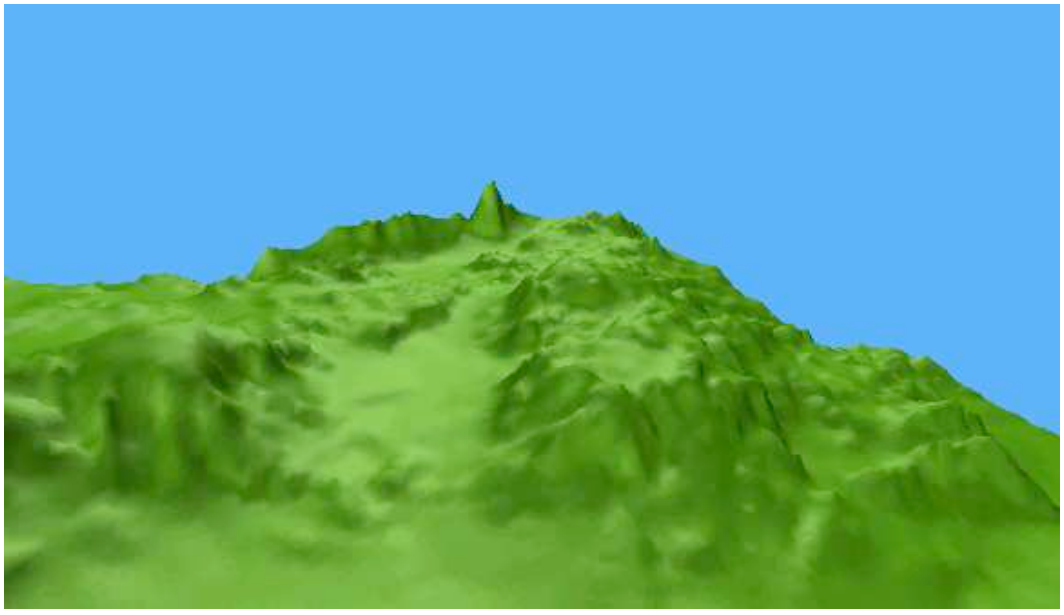


FIG. 4.19 – *Précision* = 12 *bits (compression sans perte), T. c.* = 54,6%

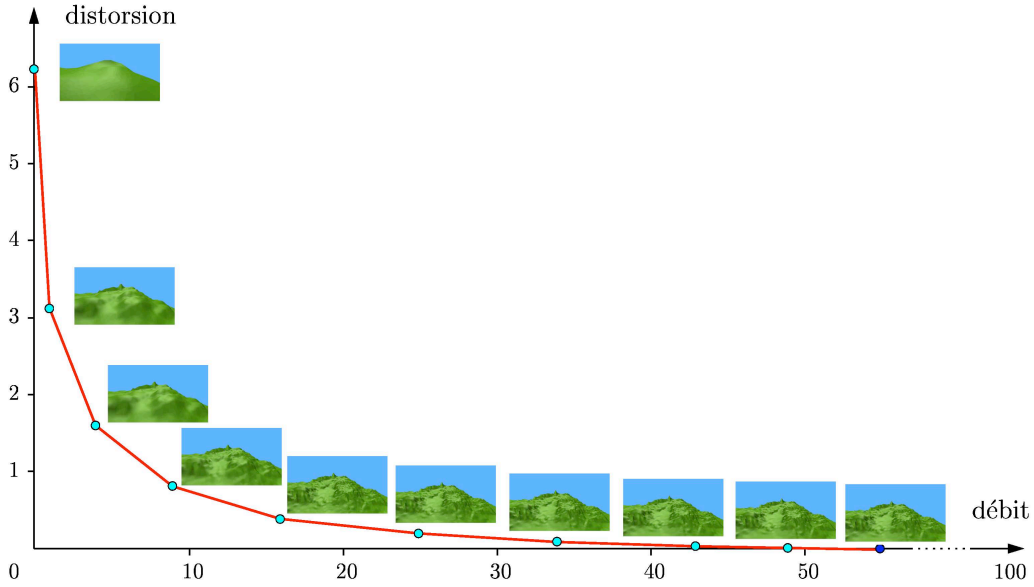


FIG. 4.20 – *Courbe débit/distortion*

Le tableau 4.2 et la figure 4.21 concernent la méthode décrite à la section 4.4, où un sous-ensemble de la connectivité du modèle (les arêtes non localement de Delaunay) est explicitement codé, tandis que la triangulation de Delaunay contrainte permet de reconstruire le reste de l'information topologique. Nous proposons dans le tableau 4.2 quelques statistiques sur la taille de l'ensemble NLD_T en pratique. Le premier exemple du tableau provient de la collection d'objets de Viewpoint (*http://avalon.viewpoint.com/*), tandis que les autres modèles de terrain testés sont issus d'un site de l'U.S. Environmental Protection Agency offrant plusieurs terrains irrégulièrement triangulés (TIN) au format VRML (*http://www.epa.gov/gisvis/vrml/*).

A cause du nombre de bits restreint utilisé pour le codage des coordonnées des sommets, les cas dégénérés constitués de quatre points cocycliques peuvent être relativement fréquents. En référence à la définition 4.3.2, une arête p_1p_3 est dite cocyclique si les quatre points $p_1\,p_2\,p_3$ et p_4 sont cocycliques. Le tableau 4.2 donne le pourcentage d'arêtes non localement de Delaunay, ainsi que le pourcentage d'arêtes cocycliques. Les deux dernières colonnes montrent les résidus de compression (en bits par sommet) obtenus avec les algorithmes décrits aux sections 3.1 et 4.4. Les résidus de compression topologique comprennent le codage des arêtes cocycliques, réalisé par une séquence additionnelle de 1 bit par arête cocyclique. Les résidus de com-

pression géométrique correspondent à une quantification préalable de 12 bits par coordonnée, soit 36 bits par sommet.

	nombre de sommets	nombre d'arêtes	arêtes NLD (%)	arêtes cocycl. (%)	coût topo. (bps)	coût géom. (bps)
Crater	5135	14814	0,3	0,6	0,2	19,8
Gatlinburg area	3037	9016	3,4	5,1	1,3	22,8
Great Smoky Mountains	7636	22736	1,0	21,4	1,1	20,4
Mariposa West California	8390	25061	0,5	18,1	0,8	19,7
Sevierville/Pigeon Forge	3516	10447	3,1	9,9	1,4	21,8

TAB. 4.2 – *Statistiques sur le nombre d'arêtes NLD de modèles de terrain usuels*

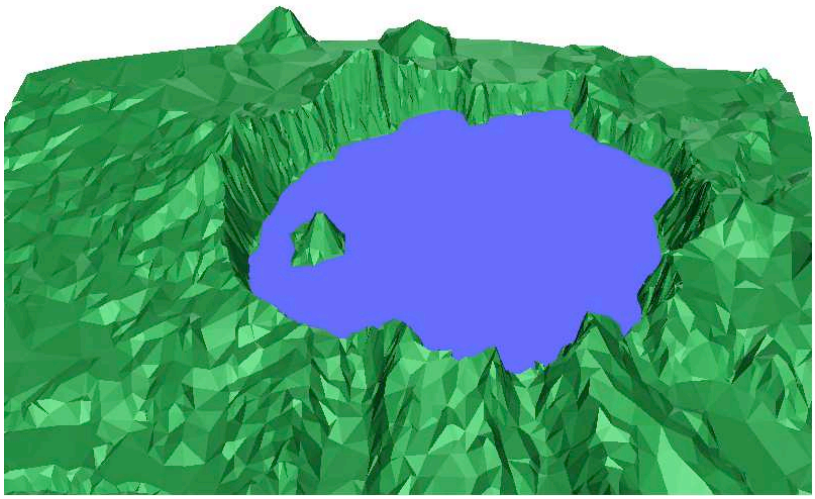


FIG. 4.21 – *Le modèle de terrain "crater"*

Chapitre 5

Maillages surfaciques

Ce chapitre aborde la compression de maillages surfaciques, c'est-à-dire de surfaces décomposées en mailles élémentaires polygonales, et plongées dans l'espace tridimensionnel euclidien. Ces structures géométriques sont sans doute les plus utilisées actuellement dans les innombrables applications de la synthèse d'image, et ce sont celles qui ont donné lieu au plus grand nombre de travaux dans le domaine de la compression géométrique.

Bien que ce mémoire soit consacré à la compression sans perte d'information, il nous paraît intéressant de montrer un exemple de reconstruction automatique de la connectivité d'un objet tridimensionnel à partir de sa géométrie. Dans ce cas, la conformité par rapport au modèle original n'est pas garantie, mais la figure 5.1 montre bien que le résultat, visuellement satisfaisant, peut s'avérer suffisant pour de nombreuses applications. Pour le modèle *triceratops* quantifié sur 12 bits par coordonnée, le coût de la géométrie seule (codée par l'algorithme du chapitre 3) est de 20,4 bits par sommet.

Dans ce chapitre, nous proposons des algorithmes permettant de décrire *explicitement* la connectivité des modèles afin de garantir un codage *sans perte* de la géométrie comme de la topologie. Ainsi, la figure 5.1 ci-dessus peut être comparée à la figure 5.22 représentant le même modèle avec un codage explicite de la connectivité originale, pour un coût total de 25,2 bits par sommet.

Dans un premier temps, la section 5.1 examine le problème du codage des maillages polygonaux quelconques, dans lesquels la maille élémentaire n'est pas nécessairement triangulaire. Nous montrons comment les algorithmes

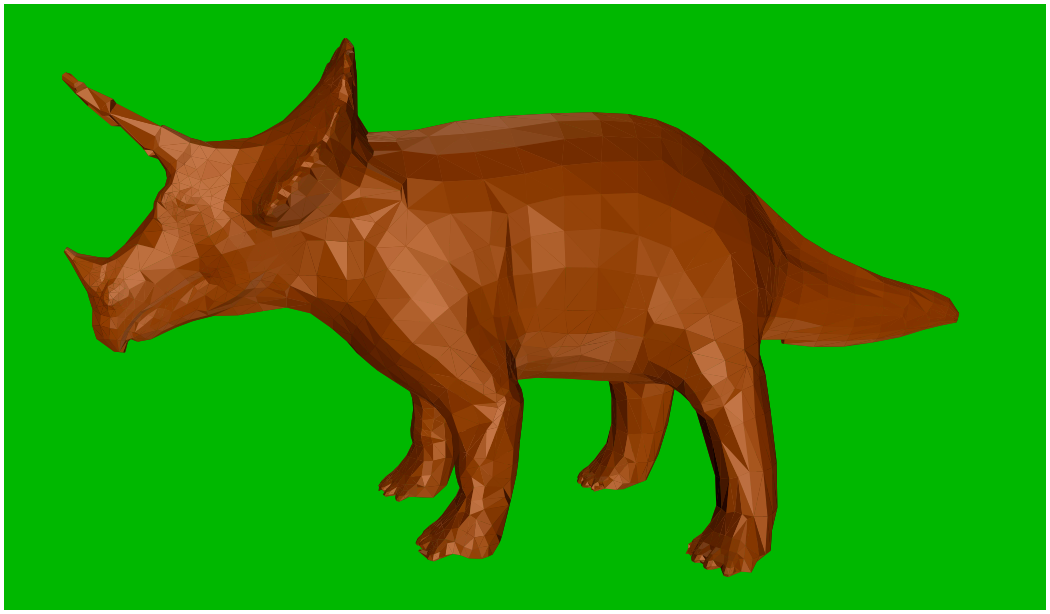


FIG. 5.1 – *Reconstruction du modèle* triceratops

présentés aux chapitres 3 et 4 peuvent être adaptés à des structures géométriques pour lesquelles, à notre connaissance, aucune méthode de compression progressive n'a été proposée jusqu'à maintenant. Le cas particulier des maillages surfaciques composés uniquement de triangles, beaucoup plus répandu en pratique, est traité séparément dans la section 5.2 : après une revue des différents travaux précédemment publiés, nous décrivons un nouvel algorithme de compression topologique spécialisé au cas des maillages triangulaires, et qui permet de réduire sensiblement la taille du code généré.

5.1 Maillages polygonaux arbitraires

On s'intéresse dans cette section au cas des maillages polygonaux généraux. S'il n'est pas rare en pratique de rencontrer des mailles polygonales de taille supérieure à 4 sur les modèles courants, la grande majorité des modèles tridimensionnels sont composés de triangles (on parlera dans la suite de *T-maillages*), de quadrilatères (*Q-maillages*), ou, très souvent, d'un mélange des deux (*QT-maillages*).

Le cas particulier très étudié des T-maillages sera traité à la section 5.2.

En ce qui concerne les Q-maillages et les QT-maillages, on peut citer notamment les applications suivantes :

- les simulations par éléments finis [78]
- la modélisation de déformation de tissu [105]
- la modélisation de phénomènes d'érosion géologique [32]

En outre, de nombreux systèmes de conception assistée par ordinateur (*CAD*) génèrent des QT-maillages où le nombre de triangles est relativement faible, et plusieurs travaux récents s'intéressent à la génération de Q-maillages irréguliers [13, 78].

Bien qu'il soit toujours possible de trianguler un maillage polygonal pour le ramener à un T-maillage, il est généralement plus intéressant du point de vue de l'encombrement mémoire de préserver sa topologie originale. Nous verrons dans la section 5.1.1 que l'information topologique des maillages principalement formés de quadrilatères peut être représentée par un code plus compact que celle de leur version triangulée. De plus, la conservation de la connectivité originale réalise une économie importante sur le codage des éventuels attributs associés aux faces du maillage. En effet, au cours de la pré-triangulation du modèle, les faces de taille supérieure à 3 sont subdivisées, ce qui implique la duplication de leur information de couleur, normale, texture, etc.

5.1.1 Précédents travaux

En l'absence de méthode pour la compression *progressive* de maillages polygonaux, nous décrivons ici deux méthodes non progressives qui nous donneront une base pour la comparaison avec notre algorithme. (Pour une vue d'ensemble sur les précédents travaux en compression de maillages surfaciques polygonaux, le lecteur est invité à se reporter à la section 2.3.)

Dans *Connectivity Compression for Irregular Quadrilateral Meshes*, King, Szymczak et Rossignac [65] développent une variante de leurs algorithmes *edgebreaker* et *wrap&zip* [89, 91] adaptée aux Q-maillages simples. Le principe est de scinder chaque quadrilatère selon une règle garantissant l'adjacence des deux triangles ainsi créés dans la séquence de description des faces construite par l'algorithme. Les quadrilatères sont donc reconstitués sans information supplémentaire, en joignant les paires de triangles adjacents pendant la décompression. Pour décrire l'arbre couvrant des faces, l'ensemble d'étiquettes

CLERS (cf figure 5.6) est étendu, après élimination des cas théoriquement impossibles, à un nouvel ensemble de 13 étiquettes représentant les combinaisons d'arêtes et de sommets visités et non visités. Deux techniques d'encodage sont proposées : la première, utilisant des codes de longueur variable, garantit un coût maximal de 3 bits par sommet dans le cas le pire, tandis que la seconde tire parti du codage entropique pour obtenir des coûts voisins de 0,6 bits par sommet dans le cas de maillages très réguliers. Ces résultats montrent que les Q-maillages sont intrinsèquement plus simples que les T-maillages contenant le même nombre de points, et plus économiques du point de vue du codage (cf la borne inférieure de 3,24 bits par sommet obtenue par Tutte [108] en énumérant les différents maillages constructibles à partir d'un ensemble de points). La méthode est ensuite étendue au cas plus général des QT-maillages contenant éventuellement des trous ou des anses.

Face Fixer : Compressing Polygon Meshes with Properties, de Isenburg et Snoeyink [55], propose un algorithme plus général que le précédent puisqu'il permet de coder la connectivité de maillages polygonaux arbitraires, c'est-à-dire contenant éventuellement des faces de taille supérieure à 4. Cet algorithme code une description entrelacée de l'arbre couvrant des sommets et de celui des polygones, à l'aide d'un jeu d'étiquettes appliquées aux arêtes du maillage. Chaque face de taille d du maillage engendre une étiquette F_d et chaque trou de taille t une étiquette T_t. Les étiquettes restantes associées aux arêtes (R, L, S et E) décrivent l'arbre couvrant des sommets correspondants et indiquent ainsi au décodeur comment reconstituer les faces et les trous du maillage. Une étiquette particulière est prévue pour la gestion des poignées (cf figure 5.2). La fréquence des différentes étiquettes étant non uniforme, la séquence se prête à un codage entropique efficace. En outre, une forte corrélation entre les étiquettes successives permet un gain supplémentaire par l'utilisation d'un codeur adaptatif d'ordre 3. Le coût maximal garanti pour le cas des Q-maillages est légèrement supérieur à ceux des autres méthodes (4 bits par sommet contre 3,5 pour Kronrod et Gotsman [69] et 3 pour King, Szymczak et Rossignac [65]), mais pour la première fois, des résultats sont présentés pour des maillages polygonaux contenant des faces de taille arbitraire et non uniforme (un résidu moyen de 2,4 bits par sommet est obtenu sur les modèles testés). De plus, la méthode permet de coder certaines informations structurelles, comme l'organisation du maillage en sous-groupes de faces.

Les autres travaux traitant de la compression de maillages polygonaux constituent les extensions de méthodes adaptées aux maillages triangulaires (cf 5.2.1), et de ce fait, ils n'exploitent pas la structure polygonale originale.

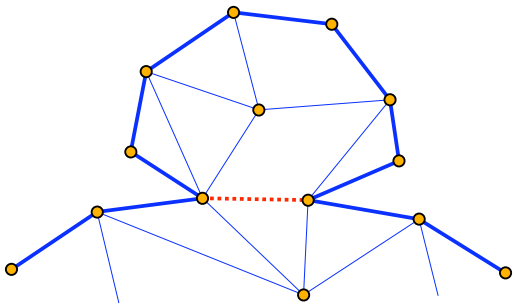


FIG. 5.2 – *Poignée dans un maillage polygonal*

Leur principe est de trianguler arbitrairement les polygones de taille supérieure à 3 et de coder le maillage ainsi obtenu en ajoutant une information supplémentaire décrivant les arêtes introduites lors de la phase de triangulation. Ce surcoût d'environ 2 bits par sommets, indispensable pour pouvoir rétablir la connectivité originale, fait que le code d'un maillage non triangulaire est plus volumineux que le code de sa version triangulée, alors qu'il contient moins d'information topologique [9, 103, 43].

Pour conclure sur l'ensemble des travaux consacrés au problème de la compression de maillages polygonaux non triangulaires, on retiendra que ces méthodes traitent uniquement de la compression topologique et ne proposent pas de technique nouvelle adaptée à la compression des positions. Le plus souvent, le lecteur est renvoyé aux méthodes de prédiction classiques utilisées sur les maillages triangulaires (cf section 5.2.1), et aucun résultat expérimental n'est présenté. Une autre limite commune à ces travaux est induite par le codage de l'objet en monorésolution. En effet, l'absence de structure hiérarchique dans le codage des arbres couvrant décrivant la connectivité du modèle rend impossible sa décompression et sa visualisation progressive.

5.1.2 Description de l'algorithme

Nous avons vu dans la section 5.1.1 que les méthodes de compression géométrique développées jusqu'à maintenant s'appuyaient sur la connaissance de l'information topologique de l'objet initial pour essayer de prédire sa géométrie et réaliser ainsi une compression sur les positions des sommets. L'originalité de la méthode exposée dans ce mémoire est d'obtenir des résidus de compression de la géométrie équivalents ou même meilleurs (cf section 5.2.4) sans le support de l'information topologique. C'est pourquoi cette mé-

thode est particulièrement adaptée aux modèles dont la topologie peut être automatiquement reconstruite.

Cependant, pour nombre d'objets tridimensionnels, une telle reconstruction automatique ne peut être réalisée sans perte d'information. En outre, les algorithmes de reconstruction ont généralement une complexité en temps supérieure à celle d'une décompression directe, et dans certains contextes, le surcoût peut être indésirable (le temps de décompression et de reconstruction effectif doit être maintenu en deçà du temps de transmission).

C'est pourquoi l'algorithme de compression topologique décrit à la section 4.4 et utilisé alors pour coder un sous-ensemble d'arêtes du modèle original, peut s'avérer utile pour coder l'intégralité de l'information topologique d'un modèle tridimensionnel. Cependant, il est important de noter que cet algorithme code l'information de connectivité minimale de la structure dans le sens où il ne gère que ses arêtes. Ainsi, un travail additionnel sera nécessaire pour reconstruire les facettes du maillage de départ, ou les relations d'incidence entre les simplexes dans le cas où l'application finale a besoin de ces informations.

5.1.3 Analyse théorique

La quantité de mémoire utilisée par cette méthode pour coder la topologie d'une structure géométrique dépend du degré moyen d'une cellule, c'est-à-dire du nombre moyen de super-arêtes incidentes à cette cellule. En effet, la subdivision d'une cellule ayant d voisins coûte exactement $d \log_2(3) + 1$ bits (s'il s'agit d'une "vraie" subdivision, i.e. d'une subdivision engendrant deux sous-cellules non vides). Or, il faut exactement $n - 1$ "vraies" subdivisions pour séparer les n points d'un objet géométrique, c'est-à-dire pour coder sa connectivité complète. Par suite, si le degré moyen d'une cellule est d, le codage progressif de sa topologie occupera environ $d \log_2(3) + 1$ bits par sommet.

La difficulté est d'évaluer ce degré moyen, qui ne doit pas être confondu avec le degré moyen de l'objet géométrique original : il s'agit du nombre moyen de voisins d'une cellule durant la phase de séparation de l'algorithme, qui est légèrement supérieur au degré moyen d'un sommet de l'objet codé.

5.1.4 Résultats expérimentaux

Nous présentons dans le tableau 5.1 les résultats de notre algorithme appliqué à quelques objets géométriques usuels. En l'absence de travaux comparables sur la compression géométrique (connectivité *et* positions) *progressive* de maillages polygonaux, nous donnons pour information les résultats obtenus par la méthode de compression *topologique monorésolution* proposée par Isenburg et Snoeyink [55]. Pour chaque modèle, la première ligne donne le nombre de bits par sommet pour le codage de la topologie, et la seconde le nombre de bits par sommet pour le codage de la géométrie (pour une quantification de 12 bits par coordonnée). Le tableau précise également la taille et le nombre de polygones qui composent chaque modèle.

modèles	nb de sommets	nb de triang.	nb de quad.	nb de $t \geq 5$	I & S 2000	notre algo.
beethoven	2655	680	2078	64	2,9 ?	7,7 20,7
cessna	3745	900	2797	250	2,8 ?	7,6 15,8
cupie	2984	384	2506	142	2,3 ?	7,8 18,1
galleon	2372	336	1947	101	2,6 ?	8,5 21,8
triceratops	2832	346	2266	222	2,1 ?	7,2 18,3
moyenne	14588	2646	11594	779	2,5 ?	7,7 18,6

TAB. 5.1 – *Banc d'essai sur la compression de maillages polygonaux*

5.2 Maillages triangulaires

Cette section est consacrée au codage progressif sans perte de maillages surfaciques triangulaires. Nous verrons, dans la section 5.2.1, 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 accélératrices 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.

5.2.1 Précédents travaux

Une vue d'ensemble sur les précédents travaux en compression de maillages surfaciques triangulaires a été proposée à la section 2.2. Nous revenons maintenant de manière plus approfondie sur quelques-unes des méthodes qui ont jalonné le domaine depuis 1995. Certains algorithmes de codage monorésolution sont abordés pour des raisons historiques et à des fins de comparaison de performances, mais ce sont bien sûr les méthodes progressives qui nous intéressent plus particulièrement dans le cadre de ce mémoire.

Compression monorésolution

Avec *Geometry Compression*, Deering [26] introduit le concept de compression géométrique. L'objectif de l'auteur est double : obtenir une représentation des maillages triangulaires à la fois *compacte* et compatible avec une décompression et un affichage *rapide* par les accélérateurs graphiques. L'algorithme utilise des bandes de triangles généralisées, une représentation qui permet de spécifier implicitement un triangle à chaque occurrence d'un sommet dans la séquence codante : le sommet courant engendre un nouveau triangle, connecté par la droite ou par la gauche au dernier triangle construit. Un bit additionnel est donc nécessaire pour préciser la position relative du nouveau triangle (cf figure 5.3). Puisqu'un sommet appartient à deux bandes, cette construction implique que chaque sommet du maillage apparaisse en moyenne deux fois dans la séquence. A la première occurrence, le sommet est complètement spécifié (position et attributs éventuels), tandis qu'à la seconde, un pointeur sur la première occurrence s'avère plus économique. L'algorithme étant destiné à des accélérateurs graphiques dont la mémoire est restreinte, l'auteur propose de maintenir un tampon limité aux 16 derniers sommets codés. Dans le cas ou le sommet courant constitue la seconde occurrence d'un sommet qui n'est plus dans le tampon, une nouvelle description complète est nécessaire. Cette technique conduit à un codage de la connectivité sur $7{,}5 + \log_2 n/8$ bits par sommet. La compression des positions et des attributs est traitée par codage différentiel, et par l'utilisation de tables pour les normales, préalablement quantifiées.

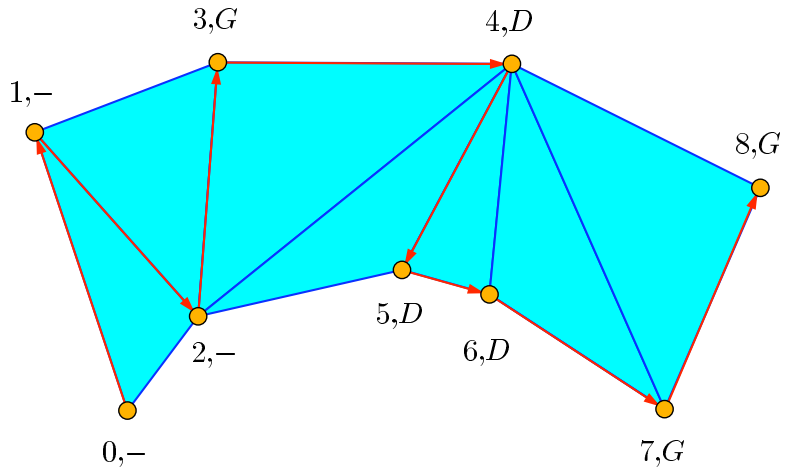


FIG. 5.3 – *Bande de triangles généralisée*

Dans *Geometric Compression Through Topological Surgery*, Taubin et Rossignac [104] représentent la connectivité d'un maillage triangulaire par le codage explicite de deux arbres entrelacés : un arbre couvrant de sommets et un arbre couvrant de faces. Le découpage du maillage suivant les arêtes de l'arbre des sommets engendre un maillage simple sans sommet intérieur (i.e. composé d'une ou plusieurs bandes connectées), qui peut donc être représenté par un arbre de triangles. Sur des maillages très réguliers, le coût de cette double représentation peut être inférieur à 2 bits par sommet, mais il n'est pas borné dans le cas de topologies plus complexes. Les positions sont codées à l'intérieur de la séquence ordonnée représentant l'arbre des sommets, par un prédicteur linéaire dont l'expression générale est :

$$P(\lambda, s_{n-1}, ..., s_{n-K}) = \sum_{i=1}^{K} \lambda_i \, s_{n-i} \, ,$$

où s_n désigne le sommet courant et les s_{n-i} sont les sommets précédents dans la séquence. Le paramètre $\lambda = (\lambda_1, ..., \lambda_K)$ (où les λ_i sont des entiers) est optimisé par une méthode de moindres carrés minimisant l'erreur de prédiction sur les sommets précédemment codés. Les résidus sont ensuite passés à un codeur entropique. Cette méthode de compression géométrique a été intégrée à la norme de compression actuelle MPEG-4.

Triangle Mesh Compression, de Touma et Gotsman [106], décrit un algorithme dont le principe général est assez proche. La première différence est la manière de parcourir la triangulation. L'algorithme maintient une liste L de sommets (initialisée avec un sommet arbitraire du maillage) formant un polygone qui contient tous les triangles déjà codés (en jaune sur la figure 5.4). L est parcourue dans le sens direct, et chacun de ses sommets (les disques verts) devient *pivot* (le disque plus grand) à tour de rôle. Le polygone grandit par conquête de tous les triangles incidents au pivot qui ne sont pas déjà codés (en blanc sur la figure). Cette conquête détermine un ordre sur les sommets du maillage qui permet de reconstruire sa topologie avec peu d'information supplémentaire : chacun des sommets de la séquence est accompagné de son degré et de l'une des 3 commandes *ajouter*, *séparer* et *fusionner*. Ces commandes permettent de gérer les changements de topologie de L : rebouclage et séparation de listes, fusion de listes. La seconde différence avec l'algorithme précédent est la méthode utilisée pour prédire la position d'un sommet à partir de ses prédécesseurs dans le code. En plus d'un schéma prédictif linéaire, l'algorithme estime le pli entre les triangles courants en fonction des plis précédents, ce qui conduit en pratique à des résidus de compression inférieurs. En ce qui concerne la topologie, le codage entropique

des degrés (dont la distribution est fortement groupée autour de 6 comme le montre la figure 5.5) fournit un coût final de l'ordre de 2 bits par sommet sur des maillages usuels, et qui tend vers 0 pour des maillages très réguliers. Sur ce dernier type de modèles, les performances de la méthode semblent être à ce jour les plus compétitives pour la compression monorésolution de surfaces triangulées.

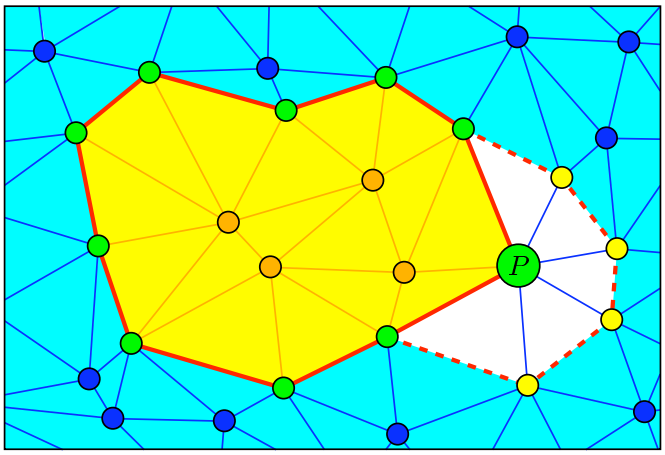


FIG. 5.4 – *Codage par conquête sur les triangles*

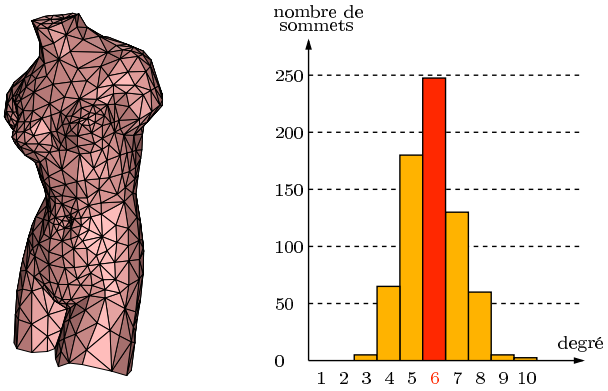


FIG. 5.5 – *Distribution des degrés des sommets d'un maillage usuel*

Dans *Guaranteed* 3,67 *V Bit Encoding of Planar Triangle Graphs*, King et Rossignac [63] décrivent un codage des étiquettes produites par l'algorithme *edgebreaker* de Rossignac [89], garantissant une borne sur le coût dans le cas le pire. Le codeur *edgebreaker*, conçu pour comprimer la connectivité de maillages triangulaires homéomorphes à une sphère, réalise une conquête des

triangles par un parcours en profondeur. Chaque triangle donne lieu à une commande (ou étiquette) indiquant au décodeur comment le rattacher au reste du maillage. La figure 5.6 illustre les 5 commandes C, L, E, R, S utilisées : le triangle courant est en jaune, les triangles déjà codés en orange, et les triangles non codés en blanc ; le disque vert représente un point qui n'est pas encore conquis. Les codes des étiquettes sont de longueur variable et adaptés au contexte (codage adaptatif d'ordre 1). Pour pouvoir garantir un coût maximal de $\frac{11}{6}$ bits par triangle (soit moins de 3,67 bits par sommet), trois dictionnaires différents sont proposés dont l'un au moins permet d'atteindre ce coût garanti pour un maillage quelconque donné. Les surfaces avec bord sont codées par l'utilisation d'un sommet fantôme, et le cas des maillages avec trous ou poignées est traité par l'adjonction d'un paramètre à la commande S qui subdivise la liste de conquête et place un nouveau triangle dans la pile (en rouge sur la figure). L'algorithme de décompression initial, d'une complexité asymptotique en $O(n^2)$ dans le cas le pire, peut être avantageusement remplacé par l'algorithme linéaire de Rossignac et Szymczak [91] ou celui de Isenburg et Snoeyink [56]. S'il est vrai que la borne théorique obtenue est moins compétitive que les résultats de Touma et Gotsman [106] sur des modèles usuels, la méthode présente l'avantage de garantir un coût maximal quelle que soit la régularité du maillage à comprimer.

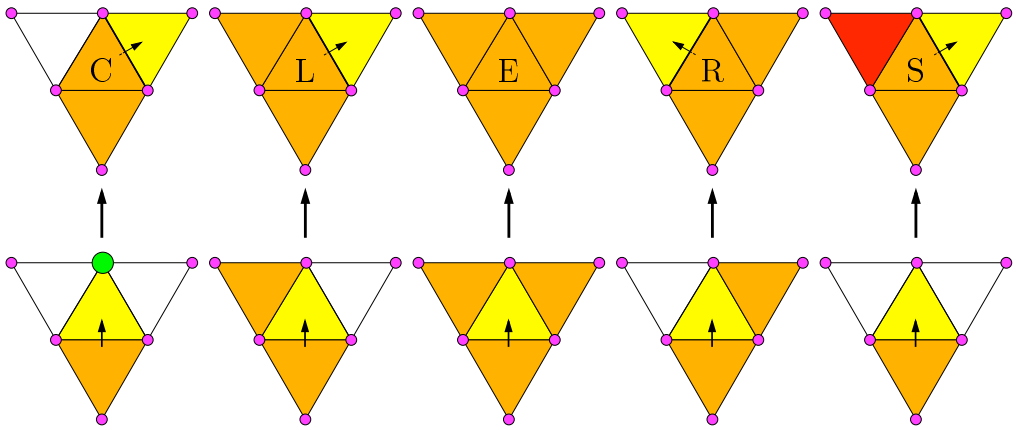


FIG. 5.6 – *Les* 5 *étiquettes de l'algorithme* edgebreaker

Alliez et Desbrun, dans leur récent article *Valence-Driven Connectivity Encoding for 3D Meshes* [5], améliorent encore les excellentes performances de Touma et Gotsman [106] par un ensemble d'heuristiques de conquête se greffant à l'algorithme original : en rendant la conquête *adaptative* (et non plus déterministe), le nombre de séparations de liste de conquête est minimisé ; en outre, la taille des codes décrivant ces séparations (et les fusions

correspondantes) est réduite en exploitant l'information géométrique associée aux sommets ; enfin, l'utilisation d'un unique sommet fantôme connecté à tous les bords rend le codage des objets à bords multiples plus efficace. Ces optimisations permettent d'obtenir de surcroît une borne supérieure de 3,5 bits par sommet pour des maillages *manifolds arbitraires*

Compression progressive

Avec *Progressive Forest Split Compression*, Taubin, Guéziec, Horn et Lazarus [102] proposent une extension naturelle de la méthode de compression monorésolution de Taubin et Rossignac [104]. La méthode repose sur un opérateur de raffinement *lisse* qui agit par subdivision d'une forêt dans le graphe des sommets et des arêtes du maillage. L'application de cet opérateur équivaut à regrouper en paquet des opérations élémentaires de raffinement (ou de décimation, dans le sens du codage). Cela permet une compression plus efficace des niveaux de détails, au prix d'une granularité moins fine, c'est-à-dire d'un nombre de niveaux restreint (le nombre de triangles est approximativement doublé à chaque itération). L'opérateur est défini par une forêt de sommets et d'arêtes, une séquence de polygones simples, et une séquence de déplacements de sommets. Le maillage courant est découpé le long des arêtes de la forêt, les bords sont dupliqués et écartés, puis chaque polygone simple ainsi obtenu est triangulé en fonction des informations contenues dans la séquence codante. Enfin, les positions des nouveaux sommets sont corrigées par une séquence de résidus. Sur les 4 modèles testés par les auteurs, les coûts oscillent autour de 9 bits par sommet pour la connectivité. Pour le coût total (topologie et géométrie), le surcoût moyen par rapport à la méthode non progressive originale correspond environ à un facteur 2.

Li et Kuo, avec *Progressive Coding of 3D Graphic Models* [73], sont les premiers à proposer un codage doublement progressif de maillages triangulaires : les informations séquentielles contenues dans le flux de données produit par l'algorithme raffinent parallèlement la connectivité *et* les positions, contribuant ainsi à réduire progressivement la distorsion par rapport au modèle original. Deux séquences sont construites par l'algorithme. La première, qui code la topologie, est obtenue par des suppressions successives de sommets (cf section 2.5.1). Le trou polygonal créé par la suppression d'un sommet est retriangulé, et codé par un indice dans une table de motifs, afin que le décodeur connaisse le voisinage du sommet à insérer (pour un coût total de $\log_2 n + 6$ bits par sommet). En ce qui concerne la géométrie, la position du nouveau sommet est prédite par l'isobarycentre de ses voisins (i.e. les sommets sur

le bord du trou polygonal). La seconde séquence, codant la géométrie, est donc formée des résidus de prédiction quantifiés et organisés en plans de bits : les résidus sont normalisés et transmis bit à bit, dans l'ordre des poids décroissants. La dernière étape de l'algorithme consiste à entrelacer ces deux séquences en tenant compte d'un critère de proximité au maillage initial : à chaque étape de la fusion, la priorité est donnée à la séquence qui apporte l'information la moins pertinente. Ainsi, au décodage, l'innovation du flux est décroissante, ce qui permet d'obtenir rapidement une bonne approximation du modèle.

Dans *Progressive Compression of Arbitrary Triangular Meshes*, Cohen-Or, Levin et Remez [23] utilisent également la simplification séquentielle de maillage par suppression de sommet, mais la technique proposée pour coder les trous polygonaux s'avère plus efficace. L'algorithme colorie chaque triangle, et les trous (retriangulés) où les nouveaux sommets doivent être insérés sont définis par les triangles adjacents de même couleur. Les auteurs montrent que 4 couleurs suffisent, quitte à s'interdire certaines suppressions de sommet, pour que deux trous adjacents n'aient pas la même couleur. Une variante est proposée où 2 couleurs suffisent si les trous sont tous retriangulés par une bande triangulaire suivant le dessin de la lettre Z : pour chaque trou, la couleur 0 (le jaune sur la figure 5.7) est affectée à 2 triangles externes, et la couleur 1 (le bleu ciel) aux triangles internes. En moyenne, cette technique de coloration permet de coder la connectivité de maillages réguliers par 6 bits par sommet. La compression de la géométrie, non progressive, se fait par prédiction à partir des sommets du trou polygonal et codage entropique des résidus. Dans le domaine de la compression géométrique progressive, cet algorithme se positionne avantageusement, puisqu'il induit une augmentation du code de seulement 10% par rapport aux résidus de compression obtenus par la méthode monorésolution de Touma et Gotsman [106].

Dans *Compressed Progressive Meshes*, Pajarola et Rossignac [82] exploitent l'opérateur de contraction d'arête utilisé par Hoppe [50] (cf section 2.5.1) pour simplifier progressivement le maillage initial. Afin d'augmenter l'efficacité du codage, les simplifications sont regroupées en lots : à chaque étape, le nombre de sommets est réduit d'environ 30% par contraction d'un ensemble d'arêtes indépendantes. La séquence permettant au décodeur d'inverser la simplification (par l'opérateur d'expansion d'arête) est organisée comme suit : i) un bit de marquage est affecté à chaque sommet du maillage simplifié pour préciser s'il est le résultat d'une contraction ou non, ii) un code indiquant les deux arêtes à éclater pour rétablir l'arête contractée et ses faces adjacentes est affecté aux sommets marqués, iii) deux résidus de prédiction sont associés

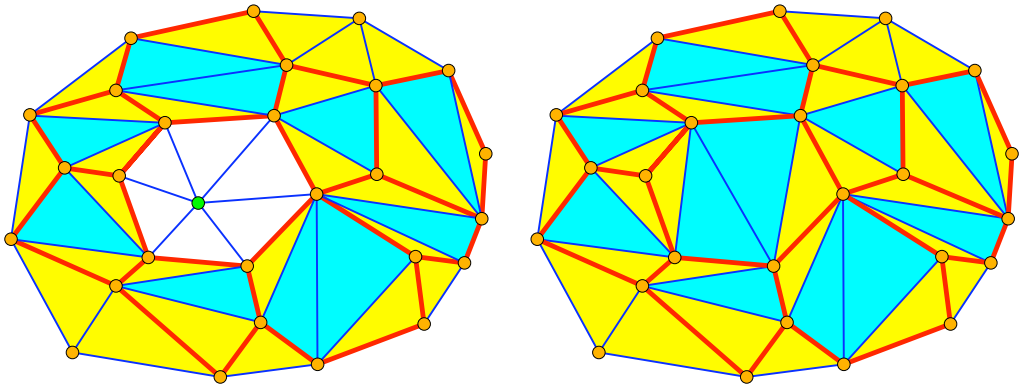


FIG. 5.7 – *Codage par 2-coloration des trous polygonaux retriangulés*

à chaque sommet marqué. Ces résidus corrigent les positions des sommets de l'arête restaurée qui ont été prédites à partir du sommet subdivisé et de ses deux arêtes marquées (cf figure 5.8). Le coût de cette méthode est d'environ 8 bits par sommet pour le codage progressif de la connectivité complète du modèle (3 bps pour marquer les sommets et 5 bps pour marquer les deux arêtes incidentes). Pour une quantification de 10 bits par coordonnée, le coût de la géométrie s'élève en moyenne à 20 bits par sommet.

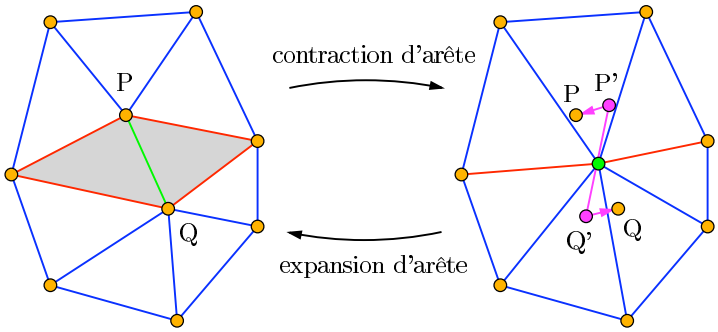


FIG. 5.8 – *Expansion d'arête et prédiction des positions*

Progressive Compression and Transmission of Arbitrary Triangular Meshes, de Bajaj, Pascucci et Zhuang [10], présente une description multirésolution capable de représenter des maillages de topologie complexe (genre ou nombre de composantes connexes élevés, surfaces non *manifold*) de manière relativement efficace. La structure de base est celle développée par les auteurs dans leurs travaux en compression non progressive [8, 11] : le maillage est partitionné en couches de sommets et en bandes de triangles généralisées,

définies par deux couches de sommets successives. La progressivité est assurée par un schéma de simplification séquentielle qui fait intervenir 3 opérateurs. Un opérateur *intra-bande* procède par suppression de sommet et retriangulation à l'intérieur de la bande de triangles, et un opérateur *inter-bandes* supprime un contour entier (i.e. une portion cyclique d'une couche de sommets) et fusionne les deux bandes de triangles adjacentes par retriangulation. Lorsque le maillage a été simplifié au maximum par ces deux primitives, l'algorithme utilise un opérateur de contraction de triangles, plus lourd à coder mais qui permet de décrire des changements de topologie. Ainsi, le niveau le plus grossier de la hiérarchie peut contenir un nombre de sommets arbitrairement petit quelle que soit la complexité du modèle de départ, ce qui n'était pas le cas des méthodes précédentes, où la topologie était fixée pour tous les niveaux de détail. La géométrie est codée par des techniques de prédiction adaptées aux opérateurs (interpolation par courbes de Bézier pour les sommets insérés sur les contours, prédiction linéaire pour les sommets insérés à l'intérieur des bandes). Le coût global engendré est légèrement supérieur à celui de la méthode de Taubin, Guéziec, Horn et Lazarus [102] sur les mêmes modèles.

Compression et Représentation Echelonnable de Maillages Triangulaires, de Alliez et Laurent [6], décrit une méthode qui comprime la topologie de manière non progressive mais très compacte, puis code les positions progressivement, par une technique utilisant des plans de bits pour les résidus de prédiction. Le codage de la connectivité repose sur un parcours des sommets et des faces par une pile de listes de conquête contenant essentiellement les degrés des sommets. Une heuristique est introduite pour optimiser le choix du sommet pivot (les sommets complets, c'est-à-dire dont toutes les faces incidentes ont été conquises, sont favorisés), ce qui permet d'améliorer sensiblement les résultats obtenus par l'algorithme de Touma et Gotsman [106]. Les positions sont prédites à partir des sommets précédemment transmis en combinant 4 techniques différentes : en fonction de l'organisation locale des sommets, le codage se fera différentiellement, par prédiction linéaire, par prédiction de type parallélogramme, ou par un procédé adapté aux sauts d'adjacence. La progressivité est introduite au moment du codage des résidus par plans de bits : la méthode consiste à raffiner la précision sur les positions en transmettant un bit supplémentaire à chaque étape pour l'ensemble des résidus. Le codage entropique de la position du premier bit significatif de chaque résidu permet d'accroître les résultats. Cet algorithme nous semble être actuellement le plus efficace en terme de performances lorsque l'application finale ne nécessite pas de progressivité topologique.

Dans *Progressive Compression for Lossless Transmission of Triangle Meshes*, Alliez et Desbrun [4] proposent un algorithme de compression progressive de maillages *manifolds* basé sur la décimation de sommets indépendants par vagues successives. Le codage se décompose en trois phases : la décimation principale, la retriangulation des trous engendrés, et la régularisation du maillage par décimation sélective. L'idée générale qui préside à chacune de ces phases est de grouper autant que possible les degrés des sommets du nouveau maillage autour d'une valeur moyenne égale à 6. Au cours de la phase de décimation principale, l'algorithme procède par conquête de sommets indépendants (cf figure 5.9), en marchant dans le maillage à partir d'une arête germe, sélectionnant les sommets rencontrés lorsque leur degré est inférieur ou égal à 6 et que leur suppression ne viole pas le caractère *manifold* du modèle (une troisième condition liée à une métrique d'erreur permettant de préserver certaines propriétés du maillage peut être éventuellement ajoutée). Chaque sommet sélectionné pour la suppression engendre un code correspondant à la valeur de son degré ; dans le cas ou le sommet rencontré ne satisfait pas les contraintes, un code spécial (*null_patch*) est transmis. Après la suppression des sommets choisis, les trous polygonaux (ou *patchs*) obtenus sont retriangulés en partant d'une arête de bord (le *portail*), et ce de manière complètement déterministe grâce à un ensemble de marqueurs associés aux sommets de bord. Ces marqueurs, initialisés lors de la phase de décimation, permettent au décodeur de localiser le bord du trou à partir du portail et du degré du sommet à insérer. La phase de décimation sélective est indispensable pour maintenir la régularité du maillage : elle transforme les sommets de degrés 3 et 9 créés à l'issue des deux premières phases en sommets de degré 6. La conquête est similaire à celle de la décimation principale, mais cette fois, seuls les sommets de degré 3 sont sélectionnés, et la suppression ne donne donc lieu à aucune retriangulation. Dans le cas favorable de maillages très réguliers, les auteurs montrent que les codes *null_patch* sont rares et que la séquence codante est essentiellement constituée d'un degré par sommet. Les valeurs des degrés étant fortement groupées autour de 6, un codeur entropique donne d'excellents résultats finaux (moins de 3,7 bits par sommet en moyenne pour des maillages usuels). Le codage des positions, complètement indépendant du codage de la connectivité, est réalisé par une prédiction barycentrique à partir des sommets du bord du *patch*. Afin d'obtenir de meilleurs résultats par codage entropique, les auteurs opèrent une décomposition préalable du vecteur déplacement entre la position prédite et la position réelle, en ses composantes normale et tangentielle. Globalement, cette méthode semble fournir les meilleurs résultats publiés jusqu'ici pour la compression progressive de surfaces triangulées.

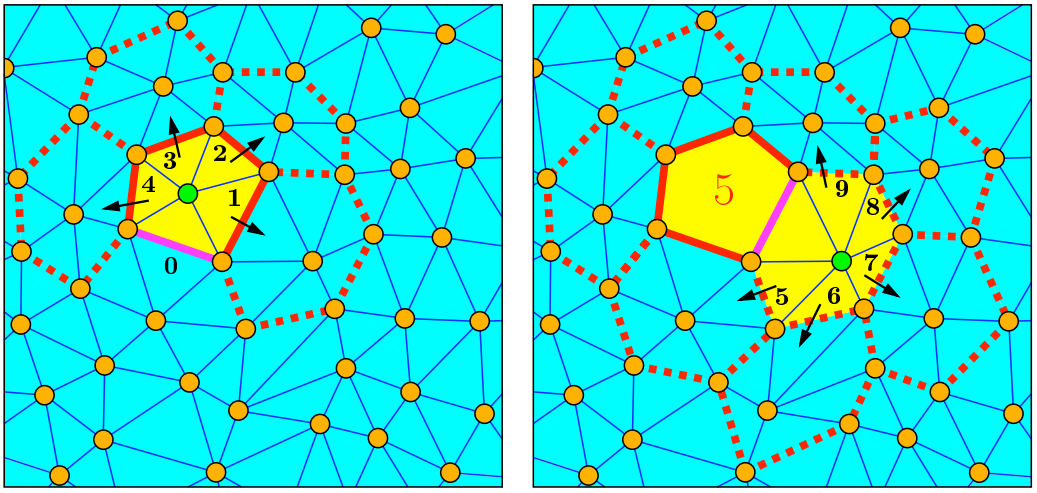


FIG. 5.9 – *Conquête sur les sommets indépendants*

5.2.2 Description de l'algorithme

Le codeur topologique que nous proposons dans cette section s'inspire des procédés utilisés en simplification séquentielle de maillages. On se reportera à la section 2.5.1 pour une présentation des méthodes classiques de *décimation*.

La contrainte principale imposée à notre codeur topologique a été dictée par le codeur géométrique. Pour pouvoir comprimer 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 cellule 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' 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. A chaque étape, l'information nécessaire à la restauration de la connectivité des cellules originales est codée.

A 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 énumérées à la section 2.5.1 :

- la contraction d'arête permet de fusionner deux cellules incidentes (cf figure 5.10) (nous appellerons sa primitive inverse l'expansion d'arête),
- la fusion de sommets permet de fusionner deux cellules non incidentes (cf figure 5.11) (nous appellerons sa primitive inverse la subdivision de sommet).

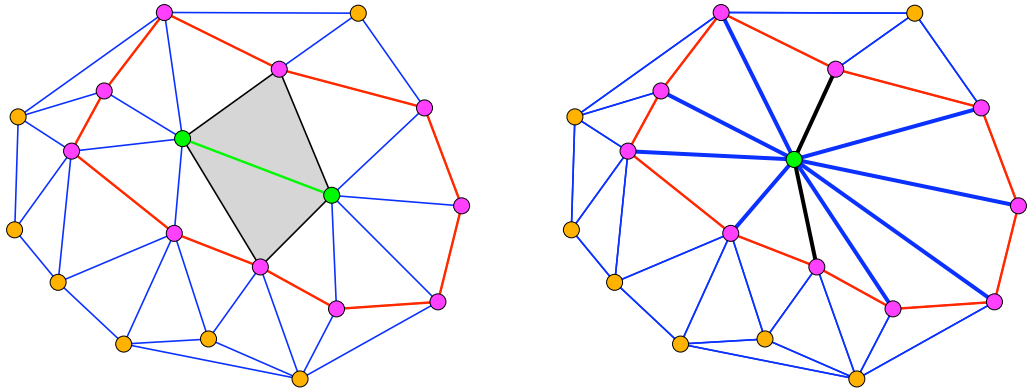


FIG. 5.10 – *La contraction d'arête*

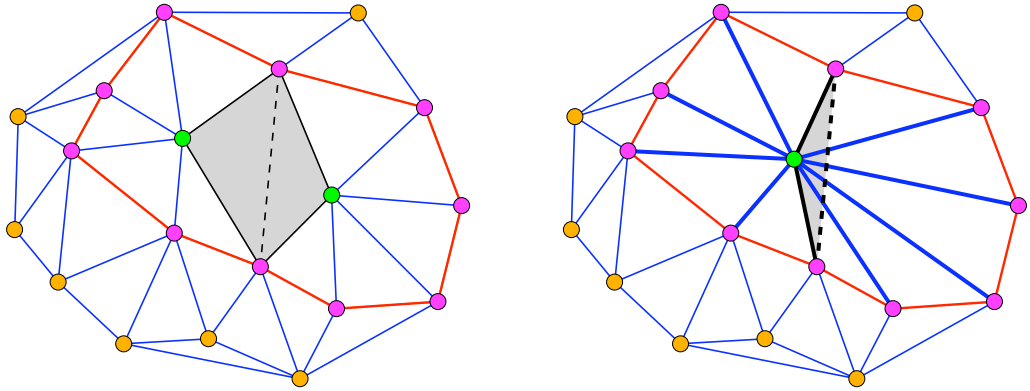


FIG. 5.11 – *La fusion de sommets*

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 appliquée. A 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 agi sur la connectivité originale.

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 *manifold* et *orientable*. Dans ces conditions, la contraction d'arête provoque la disparition de ses 2 faces adjacentes (en gris sur la figure 5.10) par dégénérescence, et l'information nécessaire à l'opérateur inverse se limite aux indices des 2 arêtes à éclater (SV_2 et SV_7 sur la figure 5.12), parmi l'ensemble des arêtes incidentes à la cellule fusionnée (SV_1 à SV_{10}).

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

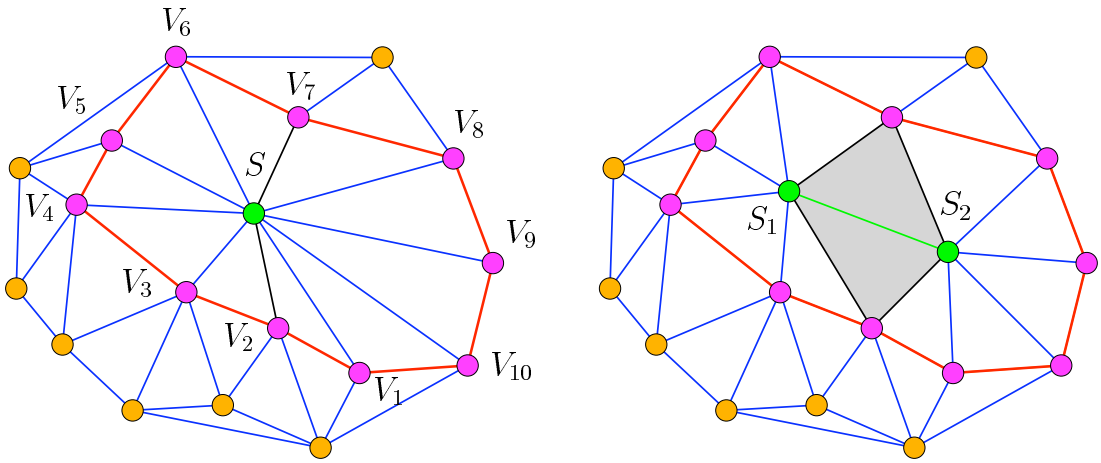


FIG. 5.12 – *L'expansion d'arête*

de prédiction permet d'atteindre des coûts inférieurs à 3 bits sur certains modèles.

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* [85]. L'utilisation de cette opération permet de simplifier (et, dans le cadre de ce mémoire, 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 comparables en compression ou simplification de maillages.

Le principe général de la fusion de sommets revient à coder de manière exhaustive l'évolution des simplexes (c'est-à-dire des sommets, des arêtes et des triangles) 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 $s' = (s \setminus v_2) \cup v_1$. Lorsque s' 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 entraîné 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 [85]. 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 moyen d'environ 8 bits par fusion de sommet.

5.2.3 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. Le tableau 5.2 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%	23,6%	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

TAB. 5.2 – *Pourcentage de contractions d'arêtes / fusions de sommets*

5.2.4 Résultats expérimentaux

Nous présentons dans le tableau 5.3 quelques résultats de notre algorithme comparés à ceux de Pajarola et Rossignac [83], Cohen-Or, Levin et Remez [23] et Alliez et Desbrun [4]. Nous faisons également apparaître à titre de comparaison les résultats obtenus en compression **non progressive** par Touma et Gotsman [106]. Pour chaque modèle et pour chaque algorithme testé, la première ligne donne le nombre de bits par sommet pour le codage de la topologie, et la seconde le nombre de bits par sommet pour le codage de la géométrie. Comme le montre la ligne correspondant aux résultats moyens, le surcoût global de notre méthode par rapport à la méthode non progressive de Touma et Gotsman (qui est actuellement la plus efficace) est inférieur à 5%. Toutefois, cette remarque doit être nuancée à cause des données non disponibles (les? qui apparaissent dans le tableau). D'une manière générale, la comparaison des performances des différents travaux est rendue très difficile par la grande disparité des modèles géométriques utilisés lors des tests quantitatifs. Comme nous n'avons pas reprogrammé les différentes méthodes, les objets testés dans le tableau 5.2.4 constituent l'intersection des modèles figurant dans les articles de Touma et Gotsman [106], Pajarola et Rossignac [83], Cohen-Or, Levin et Remez [23] et Alliez et Desbrun [4].

Remarque 5.2.1 *Le lecteur aura sans doute remarqué que le modèle* triceratops *était déjà présent dans le tableau 5.1 concernant les maillages polygonaux arbitraires. En fait, il s'agit ici d'une version du modèle fréquemment utilisée, où les polygones de taille supérieure à* 3 *ont été triangulés. Il est paradoxal de constater que le coût de la connectivité est moindre pour cette version triangulée (*6,0 *bits par sommet contre* 7,2 *pour le maillage polygonal), alors que l'information topologique qu'elle contient est plus riche (dans le sens où le nombre d'arêtes à coder est plus grand). Cela montre la non*

modèles	nombre de sommets	T & G 1998	P & R 2000	C L R 2000	A & D 2001	notre algo.
triceratops	2832	2,2 20,0	7,4 21,0	5,8 20,4	?	6,0 19,2
blob	8033	1,7 20,0	5,9 21,0	7,6 19,7	?	4,1 20,1
fandisk	6475	1,1 9,0	6,8 15,0	?	5,0 12,3	2,9 12,1
bunny	35947	?	7,0 16,0	?	?	3,2 14,7
horse	19851	2,3 17,0	?	5,7 15,4	4,6 16,2	3,9 16,4
moyenne	73138	2,0 16,5 18,5	7,1 16,9 24,0	5,8 17,0 22,8	4,7 15,2 19,9	3,6 15,7 19,3

TAB. 5.3 – *Banc d'essai sur la compression de modèles 3D*

optimalité de notre codeur polygonal.

Les figures 5.13 à 5.22 permettent de visualiser le décodage progressif du modèle *triceratops*. Pour chaque résolution, seules les facettes sont affichées ; les arêtes isolées qui peuvent apparaître lorsque la résolution est basse sont éliminées pour améliorer le rendu.

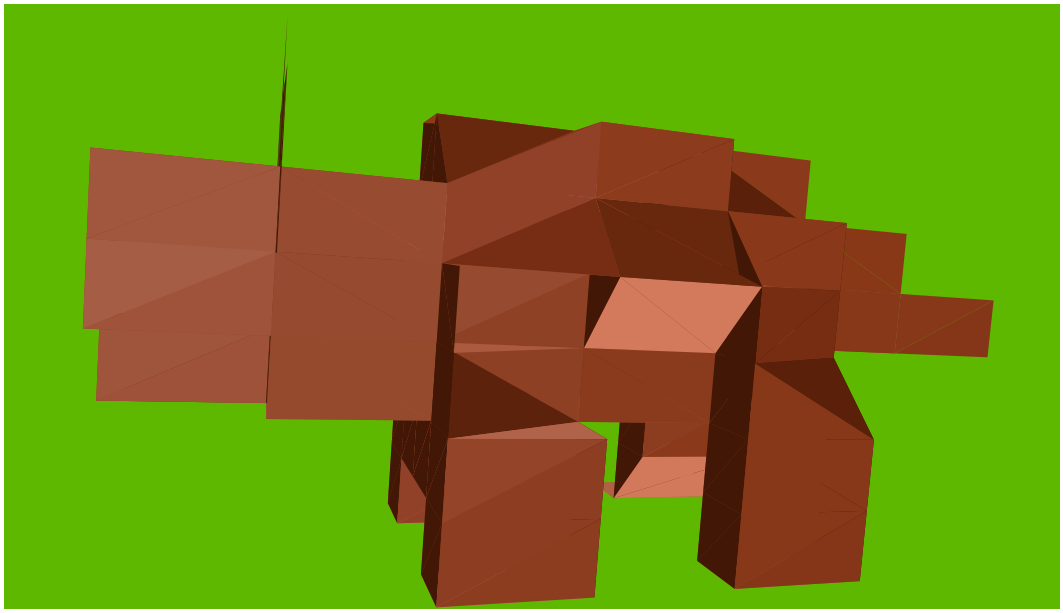


FIG. 5.13 – *Précision* = 3 *bits, Résidu de compression* = 0,4%

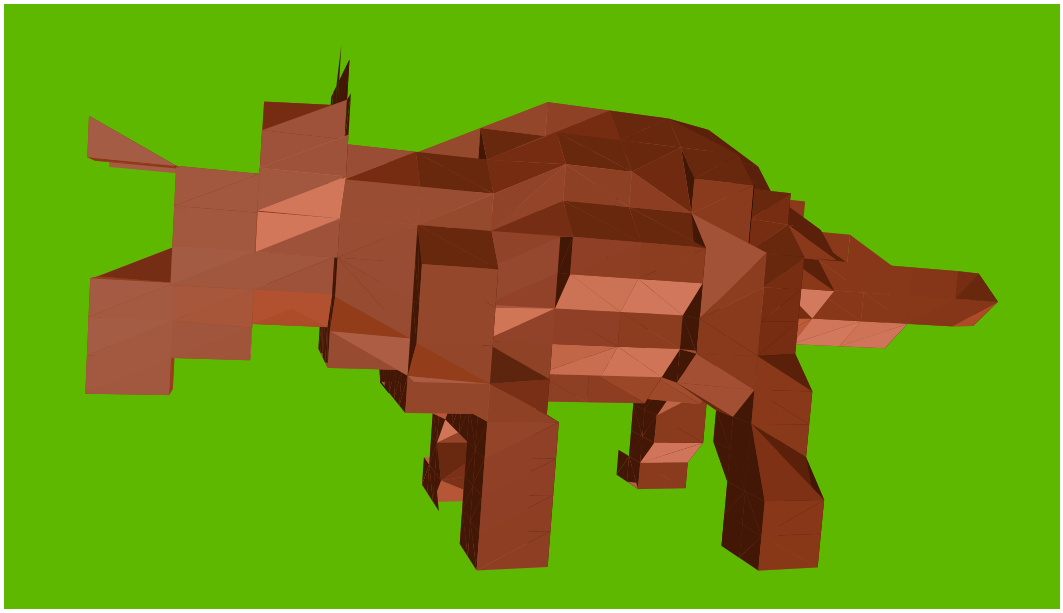


FIG. 5.14 – *Précision* = 4 *bits, Résidu de compression* = 1,2%

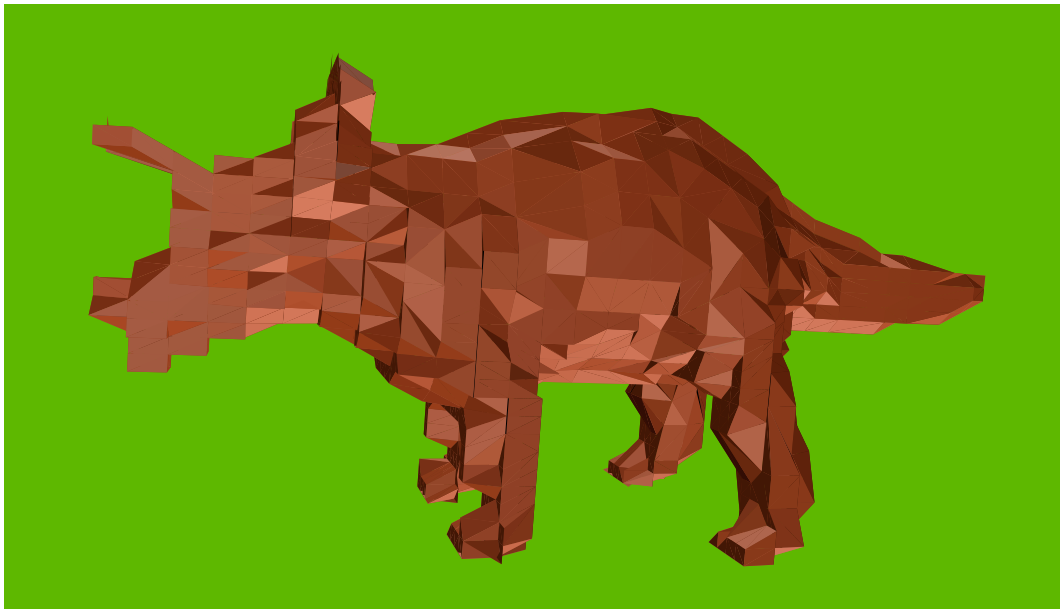


FIG. 5.15 – *Précision* = 5 *bits, Résidu de compression* = 3,0%

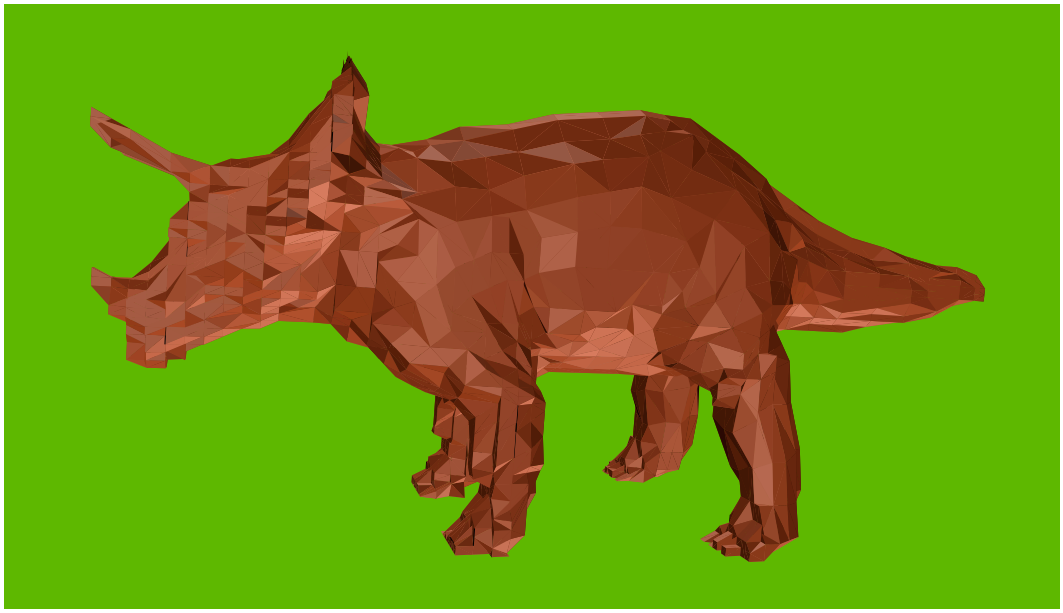


FIG. 5.16 – *Précision* = 6 *bits, Résidu de compression* = 5,5%

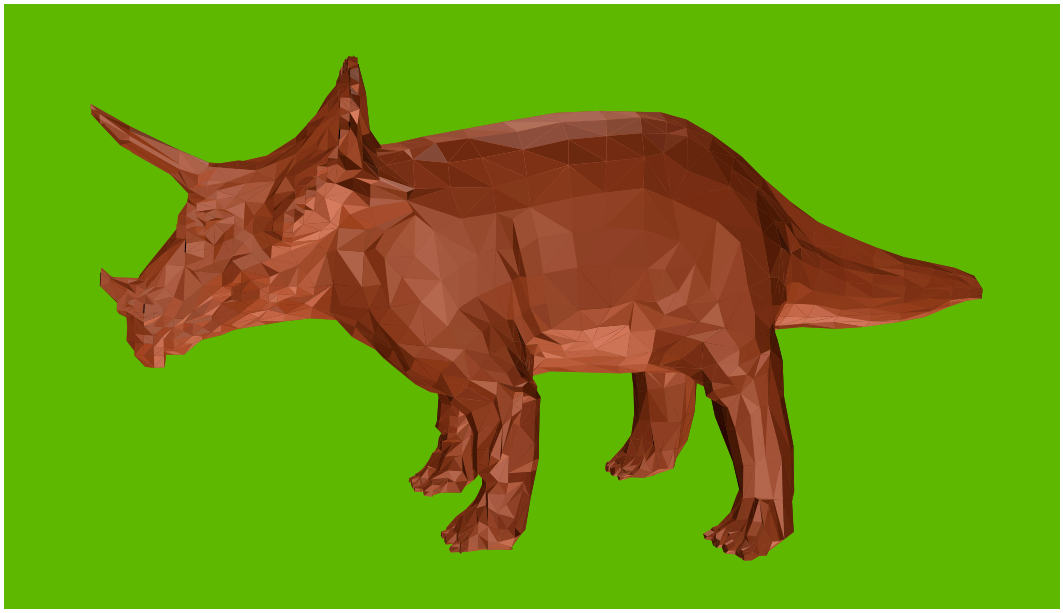


FIG. 5.17 – *Précision* = 7 *bits, Résidu de compression* = 8,1%

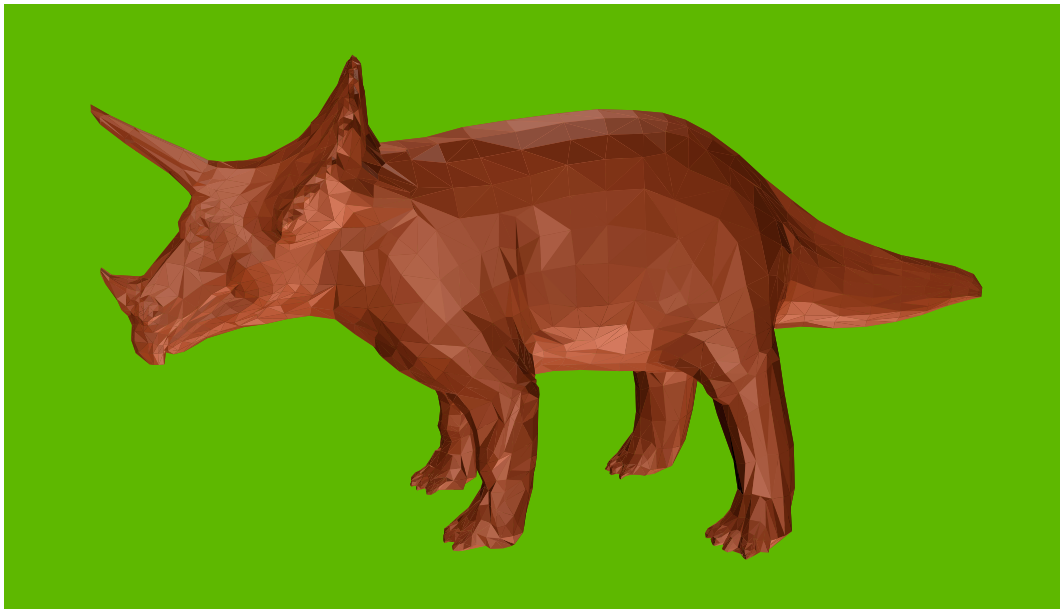


FIG. 5.18 – *Précision* = 8 *bits, Résidu de compression* = 10,3%

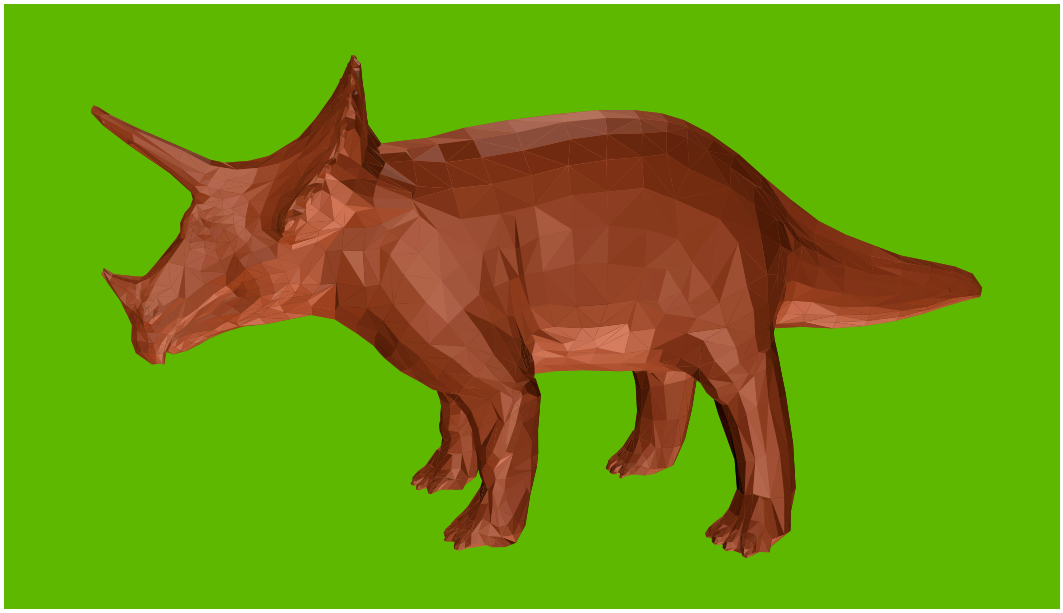


FIG. 5.19 – *Précision* = 9 *bits, Résidu de compression* = 12,4%

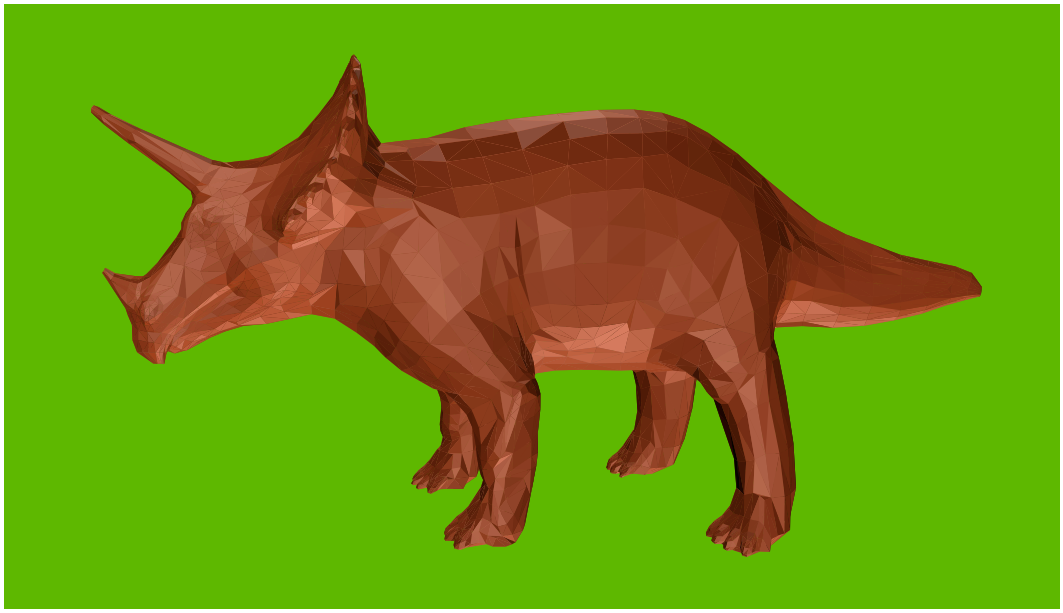


FIG. 5.20 – *Précision* = 10 *bits, Résidu de compression* = 14,6%

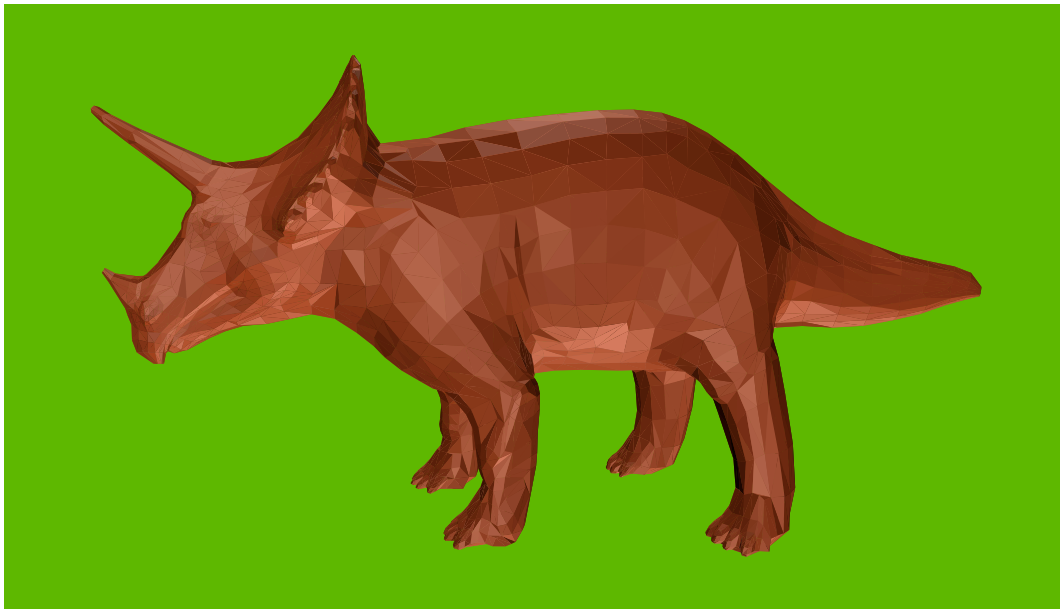


FIG. 5.21 – *Précision* = 11 *bits, Résidu de compression* = 16,8%

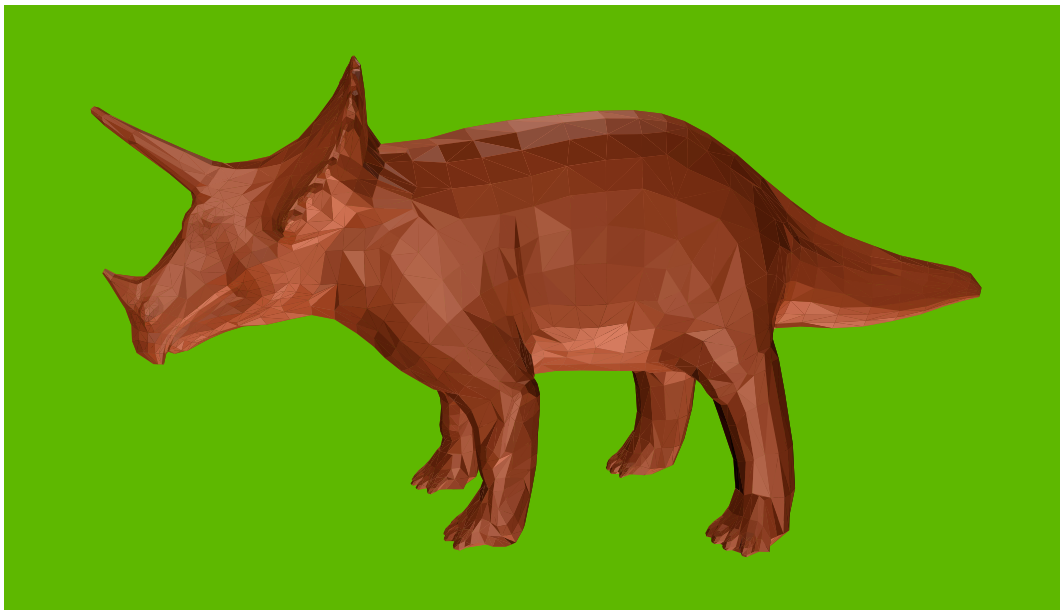


FIG. 5.22 – *Précision* = 24 *bits (compression sans perte), T. c.* = 44,2%

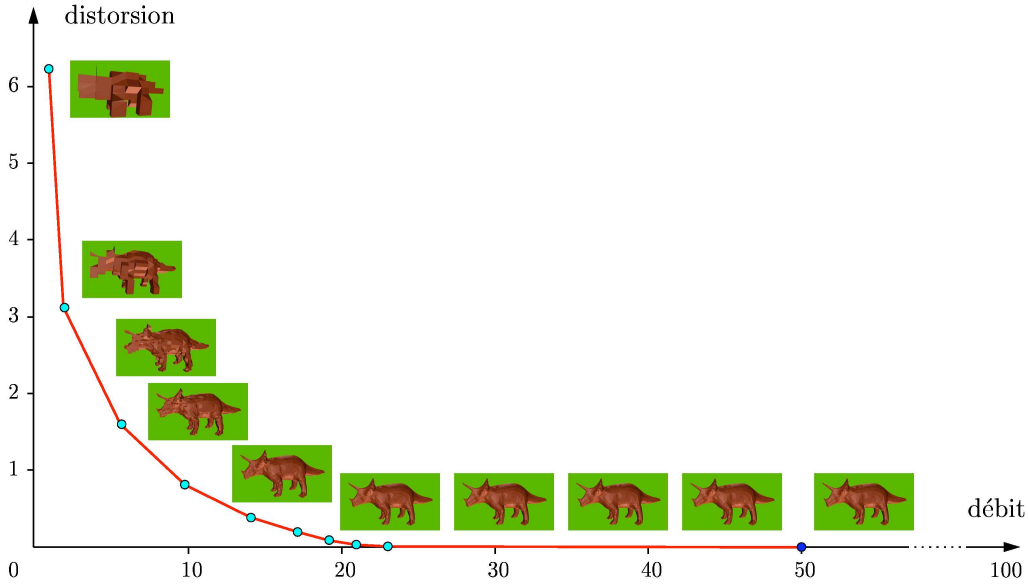


FIG. 5.23 – *Courbe débit/distortion*

Comme nous l'avons montré dans la section 5.2.2, l'opérateur de fusion de sommet rend notre algorithme applicable à une classe de modèles 3D beaucoup plus large que les surfaces triangulées *manifold*. Le tableau 5.4 regroupe les résultats de notre algorithme de compression sur des objets tridimensionnels modélisés par des soupes de triangles ; ces objets sont disponibles sur le site de 3DCafe (*http://www.3dcafe.com/asp/meshes.asp*). Comme pour les autres tableaux de résultats de ce mémoire, la première ligne donne le nombre de bits par sommet pour le codage de la topologie, et la seconde le nombre de bits par sommet pour le codage de la géométrie (pour une quantification de 12 bits sur chaque coordonnée). Contrairement aux modèles des tableaux 5.2 et 5.3, les opérations de contraction d'arête interviennent très rarement dans le processus de simplification : sur l'ensemble des objets testés, elles concernent moins de 5% des fusions de cellules. Cela a pour effet de porter le coût moyen du codage de la topologie à plus de 8 bits par sommet. A notre connaissance, il n'existe pas à ce jour de résultats comparables pour ce type de structure géométrique.

Les figures 5.24 à 5.33 montrent un exemple de décompression progressive d'une soupe de polygones (le modèle *m_tree1*). Pour chaque résolution, l'image de gauche correspond à une vue d'ensemble de l'objet, tandis que l'image de droite montre un zoom sur une sous-partie du modèle.

modèles	nombre de sommets	résultats
aqua05	16784	8,5 16,4
grass14_s	29224	7,5 18,8
maple01	45499	8,2 16,9
m_tree1	17782	8,7 16,0
willow01	47310	9,0 17,9
moyenne	156599	8,4 17,4

TAB. 5.4 – *Résultats sur des soupes de triangles*

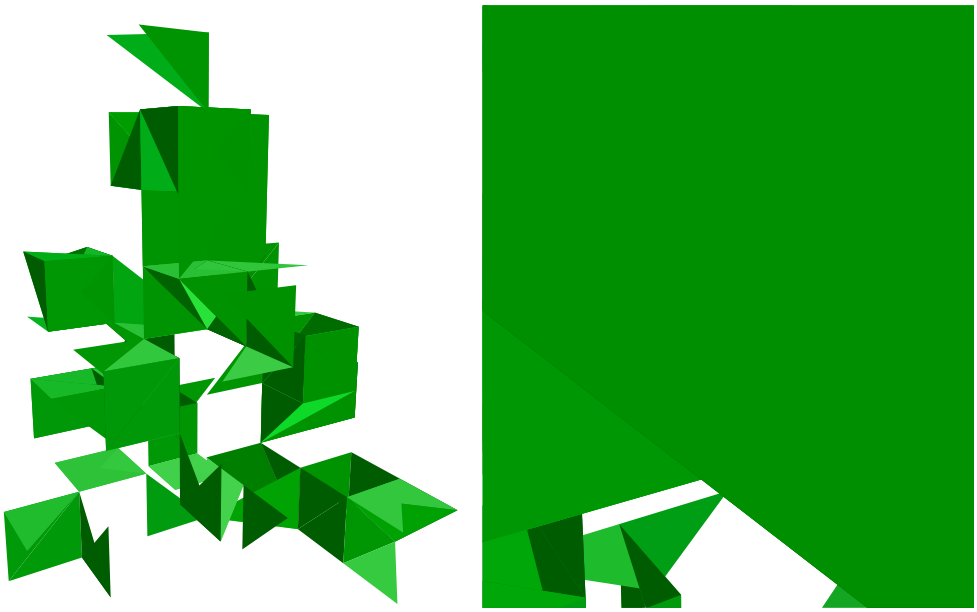


FIG. 5.24 – *Précision* = 3 *bits, Résidu de compression* = 0,3%

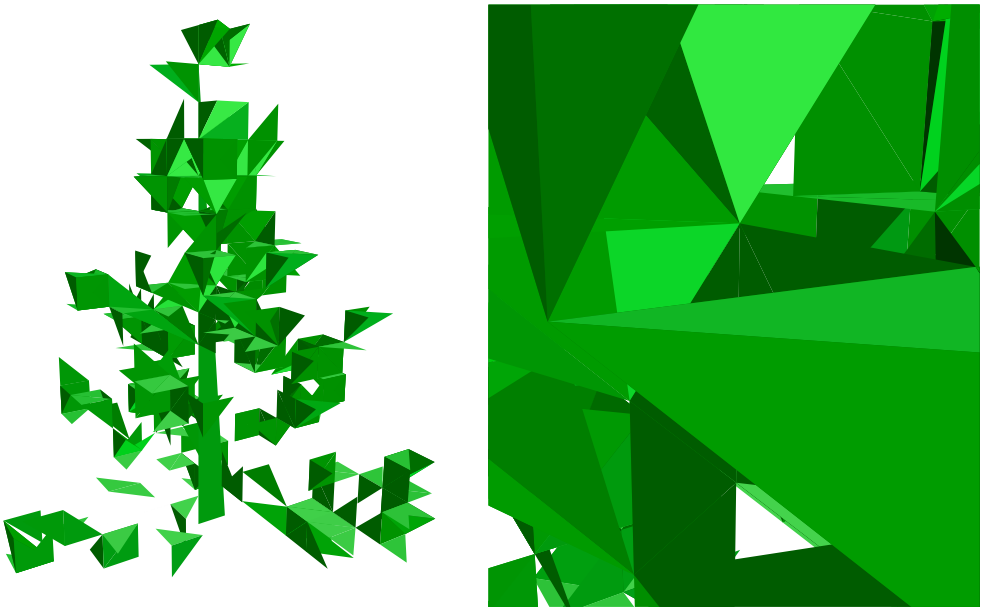


FIG. 5.25 – *Précision* = 4 *bits, Résidu de compression* = 0,9%

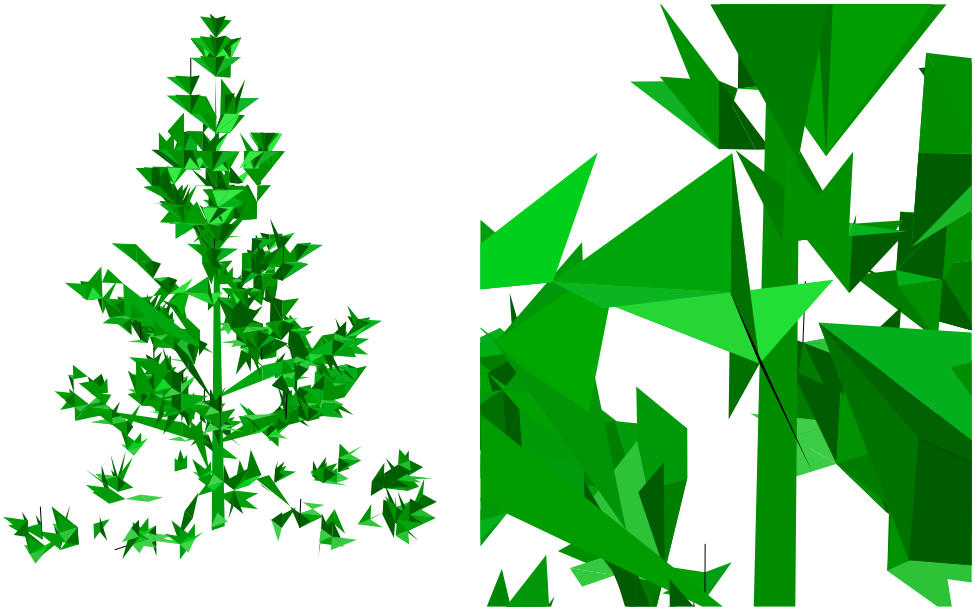


FIG. 5.26 – *Précision* = 5 *bits, Résidu de compression* = 2,2%

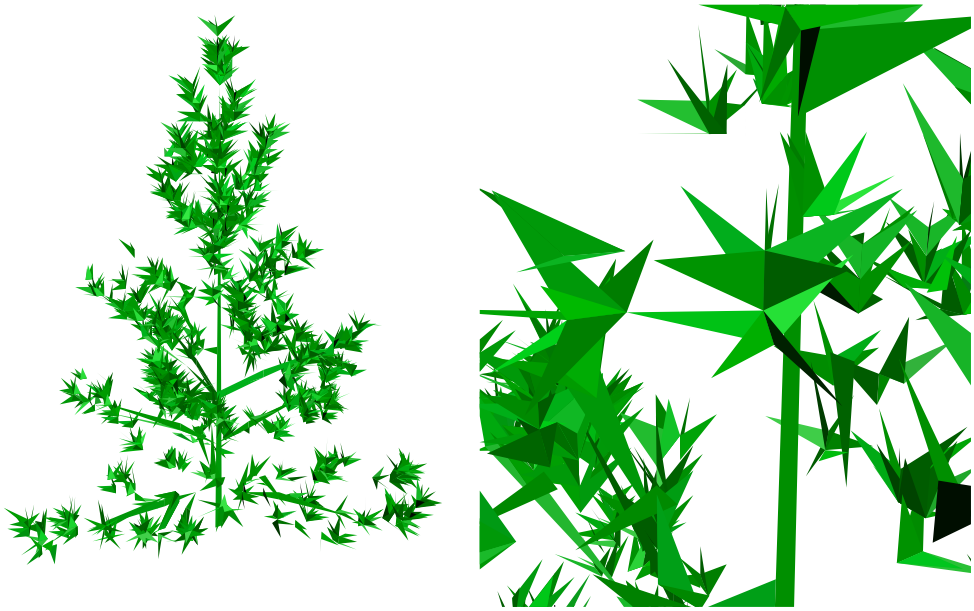


FIG. 5.27 – *Précision* = 6 *bits, Résidu de compression* = 3,8%

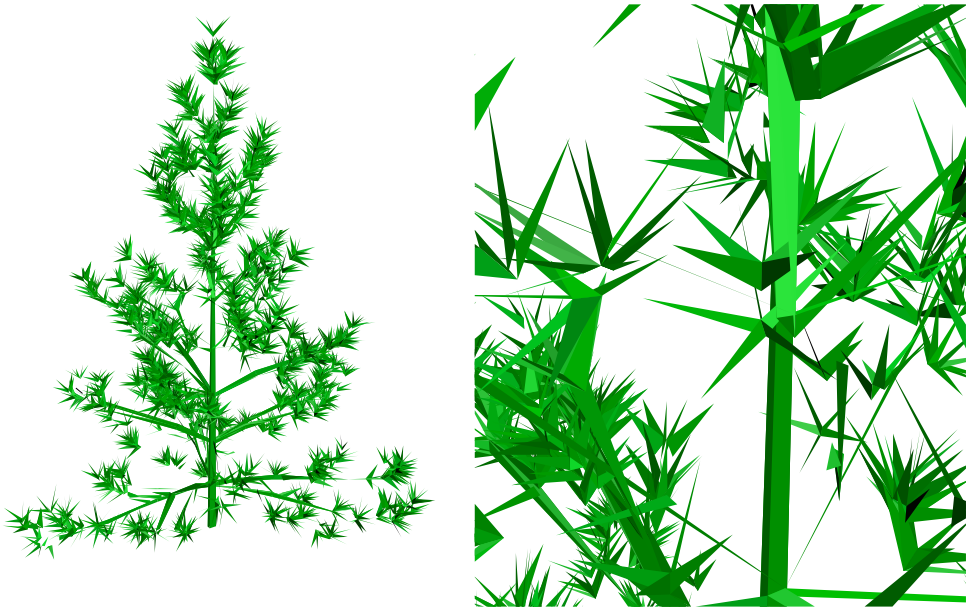


FIG. 5.28 – *Précision* = 7 *bits, Résidu de compression* = 5,6%

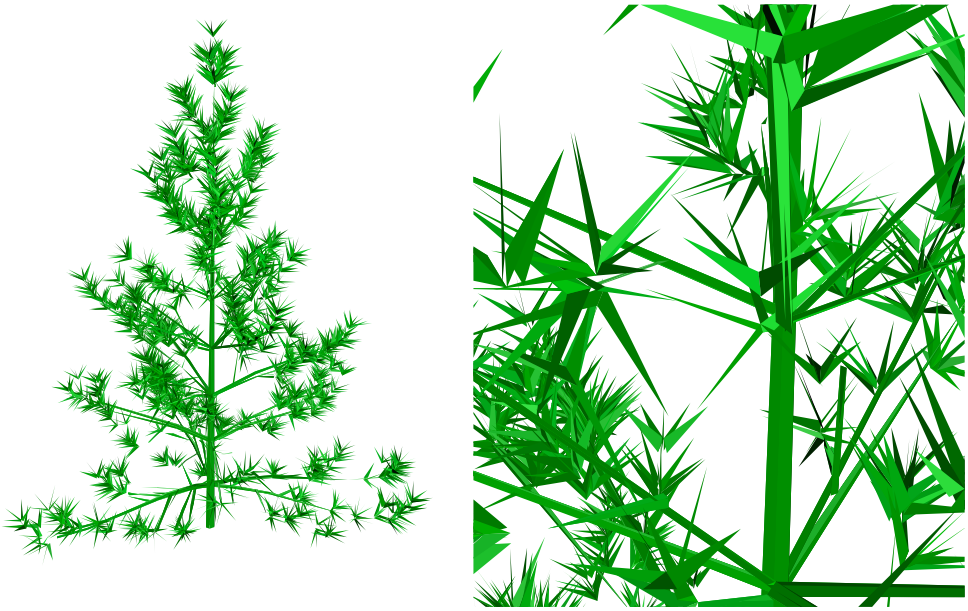


FIG. 5.29 – *Précision* = 8 *bits, Résidu de compression* = 7,6%

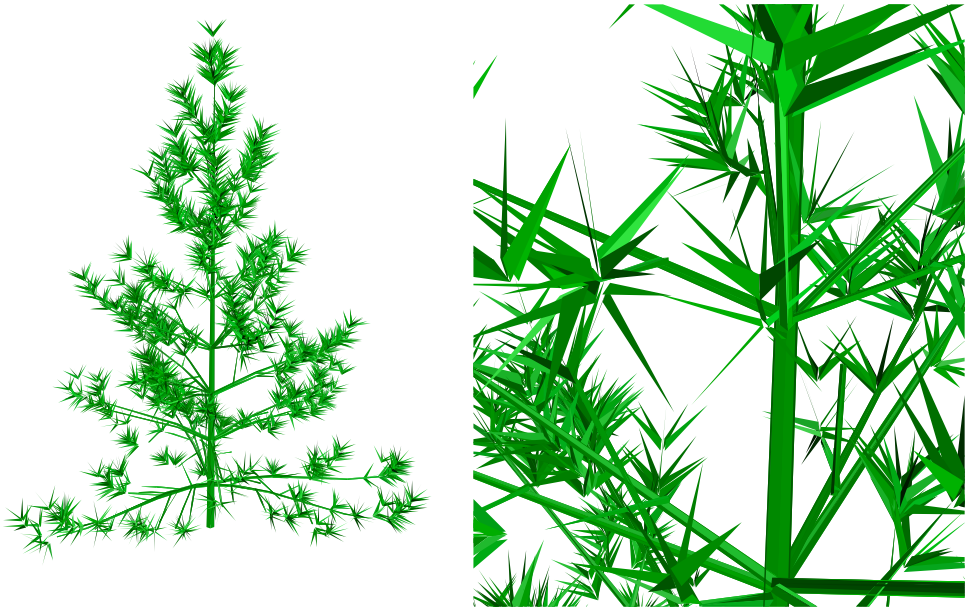


FIG. 5.30 – *Précision* = 9 *bits, Résidu de compression* = 9,9%

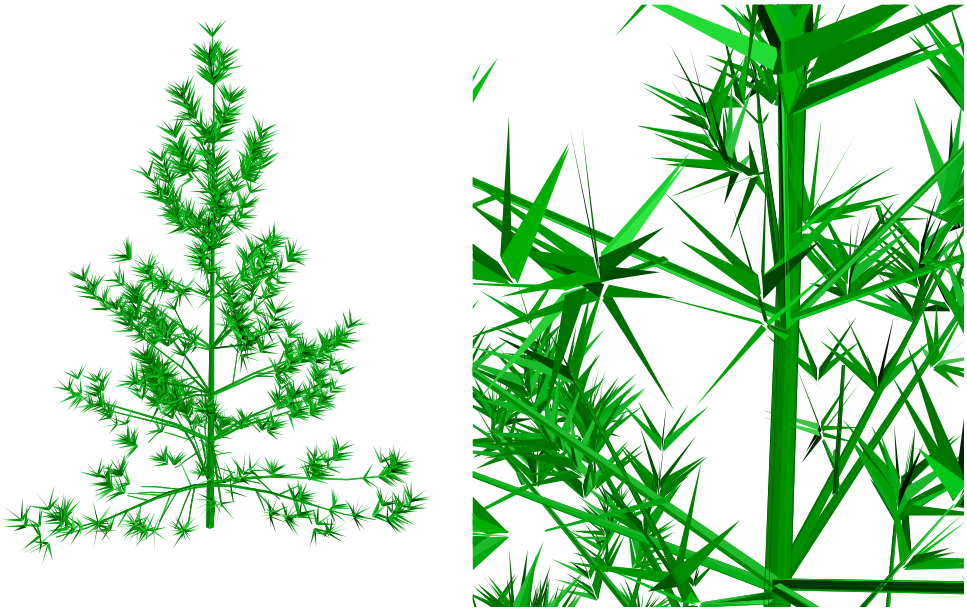


FIG. 5.31 – *Précision* = 10 *bits, Résidu de compression* = 12,2%

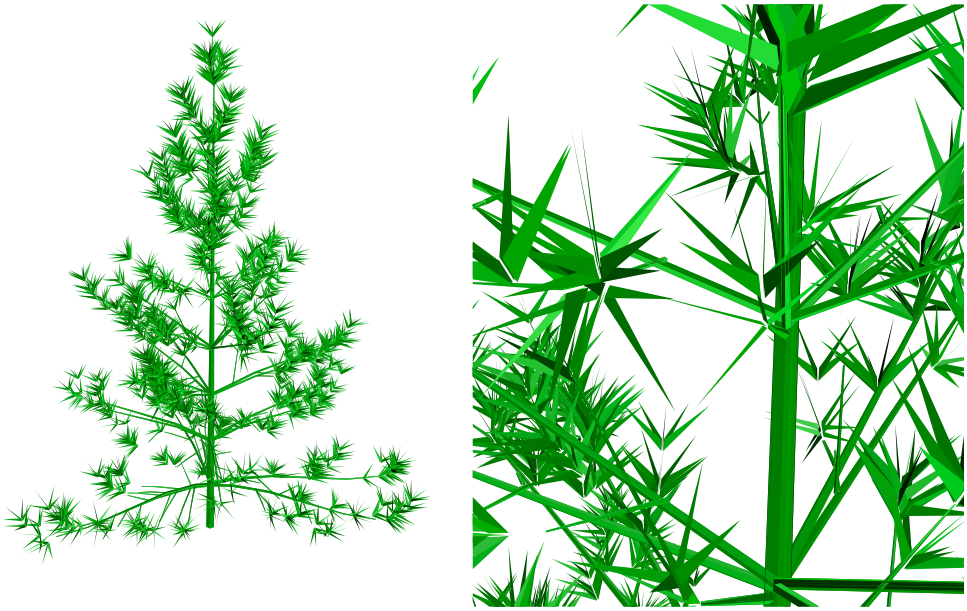


FIG. 5.32 – *Précision* = 11 *bits, Résidu de compression* = 14,2%

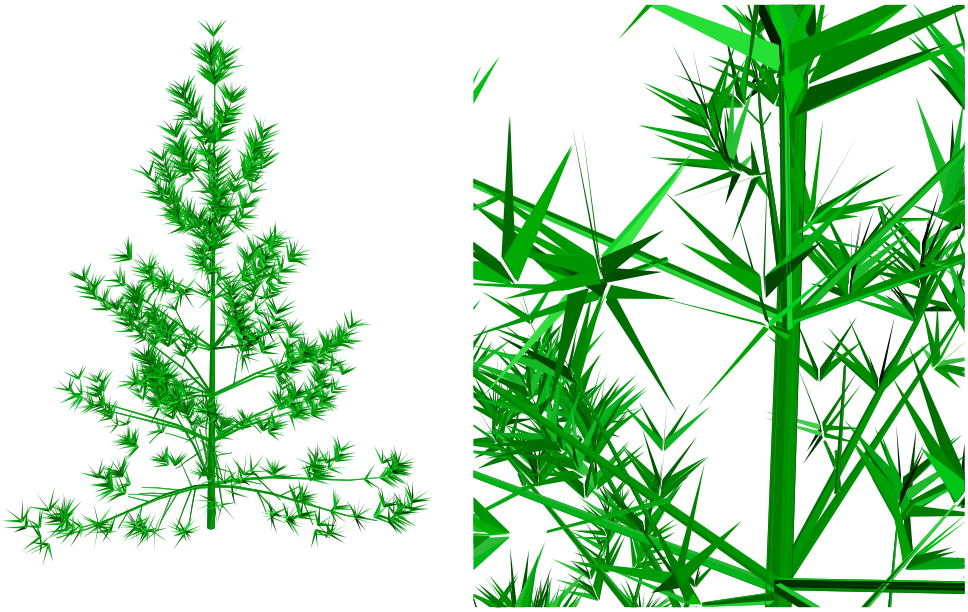


FIG. 5.33 – *Précision = 24 bits (compression sans perte), T. c. = 38,5%*

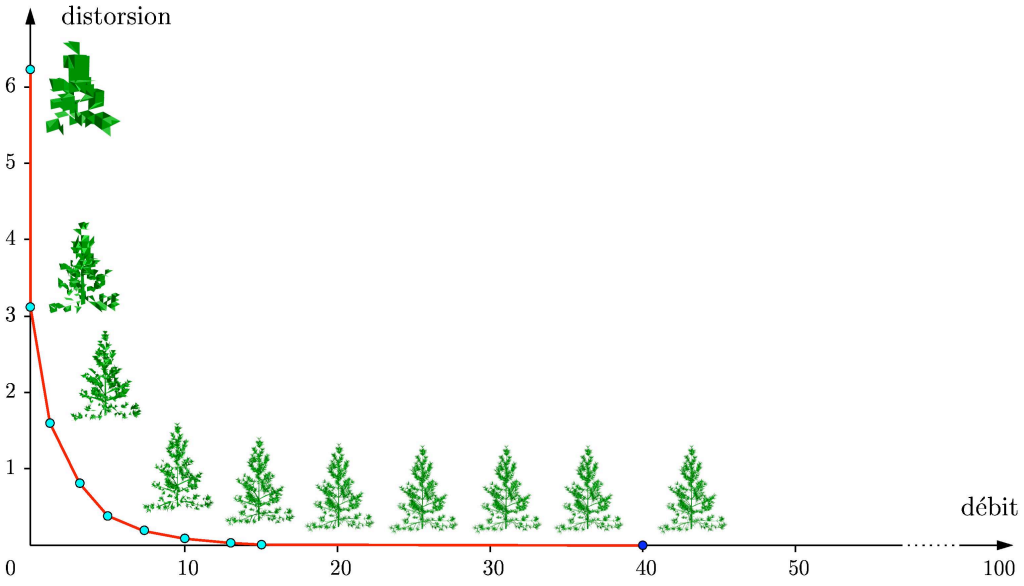


FIG. 5.34 – *Courbe débit/distortion*

Chapitre 6

Maillages tétraédriques

Les maillages tétraédriques sont utilisés dans les calculs et simulations par éléments finis sur des domaines volumiques depuis de nombreuses années ; ils sont également de plus en plus répandus en visualisation de volumes. De manière générale, de nombreux résultats d'analyse en science ou en ingénierie s'expriment sous forme d'échantillons de points répartis irrégulièrement dans l'espace tridimensionnel. A ces points sont associées des valeurs numériques et une information topologique qui permet d'interpoler ces valeurs en tout point de l'espace. Par leur simplicité et leur flexibilité, les maillages tétraédriques offrent une interpolation naturelle communément utilisée par la communauté scientifique.

Dans ce chapitre, après un état de l'art dans le domaine de la compression de maillages tétraédriques (section 6.1), nous proposons une généralisation au cas volumique de l'algorithme présenté au chapitre 5 pour la compression de maillages triangulaires surfaciques.

6.1 Précédents travaux

6.1.1 Compression monorésolution

Avec *Grow&Fold : Compression of Tetrahedral Meshes* [99, 100], Szymczak et Rossignac sont les premiers à proposer un algorithme de compression pour la connectivité des maillages tétraédriques. Jusqu'alors, des structures

de données avaient été proposées pour le codage efficace des tétraédrisations, mais leurs exigences du point de vue de la rapidité d'accès aux données étaient incompatibles avec une optimisation de l'encombrement en mémoire. Les auteurs proposent une méthode permettant de coder la connectivité du maillage en trois flots distincts : un arbre couvrant des tétraèdres du maillage, une séquence de pliages, et une séquence de collages. Comme dans de nombreuses méthodes de compression géométrique, l'arbre des tétraèdres est construit par un parcours en profondeur : à partir d'une face externe du maillage, l'algorithme réalise une conquête récursive sur les sommets. Le premier flot est donc composé des sommets réordonnés. Chaque sommet, correspondant à l'exploration d'un nouveau tétraèdre, est suivi de 3 bits indiquant quelles sont les faces "attachables" de ce tétraèdre (c'est-à-dire les faces qui donneront lieu à la conquête d'un nouveau tétraèdre). Dans cet arbre couvrant, les tétraèdres sont "éclatés", et les faces internes du maillage apparaissent donc en double. Le second flot, composé de 2 bits par tétraèdre, indique au décodeur les arêtes le long desquelles l'arbre doit être replié pour superposer les deux faces correspondantes. Enfin, un troisième flot permet de gérer le recollement de faces externes dans l'arbre des tétraèdres mais qui ne sont pas sur le bord du maillage original (dues notamment à la présence de trous et de poignées). Cette opération, plus générale mais aussi plus coûteuse, est évitée autant que possible par une heuristique gloutonne privilégiant le pliage au collage. Au total, un résidu de compression d'environ 47 bits par sommet est atteint sur des tétraédrisations de Delaunay aléatoires. La méthode ne traite pas le codage de la géométrie.

Dans *Tetrahedral Mesh Compression with the Cut-Border Machine* [44], Gumhold, Guthe et Strasser proposent d'étendre le codeur de surfaces triangulées de Gumhold et Strasser [45] au cas des maillages tétraédriques. L'algorithme procède par conquête sur les tétraèdres, en faisant croître une région contenant les tétraèdres déjà codés. La frontière de cette région (le *cut-border*) est une surface triangulée non *manifold* dont les faces sont stockées dans une file d'attente. A chaque étape, une face est retirée de la file et devient le portail courant par lequel un nouveau tétraèdre est incorporé à la région. On distingue 10 situations donnant lieu à des codes différents ; elles correspondent à la position du quatrième sommet, des 3 faces et 3 arêtes du nouveau tétraèdre par rapport au *cut-border*. Lorsque le quatrième sommet n'appartient pas au *cut-border* (dans moins de 20% des cas), le code *new vertex* est transmis, suivi des coordonnées du nouveau sommet. Lorsque le quatrième sommet du tétraèdre appartient à la frontière, un code *connect* est transmis, suivi de l'offset de ce sommet dans le *cut-border*. Ce cas, le plus fréquent en pratique, se subdivise en plusieurs sous-cas selon que les faces

et les arêtes du nouveau tétraèdre appartiennent ou pas au *cut-border*, et peut en particulier engendrer des arêtes ou des sommets non *manifold* sur la frontière (cf figure 6.1). Enfin, lorsque le portail ne permet pas d'incorporer un nouveau tétraèdre à la région de conquête (i.e. si le portail appartient au bord du maillage), un code *border* est transmis. Une heuristique est proposée sur la numérotation des sommets autour du portail, qui permet de maximiser le nombre de connexion avec un sommet de petit numéro. Ainsi, l'utilisation d'un codeur arithmétique permet de tirer parti de la disparité des fréquences des différentes commandes, pour parvenir à un coût de seulement 15 bits par sommet pour la description de la topologie. La compression de la géométrie est assurée par un codage différentiel : c'est le vecteur différence entre le centre du portail et le nouveau sommet qui est transmis. Cette technique conduit à un coût allant de 30 à 36 bits par sommet pour une quantification de 16 bits par coordonnée. La figure 6.1 montre 3 des 10 configurations possibles : les faces du *cut-border* sont en orange, les bords du portail courant en rouge, les autres arêtes du nouveau tétraèdre en vert et ses faces en violet.

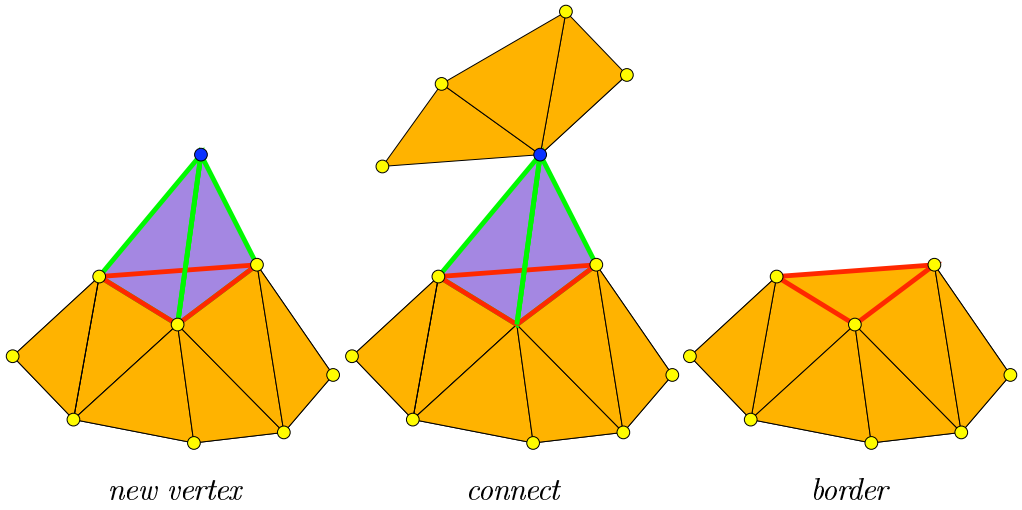


FIG. 6.1 – *La méthode du* cut-border

6.1.2 Compression progressive

Implant Sprays : Compression of Progressive Tetrahedral Mesh Connectivity, de Pajarola, Rossignac et Szymczak [84], décrit la seule méthode de compression progressive de maillages tétraédriques publiée jusqu'à présent. L'algorithme proposé repose sur une simplification du maillage par *vagues*

successives, grâce à une primitive de contraction d'arête adaptée aux tétraèdres. Cette opération a pour effet d'éliminer tous les tétraèdres adjacents à l'arête contractée: chacun de ces tétraèdres est aplati, donnant naissance à une *face-germe* (ou *cut-face*), qui sera dilatée lors de l'opération inverse (la subdivision de sommet) pour engendrer le tétraèdre original. Ainsi, une subdivision de sommet, qui permet de passer progressivement du maillage grossier au maillage original par raffinement, est complètement déterminée par le sommet à subdiviser et sa *jupe*, c'est-à-dire l'ensemble de ses faces-germes. Si l'on définit la surface orbitale O d'un sommet s par la surface triangulée formée de l'ensemble des faces des tétraèdres incidents à s privé des faces incidentes à s, la jupe J de s décrit un cycle d'arêtes sur O (cf figure 6.2). Pour chaque sommet à subdiviser, les auteurs proposent de coder une jupe de taille k comme un chemin d'arêtes sur O, soit un point de départ sur $\log_2 d$ bits (où d est le degré du sommet), et $k-1$ voisins successifs sur $\log_2 6$ bits (6 étant le degré moyen dans une triangulation planaire). En ce qui concerne le codage des sommets à subdiviser eux-mêmes, le coût naïf de $\log_2 n$ (où n est le nombre de sommets dans le maillage courant) est amélioré en procédant par vagues de subdivisions simultanées (les *implant sprays*). A chaque étape du raffinement, 1 bit est attribué à chaque sommet de la triangulation courante pour indiquer si le sommet doit être subdivisé ou pas. Le coût total de cette technique dépend du nombre d'arêtes indépendantes qui peuvent être contractées simultanément au cours du processus de simplification. En outre, des contraintes supplémentaires interdisent la contraction de certaines arêtes, en particulier pour éviter la création de zones non *manifold* dans la triangulation simplifiée. En pratique, à chaque étape de simplification, environ 1/15 du nombre de sommets total est supprimé. Finalement, en comptant le coût du codage du maillage de base, des résidus voisins de 45 bits par sommet sont obtenus pour la compression progressive de la connectivité de maillages tétraédriques aléatoires. La méthode n'aborde pas le problème du codage des positions.

6.2 Description de l'algorithme

6.2.1 La fusion de sommets

La généralisation de l'opérateur de fusion de sommets décrit à la section 5.2.2 est directe et possible en dimension quelconque. De ce fait, l'algorithme que nous avons proposé dans le cadre des maillages triangulaires surfaciques

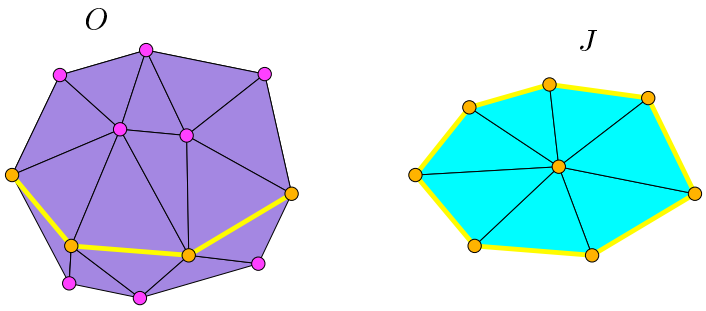


FIG. 6.2 – *La surface orbitale et le cycle d'arête formé par les faces-germes*

est facilement applicable à tout complexe simplicial en dimension d. Cependant, en l'absence d'un opérateur de coût réduit — l'équivalent en dimension 3 de la contraction d'arête décrite à la section 5.2.2 — les performances en terme de résidus 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 tétraédriques 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 moyenne si l'on utilise en outre un codeur entropique.

6.2.2 La contraction d'arête

Cet opérateur est plus délicat à généraliser ; nous suggérons ici une primitive de simplification équivalente adaptée aux maillages tétraédriques. Cette primitive est similaire à la contraction d'arête utilisée par Pajarola, Rossignac et Szymczak dans leur méthode des *implant sprays* [84]. Le principe général est de traiter séparément les fusions de sommets incidents qui ne créent pas de changement topologique dans leur voisinage. Plus formellement, il s'agit de contractions d'arêtes dont l'opération inverse remplit les conditions suivantes (cf figure 6.3) :

- le sommet à subdiviser est de code 4 (expansion d'arête),
- l'ensemble des faces de code 4 (i.e. les faces donnant naissance à un tétraèdre lors de la subdivision) forme une surface S *manifold*,
- les arêtes restantes (i.e. 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.

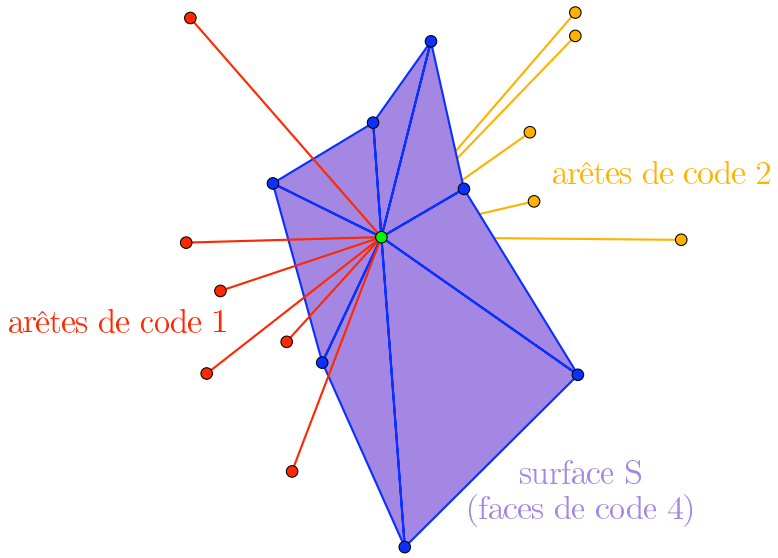


FIG. 6.3 – *Cas favorable à l'expansion d'arête 3D*

Lorsque toutes ces conditions sont réunies, la séquence 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.

6.3 Résultats expérimentaux

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 tétraédrisation 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, Rossignac et Szymczak [84], les seuls à notre connaissance à proposer un codeur topologique progressif pour les maillages tétraédriques. Pour une tétraédrisation de Delaunay aléatoire de 10000 sommets, le résidu de compression obtenu par leur algorithme est de l'ordre de 45 bits par sommet.

Chapitre 7

Conclusion et perspectives

L'objectif initial de cette thèse était d'aborder le problème de la compression de structures géométriques du point de vue original de la *géométrie algorithmique*. Plus précisément, il s'agissait de chercher comment un problème tel que la reconstruction de maillages à partir de nuages de points, étudié depuis longtemps par la communauté de géométrie algorithmique, pouvait fournir une aide précieuse pour la compression géométrique. Ainsi, en admettant que dans de nombreux cas, la topologie d'une structure géométrique pouvait être reconstruite automatiquement à partir de sa géométrie, nous avons orienté nos recherches dans le sens du codage efficace d'un ensemble de points de l'espace. En faisant ce choix, nous partions du principe que l'essentiel de l'information contenue dans un objet géométrique était porté par la position des sommets qui la composaient, et donnions la priorité au codage efficace de cette information. Cette démarche situe le présent travail à contre-courant des nombreuses recherches menées depuis plus de cinq ans dans le domaine de la compression géométrique. En effet, la plupart des méthodes existantes décrivent un parcours des sommets du maillage permettant de représenter sa connectivité à moindre coût, puis proposent des techniques de prédiction pour le codage de la géométrie dont l'efficacité dépend dans une large mesure du parcours fixé par le codeur topologique.

La première contribution de ce mémoire consiste donc en un algorithme de compression *progressif et sans perte* d'un ensemble de points non structuré en dimension quelconque. Une borne inférieure est montrée, prouvant que le gain engendré par la méthode est supérieur à $\log_2 n$ bits par point, où n est le nombre total de points.

Un outil de reconstruction bien connu en géométrie algorithmique est donné par la triangulation de Delaunay. Lorsque la topologie d'un ensemble de points coplanaires est obtenue par cette méthode, l'algorithme proposé peut être appliqué directement. Lorsque certaines faces d'un modèle ne peuvent pas être obtenues automatiquement, leur codage explicite devient nécessaire. Très souvent, ce sous-ensemble de la topologie de l'objet est de petite taille, et la *triangulation de Delaunay contrainte*, qui construit une triangulation de Delaunay sur un ensemble d'arêtes fixées, permet de retrouver la topologie du modèle. Pour traiter ces cas, nous avons proposé, après une étude théorique sur la taille minimale de l'ensemble d'arêtes contraintes, un algorithme permettant de coder cet ensemble en même temps que la géométrie du modèle.

Lorsque la structure géométrique est une surface plongée dans l'espace tridimensionnel, la reconstruction de sa topologie sans perte d'information ne peut pas toujours être garantie. Des solutions pour la compression *progressive et sans perte* ont donc été proposées dans ce mémoire pour les surfaces constituées de *polygones quelconques*. Pour le cas particulier très répandu des maillages surfaciques triangulaires, nous avons décrit un second codeur de connectivité très général puisqu'il permet de comprimer la topologie de tout complexe simplicial (en particulier les triangulations non *manifold*). La généralisation et l'adaptation de ce codeur topologique aux complexes simpliciaux volumiques fournit également un outil efficace pour la compression de maillages tétraédriques.

Outre les résultats obtenus par les différents algorithmes décrits dans ce mémoire, qui se positionnent avantageusement par rapport aux méthodes progressives actuelles les plus efficaces, un point fort de ce travail concerne la *généricité* des codeurs géométrique et topologiques proposés. En effet, la classe des structures géométriques concernées s'étend de l'ensemble de points non structuré (en dimension quelconque), aux maillages volumiques, en passant par les maillages surfaciques triangulaires non nécessairement *manifold* et les maillages polygonaux quelconques. Pour tous ces types d'objet géométrique, il est ainsi possible d'obtenir un codage efficace *sans perte d'information* (à la différence de nombreuses méthodes, nous n'imposons pas de quantification préalable avec perte de précision sur les coordonnées des points comprimés), et de manière *progressive*. Cette échelonnabilité, primordiale pour nombre d'applications réseau où la qualité de la visualisation des modèles prévaut, autorise une véritable interactivité avec l'utilisateur final : outre les désormais classiques niveaux de détails (*level of details* ou *LOD* en anglais) utilisés pour la navigation dépendant du point de vue, la hiérar-

chisation des données en *quadtree* permet d'optimiser également le volume des informations transmises lors des mouvements de *zooms* sur différentes sous-parties du modèle.

Les perspectives de recherche envisageables au terme de ce travail peuvent être groupées autour d'un pôle applicatif et d'un pôle plus théorique. Pour ce qui est des applications directes des algorithmes proposés dans ce mémoire, nous pensons que l'implantation actuelle peut être encore optimisée et enrichie. Si l'optimisation en temps n'est pas essentielle dans la mesure où la compression des modèles, qui constitue le travail le plus coûteux, est réalisée hors-ligne, l'optimisation de l'encombrement mémoire, en revanche, nous paraît importante. En effet, pour pouvoir manipuler des modèles géométriques de plusieurs centaines de milliers de points sur une machine standard sans partitionnement préalable, il est nécessaire de réduire l'espace occupé par les structures de données utilisées (par exemple en remplaçant les listes de l'implantation actuelle par des tableaux dynamiques). D'autre part, de nombreuses caractéristiques offertes par les méthodes de ce mémoire pourraient être mieux exploitées par une implantation réellement interactive comprenant les zooms et la navigation en temps réel dépendant du point de vue. La gestion de *geomorphs*, c'est-à-dire de transitions lisses entre les différentes résolutions apporterait également une amélioration sensible du point de vue du confort visuel.

Par ailleurs, il existe sans doute des domaines d'application pour lesquels les algorithmes de compression géométrique pourraient être utilisés avec profit. Par exemple, les développements récents des cartes routières et urbaines diffusées par satellite ou par ondes radio imposent la transmission de graphes complexes et volumineux sur une bande passante limitée. Dans le domaine de la recherche en biologie ou en pharmacologie, on peut également avoir besoin de comprimer de gros ensembles de molécules, représentées par des points dans un espace à 4 dimensions (3 dimensions pour la position plus 1 pour le rayon de la sphère).

En ce qui concerne les pistes de recherche plus théoriques, nous pensons qu'une collaboration étroite avec la communauté de géométrie algorithmique travaillant sur les problèmes de reconstruction pourrait s'avérer très fructueuse. En effet, dans ce mémoire, la reconstruction a été exploitée essentiellement dans le cadre des triangulations bidimensionnelles, et le cas des maillages surfaciques plongés en 3 dimensions reste à traiter. Or, certaines méthodes actuelles de reconstruction de surfaces tridimensionnelles [7, 15, 14] donnent une condition nécessaire et suffisante sur l'échantillonnage de l'objet

pour garantir une reconstruction valide. Il devrait donc être possible, en ajoutant éventuellement un petit nombre de points au modèle original, de garantir une reconstruction sans perte de sa topologie par l'une de ces méthodes. Une autre manière d'exploiter ce type d'algorithmes serait d'étendre le principe de la triangulation de Delaunay contrainte aux surfaces triangulées en 3 dimensions. L'idée générale est de tirer parti de la connaissance a priori de la connectivité de l'objet à comprimer, pour coder uniquement l'information topologique minimale permettant de le reconstruire sans perte. Cela revient à assister l'algorithme de reconstruction en lui spécifiant certaines faces ou arêtes qu'il ne trouverait pas seul.

Une autre perspective importante est d'améliorer les techniques de prédiction, dont les résultats actuels ne sont pas complètement satisfaisants. Cela pourrait se faire par une analyse statistique plus précise des distributions de points, et en appliquant des méthodes d'optimisation pour le réglage des paramètres.

Enfin, une collaboration avec la recherche dans le domaine des réseaux permettrait d'aborder les problèmes purement liés à la transmission, en vue d'améliorer la robustesse de nos algorithmes par rapport à la perte de paquets.

Bibliographie

[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 M. Desbrun. Progressive compression for lossless transmission of triangle meshes. In *SIGGRAPH 2001 Conference Proc.*, 2001.

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

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

[7] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. In *ACM SoCG 98 Conference Proc.*, pages 39–48, 1998.

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

[9] C. Bajaj, V. Pascucci, and G. Zhuang. Compression and coding of large cad models. Technical report, University of Texas, 1998.

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

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

[12] R. Bar-Yehuda and C. Gotsman. Time/space tradeoffs for polygon mesh rendering. *ACM Transactions on Graphics*, 15(2), 1996.

[13] M. Bern and D. Epstein. Quadrilateral meshing by circle packing. In *6th International Meshing Roundtable*, 1997.

[14] J.-D. Boissonnat and F. Cazals. Smooth shape reconstruction. In *ACM SoCG 2000 Proc.*, 2000.

[15] J.-D. Boissonnat and B. Geiger. 3-dimensional reconstruction of complex shapes based on the Delaunay triangulation. In *Biomedical Image Processing and Biomedical Visualization*, volume 1905, pages 964–975, 1993.

[16] J-D. Boissonnat and M. Yvinec. *Géométrie algorithmique*. Ediscience international, Paris, 1995.

[17] J.-D. Boissonnat and M. Yvinec. *Algorithmic geometry*. Cambridge University Press, UK, 1998. traduit de la version française (Ediscience international) par Hervé Brönnimann.

[18] R. Carey, G. Bell, and C. Martin. *The Virtual Reality Modeling Language ISO/IEC DIS 14772-1*. VRML97/DIS, http://www.vrml.org/Specifications, 1997.

[19] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

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

[21] L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4:97–108, 1989.

[22] M. M. Chow. Optimized geometry compression for real time rendering. In *IEEE Visualization 97 Conference Proc.*, pages 347–354, 1997.

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

[24] G. Davis and A. Nosratinia. Wavelet-based image coding : an overwiew. *Applied Computational Control, Signals, and Circuits*, 1998.

[25] M. de Berg and K. Dobrindt. On level of details in terrains. *Graph. Models Image Process.*, 1998.

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

[27] M. Deering. Hardware geometry compression : Specification from java 3d. Technical report, Sun Microsystems, 1998.

[28] M. O. Denny and C. A. Sohler. Encoding a triangulation as a permutation of its point set. In *Canadian Conference on Computational Geometry 1997 Proc.*, August 1997.

[29] M. Desbrun, M. Meyer, P. Schroeder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 99 Conference Proc.*, 1999.

[30] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[31] R. A. DeVore, B. Jawerth, and B. J. Lucier. Surface compression. *Computer Aided Geometric Design*, 1992.

[32] J. Dorsey, A. Edelman, J. Legakis, H. W. Jensen, and H. K. Pedersen. Modeling and rendering weathered stone. In *SIGGRAPH 99 Conference Proc.*, 1999.

[33] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proc.*, 1995.

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

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

[36] L. De Floriani, P. Magillo, and E. Puppo. A simple and efficient sequential encoding for triangle meshes. In *15th European Workshop on Computational Geometry*, 1999.

[37] L. De Floriani, P. Marzano, and E. Puppo. Hierarchical terrain models : Survey and formalization. In *ACM Symposium on Applied Computating Proc.*, 1994.

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

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

[40] 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.

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

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

[43] A. Guéziec, F. Bossen, G. Taubin, and C. Silva. Efficient compression of non-manifold polygonal meshes. In *IEEE Visualization 99 Conference Proc.*, 1999.

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

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

[46] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *SIGGRAPH 2000 Conference Proc.*, 2000.

[47] J. Hartman and J. Wernecke. *The VRML 2.0 Handbook.* Addison-Wesley, 1996.

[48] P. S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, Carnegie Mellon University, 1997.

[49] P. S. Heckbert, J. Rossignac, H. Hoppe, W. Schröder, M. Soucy, and A. Varshney. Multiresolution surface modeling. Technical report, SIGGRAPH 97 Course Notes, 1997.

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

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

[52] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proc. Inst. Electr. Radio Eng.*, September 1952.

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

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

[55] M. Isenburg and J. Snoeyink. Face fixer : Compressing polygon meshes with properties. In *SIGGRAPH 2000 Conference Proc.*, 2000.

[56] M. Isenburg and J. Snoeyink. Spirale reversi : Reverse decoding of the edgebreaker encoding. In *12th Canadian Conference Computational Geometry Proc.*, 2000.

[57] A. Itai and M. Rodeh. Representations of graphs. *Acta Informatica*, 1982.

[58] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH 2000 Conference Proc.*, 2000.

[59] K. Keeler and J. Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58, 1995.

[60] A. Khodakovsky and I. Guskov. Normal mesh compression. In *to appear*, 2000.

[61] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *SIGGRAPH 2000 Conference Proc.*, 2000.

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

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

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

[65] D. King, A. Szymczak, and J. Rossignac. Connectivity compression for irregular quadrilateral meshes. Technical report, Georgia Institute of Technology, 1999.

[66] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computating*, 1983.

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

[68] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 1999.

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

[70] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parametrization of surfaces. In *SIGGRAPH 98 Conference Proc.*, 1998.

[71] D. T. Lee and A. K. Lin. Generalized Delaunay triangulation for planar graphs. *Discrete Comput. Geom.*, 1:201–217, 1986.

[72] 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.

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

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

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

[76] D. Luebke. A survey of polygonal simplification algorithms. Technical report, University of North Carolina at Chapel Hill, 1997.

[77] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH 97 Conference Proc.*, 1997.

[78] M. Müller-Hannemann. *Quadrilateral Mesh Generation in Computer-Aided Design*. PhD thesis, Technische universität Berlin, 1997.

[79] M. Naor. Succinct representation of general unlabeled graphs. *Discrete Applied Mathematics*, 29, 1990.

[80] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide*. Addison-Wesley, 1993.

[81] Y. Ohtake, A. Belyaev, and I. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. In *Proceedings of the Geometric Modeling and Processing 2000*, 2000.

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

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

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

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

[86] F. Preparata and I. Shamos. *Computational Geometry: An Introduction*, pages 72–77. Springer-Verlag, 1985.

[87] J. Rissanen. A universal data compression system. *IEEE Transactions on Information Theory*, 29, 1983.

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

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

[90] J. Rossignac and P. Borrel. *Geometric Modeling in Computer Graphics*, chapter Multi-Resolution 3D Approximations for Rendering Complex Scenes, pages 455–465. Springer-Verlag, July 1993.

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

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

[93] M. Segal and K. Akeley. The opengl graphics system. Technical report, silicon Graphics, 1998.

[94] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[95] J. Snoeyink and M. Van Kreveld. Linear time reconstruction of Delaunay triangulations with applications. In *European Symposium on Algorithms Proc.*, 1997.

[96] C. Sohler. Fast reconstruction of Delaunay triangulations. In *11th Canadian Conference Computational Geometry Proc.*, 1999.

[97] O. G. Staadt, M. H. Groos, and R. Weber. Multiresolution compression and reconstruction. In *Eurographics 98 Conference Proc.*, 1998.

[98] A. Szymczak, D. King, and J. Rossignac. An edgebreaker-based efficient compression scheme for regular meshes. In *12th Canadian Conference Computational Geometry Proc.*, 2000.

[99] A. Szymczak and J. Rossignac. Grow and fold : Compression of tetrahedral meshes. In *ACM Symposium on Solid Modeling Proc.*, 1999.

[100] A. Szymczak and J. Rossignac. Grow and fold : Compressing the connectivity of tetrahedral meshes. In *Computer-Aided Design Proc.*, 2000.

[101] G. Taubin. A signal approach to fair surface design. In *SIGGRAPH 95 Conference Proc.*, 1995.

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

[103] G. Taubin, W. Horn, F. Lazarus, and J. Rossignac. Geometry coding and vrml. *Proceedings of the IEEE*, 86(6), 1998.

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

[105] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH 87 Conference Proc.*, 1987.

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

[107] G. Turan. Succinct representations of graphs. *Discrete Applied Mathematics*, 8, 1984.

[108] W. Tutte. A census of planar triangulation. *Canadian Journal of Mathematics*, 14, 1962.

[109] W. Tutte. *The Enumerative Theory of Planar Graphs*. J. N. Srinivasan et al., 1973.

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

[111] X. Xiang, M. Held, and J. D. B. Mitchell. fast and effective stripification of polygonal surface models. In *ACM Symposium on Interactive 3D Graphics Proc.*, 1999.

[112] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[113] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

[114] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution surface viewing. In *SIGGRAPH 97 Conference Proc.*, 1997.

Publications de l'auteur

[115] O. Devillers and P.-M. Gandoin. Compression géométrique pour une transmission progressive. Research Report 3766, INRIA, Septembre 1999.

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

[117] O. Devillers and P.-M. Gandoin. Geometric compression for interactive transmission. Research Report 3910, INRIA, Mars 2000.

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

[119] O. Devillers, R. Estkowski, P.-M. Gandoin, F. Hurtado, P. Ramos, and V. Sacristán. Minimal set of constraints for 2D constrained Delaunay reconstruction. Research Report 4119, INRIA, Février 2001.

[120] O. Devillers and P.-M. Gandoin. Compression interactive de maillages triangulaires arbitraires. Research Report 4158, INRIA, Avril 2001.