Compression progressive de maillages surfaciques texturés

Florian Caillaud

N° d'ordre NNT : 2017LYSEI004

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'INSA Lyon

École Doctorale EDA512
Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le 17/01/2017, par :
Florian CAILLAUD

Compression progressive de maillages surfaciques texturés

Devant le jury composé de :

ALLIEZ Pierre, Directeur de recherche, INRIA Sophia-Antipolis	Rapporteur
BONNEAU Georges-Pierre, Professeur des universités, Université Grenoble Alpes	Rapporteur
BECHMANN Dominique, Professeure des universités, Université de Strasbourg	Examinatrice
HUDELOT Céline, Maître de conférence, CentraleSupélec	Examinatrice
LAVOUÉ Guillaume, Maître de conférence HDR, INSA Lyon	Directeur de thèse
VIDAL Vincent, Maître de conférence, Université Claude Bernard Lyon	Co-encadrant
DUPONT Florent, Professeur des universités, Université Claude Bernard Lyon	Examinateur

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Sylvie ROBERJOT Bât Atrium - UCB Lyon 1 04.72.44.83.62 Insa : H. CHARLES secretariat.e2m2@univ-lyon1.fr	M. Fabrice CORDEY CNRS UMR 5276 Lab. de géologie de Lyon Université Claude Bernard Lyon 1 Bât Géode 2 rue Raphaël Dubois 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 cordey@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Sylvie ROBERJOT Bât Atrium - UCB Lyon 1 04.72.44.83.62 Insa : M. LAGARDE secretariat.ediss@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Direction Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Direction mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT Tél : 04 78 69 72 76 viviane.polsinelli@univ-lyon2.fr	M. Christian MONTES Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Christian.montes@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Ma thèse de doctorat a occupé trois longues années de ma vie. Durant cette période, et j'imagine comme dans la plupart des thèses, j'ai connu des hauts et des bas. Lors de ces moments de joie et de fierté, mais aussi de désespoir et de remise en question, j'ai pu compter sur un certain nombre de personnes. Sans elles, il est possible que cette expérience n'ait jamais abouti.

En premier lieu, évidemment, je voudrais remercier mes encadrants : Guillaume Lavoué, Vincent Vidal et Florent Dupont. Honnêtement, je n'ai pas grand-chose à leur reprocher ... en fait, rien du tout. J'ai eu la chance de profiter de leurs conseils éclairés et surtout de leur présence bienveillante. Bref, si je devais renouveler l'aventure, j'aurais du mal à trouver meilleurs encadrants.

Je suis également très reconnaissant à l'INSA-Lyon de m'avoir accueilli dans ses murs et au LIRIS de m'avoir compté dans ses effectifs. J'ai bénéficié pendant trois ans d'un environnement de travail propice. Je profite d'ailleurs de l'occasion pour remercier chaleureusement les membres de l'équipe M2DisCo. J'ai adoré interagir avec eux. Mes travaux de recherche n'auraient sans doute pas été les mêmes sans leur participation.

Enfin, je ne pouvais pas conclure ces remerciements autrement qu'en exprimant un grand "MERCI" à ma femme, Sarah. Merci de m'avoir supporté pendant ces années. Merci de m'avoir soutenu et encouragé. Merci de m'avoir amené là où je suis aujourd'hui.

Résumé

Depuis plusieurs années, les modèles 3D deviennent de plus en plus détaillés. Cela augmente considérablement le volume de données les décrivant. Cependant, dans un même temps, un nombre croissant d'applications sont contraintes en mémoire et/ou en vitesse (visualisation sur périphériques mobiles, jeux vidéos, etc.). Dans un contexte Web, ces difficultés sont encore plus présentes. Cette situation peut entraîner des incompatibilités, des latences de transmission ou d'affichage qui sont souvent problématiques.

La compression progressive de ces modèles est une des solutions envisageables. Le but étant de compresser les informations (géométrie, connectivité et attributs associés) de façon à pouvoir reconstruire progressivement le maillage. À la différence d'une compression dite single-rate, la compression progressive propose très rapidement un aperçu fidèle du modèle 3D pour ensuite le raffiner jusqu'à retrouver le maillage complet. Ceci permet un meilleur confort pour l'utilisateur et une adaptation de la quantité d'éléments à visualiser ou à traiter en fonction des capacités du périphérique de réception.

Généralement, les approches existantes pour la compression progressive se focalisent sur le traitement de maillages 2-variétés triangulaires. Très peu de méthodes sont capables de compresser progressivement des maillages surfaciques non-variétés et, à notre connaissance, aucune ne permet de compresser génériquement un maillage surfacique quel que soit son type (i.e. non-variété et polygonal). Pour supprimer ces limitations, nous présentons une méthode de compression progressive générique permettant de traiter l'ensemble des maillages surfaciques (non-variétés et polygonaux). De plus, notre approche tient compte de l'attribut de texture potentiellement associé à ces maillages, en gérant correctement les coutures éventuelles.

Pour ce faire, nous décimons progressivement le maillage à l'aide d'un nouvel opérateur générique de simplification. Cette décimation est guidée par une métrique locale qui a pour but de préserver la géométrie et la paramétrisation de la texture. Durant cette simplification, nous encodons progressivement les informations nécessaires à la reconstruction. Afin d'améliorer le taux de compression, nous mettons en œuvre certains procédés de réduction de l'entropie, ainsi que des dispositifs de prédiction basés sur la géométrie pour l'encodage de la connectivité et des coordonnées de texture. Pour finir, l'image de texture est compressée progressivement puis multiplexée avec les données relatives au maillage. Ce multiplexage est réalisé à l'aide d'une métrique perceptuelle afin d'obtenir le meilleur rapport débit-distorsion possible lors de la décompression.

Abstract

Since several years, 3D models become more and more detailed. This increases substantially the amount of data needed to describe them. However, in the same time, a rising number of applications are constrained in memory and/or in speed (mobile device visualization, video games, etc.). These difficulties are even more visible within a Web context. This situation can lead to incompatibilities, latency in transmission or rendering, which is generally an issue.

The progressive compression of these models is a possible solution. The goal is to compress the information (geometry, connectivity and associated attributes) in order to allow a progressive reconstruction of the mesh. Contrary to a single-rate compression, progressive compression quickly proposes a faithful draft of the 3D model and, then, refines it until the complete mesh is recovered. This allows a better comfort for the user and a real adaptation of the rendered element number in adequacy with the terminal device properties.

The existing approaches for progressive compression mainly focus on triangular 2-manifold meshes. Very few methods are able to compress progressively non-manifold surface meshes and, to our knowledge, none can deal with every surface meshes (i.e. no-manifold and polygonal), in a generic way. So as to suppress these limitations, we present a new generic progressive method allowing the compression of polygonal non-manifold surface meshes. Moreover, our approach takes care of the texture attribute, possibly associated to these meshes, by handling properly potential texture seams.

For that purpose, we progressively decimate the mesh using a new generic simplification operator. This decimation is driven by a local metric which aims to preserve both the geometry and the texture parametrisation. During the simplification, we progressively encode the necessary information for the reconstruction. In order to improve the compression rate, we propose several entropy reduction mechanisms, as well as geometry based prediction strategies for the connectivity and UV coordinates encoding. Finally, the texture map is progressively compressed then multiplexed with mesh data. This multiplexing is achieved using a perceptual metric to provide the best rate-distortion ratio as possible during the decompression.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Informatique graphique

La révolution numérique, de la fin du 20ème siècle jusqu'à maintenant, a fortement contribué à forger le monde d'aujourd'hui. Pendant plusieurs décennies, l'informatique n'a eu de cesse de répondre aux besoins de la société, notamment au niveau de l'utilisation et de la diffusion des contenus multimédia. Une quantité importante de verrous ont été levés concernant différents supports comme la musique, l'image ou la vidéo. Néanmoins, il reste énormément de défis liés au monde du numérique. De nouvelles problématiques ont justifié l'émergence du domaine de l'informatique graphique. L'essor de cette discipline correspond à une tendance croissante et durable. Les secteurs d'applications sont d'ailleurs innombrables.

L'industrie du jeu vidéo est un des grands acteurs du domaine. Beaucoup de chemin a été parcouru depuis les premiers jeux évoluant dans un univers en 3 dimensions (3D) [1] comme *Battlezone*, en 1980, ou *Flight Simulator*, en 1982 (voir figure 1.1). Aujourd'hui, la rapidité des calculs, et donc la fluidité de l'affichage, ainsi que la qualité du rendu sont devenus une priorité pour répondre aux exigences du marché.

FIGURE 1.1 – Aperçus des jeux vidéos Battlezone (à gauche) et Flight Simulator (à droite).

D'autres domaines, comme l'animation, plus généralement pour la création de contenus vidéos, doivent répondre à d'autres problématiques. En effet, dans ce cadre précis, l'expérience visuelle se doit maintenant d'être la plus réaliste possible. Ces conditions

1. http://www.01net.com/astuces/quel-a-ete-le-premier-jeu-video-realise-en-3d-405989.html

ont porté de nombreuses innovations sur les techniques de rendu, mais également sur les méthodes et le matériel de calcul (par exemple, les cartes graphiques). Ces avancées permettent de mettre en œuvre les procédés d'illumination et d'ombrage nécessaires à une visualisation de qualité pour des scènes géométriquement complexes (voir figure 1.2).

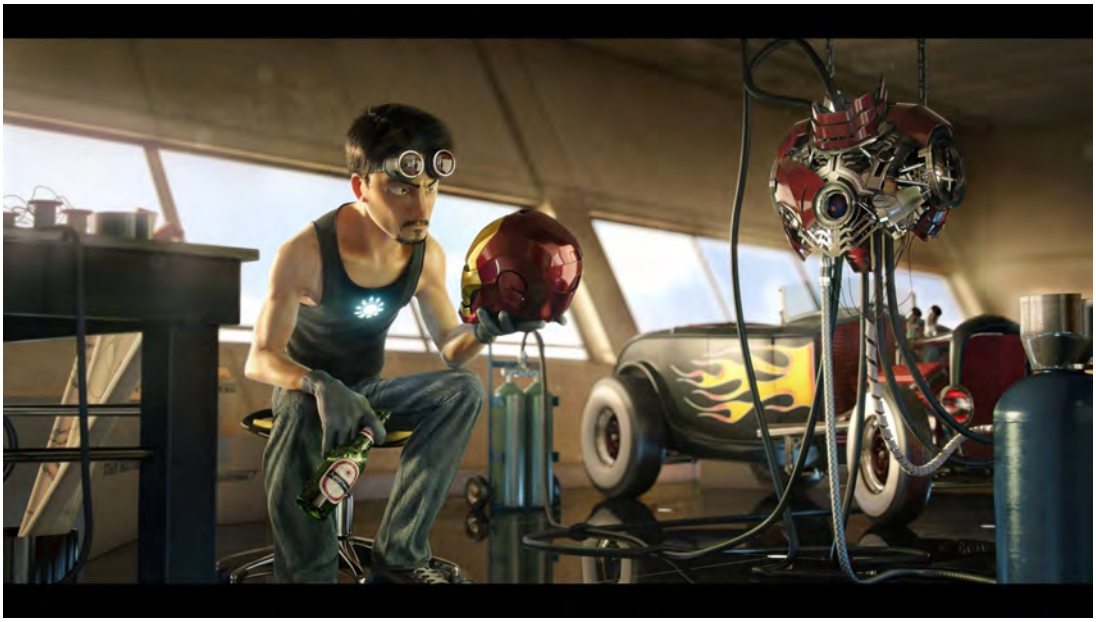


FIGURE 1.2 – Rendu 3D d'une scène inspiré du héros de Marvel, Ironman, par l'artiste Victor Hugo [3].

Il existe également des applications plus sérieuses (les jeux sérieux ou *serious games* en anglais) dans un contexte de simulation 3D, réaliste ou pas. Ces outils peuvent être utilisés à des fins d'apprentissage dans des secteurs d'activité aussi divers que la conduite de camion, l'aviation (civile ou militaire) ou la CAO [4] pour l'industrie (voir figure 1.3) et le bâtiment, sans oublier la stratégie militaire ; les *serious games* peuvent dans une certaine mesure permettre le développement des capacités cognitives ou de mémorisation du cerveau. La simulation 3D est également employée pour analyser et/ou interagir avec un système physique ayant des propriétés mécaniques particulières (par exemple, chirurgie assistée par ordinateur, voir figure 1.3, ou entraînement à l'accouchement pour les sage-femmes). Dans ce cadre, elle est régulièrement utilisée pour former des praticiens, réduire les coûts de certaines simulations réelles (par exemple, le secteur de l'automobile simule une grande partie de leurs crash-tests). Généralement, ces applications demandent une grande précision et donc une complexité de calcul encore problématique de nos jours.

Certains secteurs plus récents, comme la réalité virtuelle ou augmentée, l'impression 3D, le travail collaboratif par CAO et la visualisation grand public d'objets ou de scènes 3D (par exemple, la visite virtuelle d'appartements ou de musées), sont encore émergents. Mais ces derniers, afin de se démocratiser, profitent largement d'une autre grande tendance du 21ème siècle : le *Web*.

3. https://torugo.wordpress.com
4. CAO : Conception Assistée par Ordinateur
6. http://www.geomagic.com

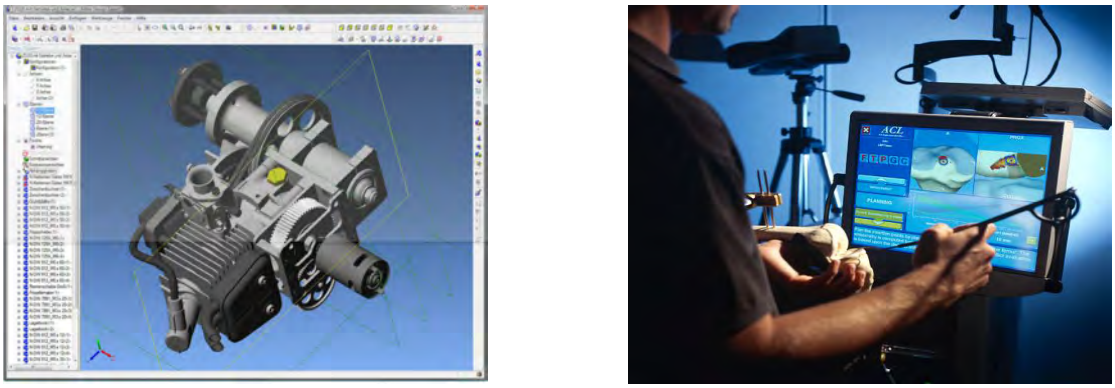


FIGURE 1.3 – À gauche, vue en perspective d'un moteur créé par CAO avec le logiciel Geomagic Design[6]. À droite, numérisation de données morphologiques du genou pour une opération assistée par ordinateur.

1.2 Transmission de données sur Internet

En effet, l'apparition et la croissance d'*Internet* dans la société a très fortement modifié les comportements. Nous tendons à nous affranchir de l'ordinateur tel que nous le connaissions pour gagner en mobilité. Ainsi, les périphériques utilisés deviennent multiples (ordinateur fixe, ordinateur portable, tablette, smartphone, etc.). Le stockage et les services s'adaptent en s'externalisant grâce aux nouvelles technologies comme le "*Cloud*" et son modèle SPI[7]. Les transmissions ont évolué, tant au niveau du réseau, avec différents protocoles et infrastructures, qu'au niveau de l'information traitée. En effet, de nombreux formats de données aux propriétés différentes existent. Ceux-ci permettent une efficacité grandissante des transmissions sur Internet. Quant aux traitements, ou applications, ils sont de plus en plus portés sur des serveurs distants afin de répondre à ce besoin de mobilité (par exemple, les applications SaaS ou Logiciel en tant que Service en Français). Cela a également permis de construire des outils de travail collaboratif dans de multiples domaines, aussi bien pour les professionnels que pour les particuliers. Avec ces nombreux paramètres, l'accès à des données sur Internet va alors dépendre des caractéristiques et des capacités du périphérique de réception utilisé ainsi que de la nature et de la qualité du réseau qui le dessert. Dans ce cadre, les données 3D n'échappent pas à la règle.

1.2.1 Les données 3D

Les objets 3D que l'on visualise et manipule au travers d'un écran sont représentés par un réseau d'éléments géométriques (sommets, arêtes, faces, etc.) appelé "maillage" (voir figure 1.4-gauche). Un maillage peut être caractérisé par la plus haute dimension des éléments qui le constituent. Nous parlons alors de maillages surfaciques pour des maillages ayant au maximum des éléments de dimension 2 (appelés faces).

Des données complémentaires de formes diverses peuvent également être associées à un maillage ou à ses éléments. Pour colorer un objet 3D, il est possible d'attacher à chaque sommet, ou à chaque face, une couleur. Il existe également un mécanisme plus complexe

7. modèle SPI : *Software as a Service* (SaaS), *Platform as a Service* (PaaS) et *Infrastructure as a Service* (IaaS).

9. http://www.cgal.org/

FIGURE 1.4 – À gauche, visualisation d'un maillage surfacique représentant un éléphant, venant de CGAL [9]. À droite, démonstration de l'ajout d'une texture en tant que donnée complémentaire au maillage TigerFighter.

qui consiste à utiliser une image, que l'on appelle alors texture, pour décorer les différentes faces du maillages (voir figure 1.4-droite). Ce type de données complémentaires se décompose en deux parties : l'image en elle-même et une paramétrisation indiquant comment plaquer cette image sur le maillage. Un autre exemple d'attribut de sommet peut être la normale (ou vecteur normal). Celle-ci définit la direction du plan tangent à la surface en un point donné. Elle est largement utilisée, entre autres, pour les calculs d'illumination nécessaires à la visualisation d'un objet 3D. Cette liste d'attributs possibles n'est pas exhaustive. Il existe de nombreux labels ou propriétés physiques utilisés, notamment, dans le domaine de la simulation ou du bâtiment (notamment avec les formats BIM [10] et CityGML [11]).

Un maillage a l'avantage d'être une structure discrète. Grâce à l'architecture parallèle du GPU [12] d'un ordinateur ou de tout autre périphérique déjà évoqué plus haut, il est possible d'afficher efficacement ce type de structure. Néanmoins, aujourd'hui, une grande quantité de maillages ont une résolution élevée, voir très élevée. Cela s'explique par la précision croissante des systèmes de production 3D, que ce soit la numérisation d'objets existants (par laser, photogrammétrie, etc.), la création artistique par des infographistes, ou la construction procédurale. Chacune de ces méthodes est capable de générer un maillage très détaillé et donc potentiellement volumineux sur disque (de l'ordre de la centaine de Mo [13]).

Le traitement de telles masses de données, de manière locale, sur un ordinateur actuel est assez bien géré. Cependant, dès lors qu'il s'agit de manipuler des maillages stockés à distance, au travers d'un réseau, par exemple dans le cas d'une visualisation de données 3D sur le Web, plusieurs problèmes se posent, comme le temps de transfert ou la capacité de traitement des périphériques de réception.

10. BIM :Building Information Modeling
11. City Geography Markup Language
12. GPU :*Graphic Processing Unit* ou Processeur Graphique en Français.
13. Mo :megaoctet

FIGURE 1.5 – Visualisation du modèle Dwarf (1,288,973 sommets et 2,577,938 faces pour 319 Mo + 19 Mo de texture). Ce modèle a été acquis par numérisation.

1.2.2 Stockage

Dans de nombreuses situations, les données 3D à traiter sont hébergées sur un(des) serveur(s) distant(s). Malgré le fait qu'une grosse partie du coût de fonctionnement de ces serveurs soit liée au nombre d'entrées/sorties réalisées (ce qui influence le type de matériel utilisé), le volume de données stockées reste un facteur important. En effet, ce volume conditionne le nombre d'unités de stockage utilisées et donc la puissance électrique nécessaire au maintien de ces équipements. Évidemment, cette énergie a un coût. Ainsi, économiquement, il y a un intérêt certain à réduire, par tous les procédés possibles, la taille des données en question.

1.2.3 Réseau

Dans un contexte de manipulation de données non locales, des difficultés peuvent être rencontrées lors de leur transmission. Selon une étude [14] sur l'année 2013, le débit moyen d'une connexion Internet (tous périphériques et types de réseau confondus) serait, au mieux, de 12 Mbit/s [15] (au Pays-Bas). En 2015, le site Hidden Haze [16] annonce une moyenne mondiale de 4.5 Mbit/s. En prenant l'exemple du modèle 3D Dwarf (voir figure 1.5), afin de transmettre la totalité des 340 Mo de ce maillage, il faudrait attendre en moyenne 10 minutes ! Il est donc impossible, dans l'état, d'assurer de façon globale une interactivité sur des maillages de cette taille. Il faut également ajouter que le comportement d'un utilisateur moyen a nettement évolué depuis l'émergence d'Internet. Aujourd'hui, la rapidité de transmission et d'affichage du contenu souhaité est primordiale dans la quasi totalité des applications.

14. http://www.ariase.com/fr/news/etude-internet-akamai-t4-2013-article-3259.html
15. Mbit/s :megabit par seconde
16. https ://hazelifestyle.wordpress.com/

1.2.4 Diversité des périphériques

Avec la démocratisation des smartphones et autres appareils mobiles, la gamme des périphériques ne cesse de grandir. Cela a un impact sur la façon dont les applications en ligne sont mises en œuvre, mais également sur la forme des données transmises. L'apparition de périphériques aux performances et aux caractéristiques diverses (voir figure 1.1) impose des systèmes souples et adaptatifs afin de répondre au mieux à chaque utilisateur. Effectivement, il est souvent important de prendre en compte des propriétés comme le stockage de l'appareil, sa mémoire vive, ses performances réseau, ses capacités d'affichage ou la résolution et la taille de son écran. Il n'est, par exemple, pas raisonnable de penser que l'on puisse télécharger et visualiser des objets 3D de la même façon sur un ordinateur de bureau que sur un smartphone.

	Stockage	Mémoire vive	GPU Flop	Résolution
Iphone 6	16/64 Go	1 Go	115.2 GFlops	5.5" (1334x750p)
Samsung S7	32 Go	4 Go	265.2 GFlops	5.1" (2560x1440p)
Microsoft Surface 3	128 Go	4 Go	153.6-163.2 GFlops	10.8" (1920x1080p)
Ipad Air 2	16/64 Go	2 Go	230.4 GFlops	9.7" (2048x1536p)
Lenovo Ideapad Y700-17"	1 To	8 Go	1024 GFlops	17.3" (1920x1080p)
Dell XPS 15	256/512/1024 Go	8/16/32 Go	1024 GFlops	15.6" (1920x1080p)

Tableau 1.1 – Tableau comparatif de différents périphériques de réception : deux smart phones (Iphone 6, Samsung S7), deux tablettes (Surface 3, Ipad Air 2) et deux ordinateurs portables (Ideapad Y700-17", XPS 15).

1.3 Problématique

À la lumière de ces faits, une problématique se dégage. Comment peut-on manipuler efficacement des données 3D distantes de façon adaptée pour chaque type de terminal ?

Plusieurs solutions, ou pistes de réflexion, permettent de lever un certain nombre de verrous. L'idée principale, déjà largement adoptée pour la manipulation d'autres médias comme l'image ou la vidéo, est d'utiliser la compression. Cette approche diminue le volume des données traitées. Ceci offre une facilité de stockage et, par là même, une réduction du temps de transmission.

Malheureusement, il est encore difficile d'obtenir une compression des données 3D s'adaptant parfaitement à la diversité des périphériques. En outre, les méthodes de compression existantes ne permettent pas de traiter tous les types de maillages qui, eux aussi, peuvent avoir différentes caractéristiques (notamment topologiques). De plus, les données additionnelles associées aux maillages sont souvent laissées de côté par la majorité des algorithmes de compression existants. Pourtant, elles sont souvent essentielles, que ce soit pour de l'analyse (scientifique ou industrielle) ou pour l'obtention d'un rendu réaliste des modèles 3D.

1.4 Contributions

Dans ce contexte, cette thèse présente nos travaux réalisés durant trois années à l'INSA de Lyon autour de ces problématiques. Nous proposons une méthode de compression prenant en charge un maillage surfacique arbitraire (pouvant être polygonal et non-variété), ainsi que certains de ses attributs (normales et textures). Notre approche permet également une décompression progressive qui offre une visualisation immédiate de la globalité du maillage sous une forme simplifiée. Le processus de décompression raffine, au fur et à mesure de la transmission, la version simplifiée pour finalement obtenir l'intégralité du maillage distant sans perte d'information. Nos travaux apportent plusieurs contributions :

— De nouveaux opérateurs de simplification et de raffinement locaux pouvant être appliqués aux maillages surfaciques, quelle que soit leur topologie. Ces opérateurs prennent également en compte les attributs de normales et de textures, dans toutes les configurations possibles.
— Une nouvelle métrique pour la sélection des sites de simplification permettant de minimiser la distorsion engendrée sur la géométrie du maillage et la paramétrisation de son attribut de texture.
— De nouveaux schémas de prédiction géométrique et de réduction d'entropie pour le codage de la connectivité du maillage, ainsi que pour la paramétrisation de la texture.

Afin de respecter au mieux le caractère progressif de la méthode, nous utilisons également un encodage progressif de l'image de texture. Les niveaux de détails de la texture sont alors multiplexés aux niveaux de détails du maillage de façon à optimiser la qualité visuelle en fonction du coût en terme de taille des données transmises.

1.5 Vue d'ensemble

Le reste de ce manuscrit est organisé en quatre grandes parties.

Le Chapitre 2 débute par la présentation des différentes notions de bases qui seront manipulées par la suite. Nous présentons ensuite un état de l'art de différents domaines liés à la compression de maillages surfaciques. Cette partie permet de poser la base de la réflexion qui nous a conduit à la réalisation de notre méthode de compression.

Le Chapitre 3 décrit en profondeur les divers mécanismes constituant notre méthode. Nous expliquerons alors le parcours d'un maillage de son premier chargement dans notre structure jusqu'à la visualisation interactive par le périphérique de réception, en passant par les multiples stratégies d'encodage et de réduction de l'entropie.

Le Chapitre 4 présente les résultats obtenus à l'aide de notre méthode. Ces expérimentations ont pour but de valider plusieurs stratégies utilisées durant la compression et la décompression. De plus, afin de situer notre approche au sein de l'état de l'art, nous proposons plusieurs comparaisons, quantitatives et qualitatives, avec d'autres méthodes existantes.

Enfin, le Chapitre 5 résume notre cheminement et conclut sur les travaux engagés. Nous évoquons également les limites de notre méthode ainsi que les différentes pistes de

recherche détectées qu'il resterait à explorer.

Chapitre 2

État de l'art

2.1 Préliminaires

2.1.1 Éléments composant un maillage

Tout au long de ce manuscrit, nous manipulerons différents éléments géométriques composant un maillage 3D (voir figure 2.1). Les sommets sont des éléments de dimension 0 ; les arêtes, composées de 2 sommets, sont de dimension 1 ; les faces ou polygones, composés de n arêtes formant un cycle, sont de dimension 2. Les faces comportant trois arêtes sont dites triangulaires. Il existe des éléments de dimension supérieure à 2, mais nous ne les aborderons pas ici.

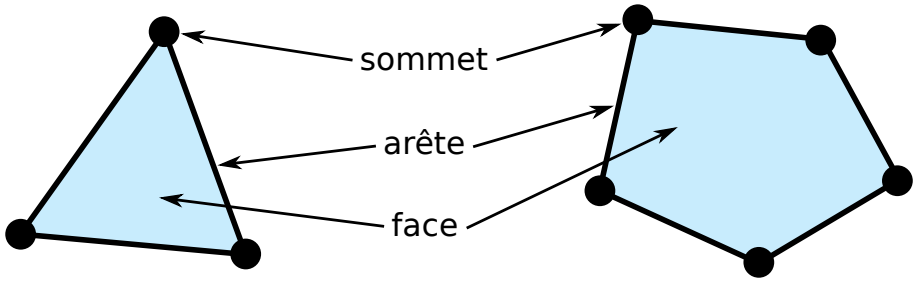


FIGURE 2.1 – Définition des différents éléments géométriques constituant un maillage surfacique.

Afin de décrire correctement un maillage, ses éléments et leurs relations, nous proposons les définitions suivantes :

Définition 1. *Un ensemble d'éléments géométriques peut être organisé sous forme de maillage. Ces éléments sont alors réunis entre eux par des relations d'incidence et d'adjacence (voir figure 2.2).*

Définition 2. *Deux éléments, e et e', de dimension d et d' (avec $d \neq d'$) sont dits incidents si e compose e' ou si e' compose e. On note alors $e \prec e' \Leftrightarrow e' \prec e$. Deux éléments, e et e', de dimension d et d' (avec $d = d'$) sont dits adjacents s'il existe un élément e'' de dimension $d-1$ ($d+1$ lorsque $d = 0$) tel que $e \prec e''$ et $e' \prec e''$. On note alors $e \succ e' \Leftrightarrow e' \succ e$.*

Définition 3. *La valence d'un élément e de dimension d est le nombre d'éléments e' de dimension $d+1$ tels que $e \prec e'$.*

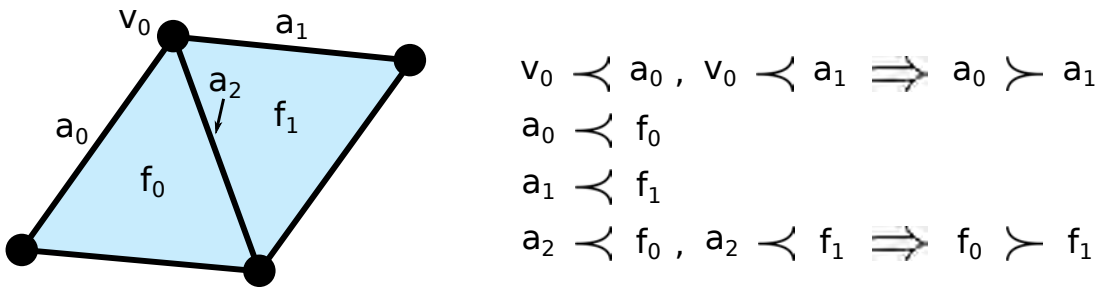


FIGURE 2.2 – Relations d'incidence et d'adjacence entre les différents éléments géométriques d'un maillage surfacique.

Un exemple de relations d'incidence et d'adjacence présentes au sein d'un maillage est donné dans la figure 2.2. La notion de valence est notamment utilisée pour catégoriser les sommets et les arêtes d'un maillage (voir figure 2.3). Nous différencions ces deux types d'éléments grâce aux définitions suivantes :

Définition 4. *Un sommet de valence $val_s = 0$ est un sommet isolé.*

Définition 5. *Une arête de valence val_a est :*
- *une arête isolée si $val_a = 0$ et les valences de ses sommets incidents sont toutes égales à 1.*
- *une arête pendante si $val_a = 0$ et la valence de l'un de ses sommets incidents est supérieure à 1.*
- *une arête de bord si $val_a = 1$.*
- *une arête normale si $val_a = 2$.*
- *une arête complexe si $val_a > 2$.*

À l'aide des définitions 4 et 5, il est également possible de distinguer différentes catégories de faces de la façon suivante (voir figure 2.3) :

Définition 6. *Une face est dite :*
- *isolée si les valences de ses arêtes incidentes sont égales à 1 et celles de ses sommets incidents à 2.*
- *pendante si les valences de ses arêtes incidentes sont égales à 1 et que la valence de l'un de ses sommets incidents est supérieure à 2.*

De plus, on définit la notion de degré d'une face telle que :

Définition 7. *Le degré d'une face est égal au nombre de ses arêtes incidentes.*

Nous présentons ensuite la notion de patch, illustrée par la figure 2.4 :

Définition 8. *Un patch est un sous ensemble du maillage relatif à un élément. Le patch d'un sommet s représente l'ensemble des faces f tel que $s \prec f$. Le patch d'une arête a, composée des sommets s et s', est l'union des faces f telles que $s \prec f$ et des faces f' telles que $s' \prec f'$.*

2.1.2 Différent types de maillages

Un maillage formé d'éléments e de dimension d telle que $d < 3$ est dit surfacique. Nos travaux traitent uniquement des maillages surfaciques. Donc, dans un souci de lisibilité,

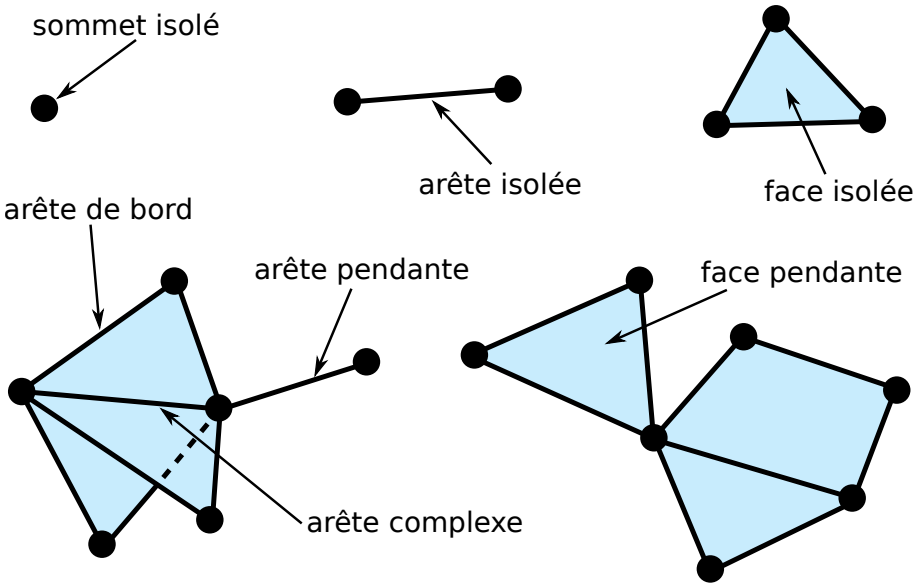


FIGURE 2.3 – Différentes caractéristiques des éléments géométriques d'un maillage surfacique.

le terme maillage sera utilisé par la suite pour décrire un maillage surfacique.

De la même façon que les surfaces dans un espace Euclidien en 3D, un maillage a plusieurs caractéristiques intéressantes dont il faut généralement tenir compte lors de sa manipulation. Pour notre part, nous utilisons les caractéristiques topologiques d'un maillage pour le situer dans trois catégories : les maillages 2-variétés, les maillages 2-variétés à bord et les maillages non-variétés. Afin de déterminer la catégorie d'un maillage, nous utilisons la notion d'homéomorphisme.

Définition 9. *Un maillage $\mathcal{M}$ est homéomorphe à un maillage $\mathcal{M}'$ si et seulement s'il existe une déformation continue permettant de passer de $\mathcal{M}$ à $\mathcal{M}'$ et réciproquement.*

Les différentes catégories de maillages sont alors définies de cette manière :

Définition 10. *Un maillage $\mathcal{M}$ est dit 2-variété (ou variété de dimension 2) s'il est, en tout sommet, localement homéomorphe à un disque.*

Définition 11. *Un maillage $\mathcal{M}$ est dit 2-variété à bord s'il est localement homéomorphe à un demi-disque, sur les bords, et à un disque, à l'intérieur.*

En pratique, pour ce qui concerne les définitions 10 et 11, un maillage est 2-variété (resp. 2-variété à bord) si, pour toute arête a de $\mathcal{M}$, la valence de a, noté val, est tel que $val = 2$ (resp. $1 \leq val \leq 2$). Un maillage 2-variété ne comporte pas de sommet isolé, ni d'arête isolée, pendante ou complexe. Dans le cas contraire, cela devient un maillage non-variété.

En vertu des contraintes intrinsèques des maillages 2-variétés (avec ou sans bord), leur manipulation est grandement facilitée. En effet, la régularité de leurs connectivités permet la mise en place de structures et d'algorithmes très performants. À l'inverse, il est nettement plus compliqué de traiter des maillages non-variétés. Le caractère totalement libre de leur connectivité complique leur manipulation. Toutefois, il est souvent essentiel de les prendre en compte pour deux raisons principales. Premièrement, les structures ou algorithmes traitant des maillages non-variétés sont également capables de gérer des

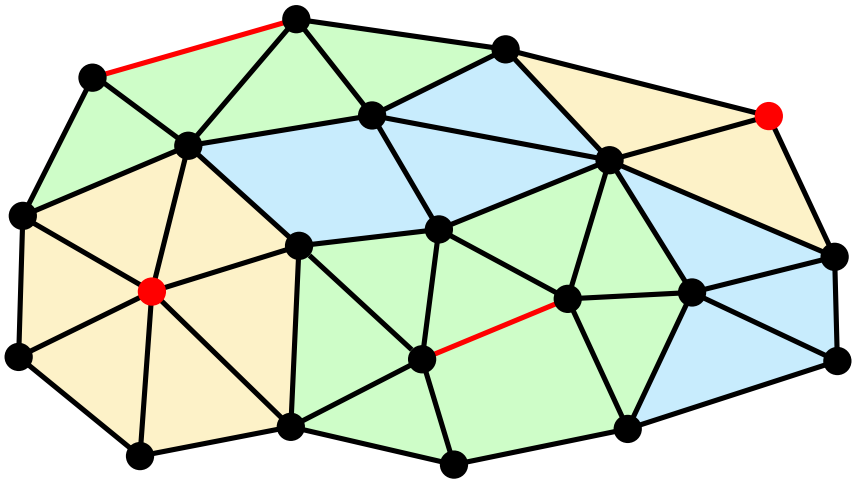


FIGURE 2.4 – Illustration des patchs relatifs aux sommets et aux arêtes sur un maillage surfacique. Pour chaque sommet rouge, les faces jaunes autour du sommet représentent le patch du sommet. Pour chaque arête rouge, les faces vertes autour de l'arête représentent le patch de l'arête.

maillages variétés (avec ou sans bord). Ensuite, du fait des techniques d'acquisition de maillages ou du besoin des designers, un grand nombre de maillages existants sont non-variétés. Il peut donc être intéressant de pouvoir les manipuler tels quels, sans modifier leur topologie pour les transformer en maillages 2-variétés.

2.1.3 Données attachées aux maillages

Sous leur plus simple forme, les maillages ne sont constitués que d'éléments géométriques (sommets, arêtes, faces) et des relations d'incidences qui les lient entre eux au moyen d'éléments de dimensions supérieures. L'ensemble de ces relations est appelé la connectivité du maillage. À ces éléments géométriques, il est possible d'associer différentes informations (voir figure 2.5). Ces dernières peuvent également être relatives au maillage lui-même, ou encore à une combinaison d'éléments.

Géométrie

L'information la plus utilisée est la position géométrique des sommets dans un espace donné. Ces données complémentaires décrivent la géométrie du maillage. Elle est nécessaire pour la visualisation du maillage et le calcul de la plupart des propriétés géométriques couramment utilisées.

Couleurs

Afin de visualiser un maillage, il est souvent utile de fournir d'autres propriétés que la géométrie. La coloration des faces du maillage est alors une solution pour augmenter la qualité visuelle du rendu. Cette information de couleur est donnée soit par sommet soit par face. Si la couleur est relative aux faces, dans ce cas le GPU dessinera à l'écran l'élément concerné dans la couleur qui lui est attachée. Si elle est relative aux sommets, alors le

GPU devra faire une interpolation des différentes couleurs de chaque sommet d'une face afin d'afficher un résultat cohérent. Cela permet d'obtenir des dégradés et d'éviter les frontières nettes de couleurs lors de la visualisation du maillage. La donnée en elle-même peut prendre plusieurs formes. Cela peut être une combinaison de trois valeurs flottantes entre 0 et 1 (ou quatre pour la transparence), une combinaison de trois ou quatre valeurs entières entre 0 et 255 (pour des coordonnées RGB et RGBA, par exemple), ou un indice correspondant à une couleur prédéfinie.

Normales

L'affichage du maillage gagne également en qualité visuelle grâce à l'utilisation de données géométriques supplémentaires. Les normales aux sommets en sont l'exemple. Soit f_i une face du patch d'un sommet s ($0 \leq i \leq n_f^s$, n_f^s étant le nombre de faces du patch), la normale à s, notée $\overrightarrow{n}$, est obtenue par la moyenne des normales $\overrightarrow{n}_{f_i}$:

$$\overrightarrow{n} = \frac{1}{n_f^s} \sum_{i=0}^{n_f^s} \overrightarrow{n}_{f_i} \tag{2.1}$$

Pour une mesure plus précise de la normale à un sommet, il est possible de pondérer les normales de faces f_i par leur aires $\mathcal{A}_{f_i}$:

$$\overrightarrow{n} = \frac{1}{\mathcal{S}_{\mathcal{A}_{f_i}}} \sum_{i=0}^{n_f^s} \mathcal{A}_{f_i} \times \overrightarrow{n}_{f_i} \tag{2.2}$$

$$\mathcal{S}_{\mathcal{A}_{f_i}} = \sum_{i=0}^{n_f^s} \mathcal{A}_{f_i} \tag{2.3}$$

Le calcul de la normale à une face est direct si celle-ci est planaire, i.e. tous ses sommets font partie d'un même plan. Les faces triangulaires sont par définition planaires. Cependant, ce n'est pas forcément le cas des faces de degré supérieur à 3. Si une face polygonale f n'est pas planaire, la méthode la plus simple est de déterminer $\overrightarrow{n}_f$ comme la normale au plan moyen défini par l'ensemble de ses sommets.

Les normales aux sommets permettent au GPU de ne plus utiliser la géométrie des faces, qui généralement fait ressortir le caractère discret de l'objet 3D lors de l'affichage. À la place, le GPU se sert de l'interpolation des normales en chaque sommet d'une face pour réaliser un rendu lisse. Dans de nombreux cas, cela augmente la qualité visuelle de l'objet traité.

Texture

L'utilisation de couleurs pour la décoration d'un maillage peut avoir ses avantages mais elle a également ses limites. Un maillage peut comporter des schémas complexes de couleurs ou le rendu de l'objet 3D peut avoir besoin de beaucoup de réalisme. Cela

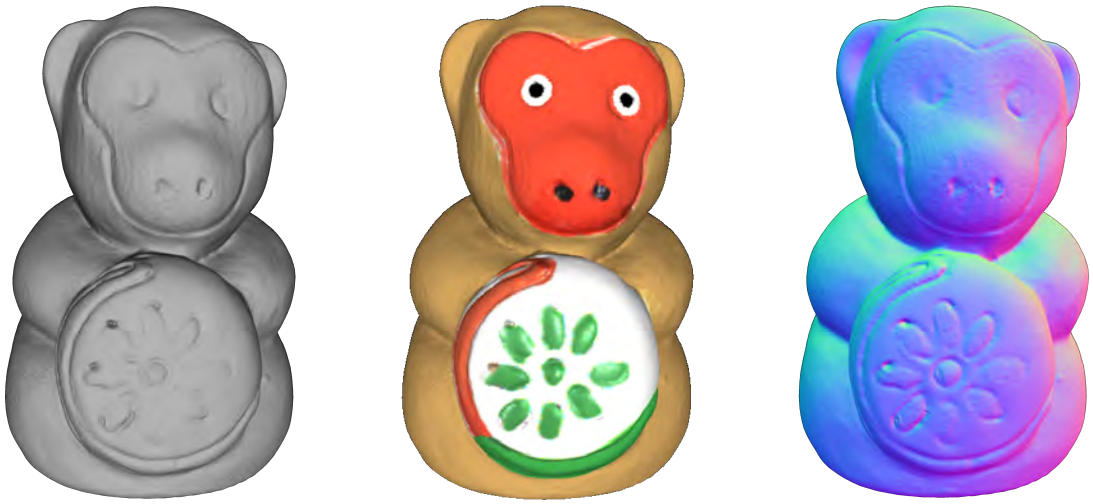


FIGURE 2.5 – Différents rendus du modèle *Monkey*. De gauche à droite, le modèle *Monkey* avec connectivité et géométrie ; le modèle *Monkey* avec connectivité, géométrie et couleurs ; le modèle *Monkey* avec connectivité, géométrie et normales (transformées ici en couleur). Les trois modèles sont affichés grâce à un rendu non lisse.

demande un nombre élevé de sommets afin d'obtenir la précision visuelle voulue, que l'information de couleur soit relative aux sommets ou aux faces.

Il existe une solution permettant d'ajouter une grande quantité de détails en conservant un nombre relativement modéré de sommets. Cela passe par l'utilisation d'images en 2 dimensions (2D), appelées textures. La correspondance entre les faces du maillage et des parties de la texture est appelée la paramétrisation de la texture. La paramétrisation exprime la façon dont la texture est plaquée sur le maillage en augmentant ainsi la qualité visuelle à moindre frais.

Il est possible d'associer plusieurs textures à un maillage. Celles-ci ne sont pas toujours de même résolution et ne comportent pas forcément le même type d'informations. Le plus souvent il s'agit de couleurs, mais on peut trouver également des normales, des reliefs, des déplacements, etc. Néanmoins, dans l'ensemble, quel que soit le type de données exprimées par le biais d'une texture, le but principal est de fournir des détails en minimisant la résolution de la géométrie.

Autres types de données complémentaires

Il existe une infinité d'attributs pouvant être attachés à un maillage ou à ces éléments. On trouve aussi bien des données géométriques, comme des courbures, que des données physiques, comme des forces. Un maillage est complet quand il est composé de sa connectivité et de l'ensemble des informations complémentaires nécessaires à son utilisation.

2.1.4 Paramétrisation de la texture

Si l'usage des textures de couleur est un outil formidable, certaines difficultés peuvent être rencontrées durant leur mise en place. En effet, en fonction des propriétés de la texture, les informations ne sont pas relatives aux mêmes éléments. Il existe donc trois

façons d'attacher la paramétrisation de la texture au maillage (par sommet, par *corner* et par *wedge*). Ces différentes approches sont décrites plus loin. Dans tous les cas, l'objectif est de relier des coordonnées 2D (ou coordonnées UV), correspondant à une valeur sur la texture, aux éléments choisis du maillage.

Région de texture

Une texture n'étant qu'un ensemble de données sur une image 2D, c'est la paramétrisation qui va définir ses propriétés. Tout d'abord, la paramétrisation définit une relation entre une face du maillage et une face de la texture de même degré. Une paramétrisation est continue si et seulement si, pour toutes faces f_0 et f_1 du maillage ainsi que f'_0 et f'_1 de la texture, avec $f_0 \neq f_1$ et $f'_0 \neq f'_1$, $f_0 \succ f_1 \Leftrightarrow f'_0 \succ f'_1$. Dans le cas contraire, elle est dite discontinue. Ces discontinuités sont appelée les coutures de la texture. Les coutures définissent les frontières des différentes régions de texture. Une région représente alors un ensemble connexe de la texture (voir figure 2.6).

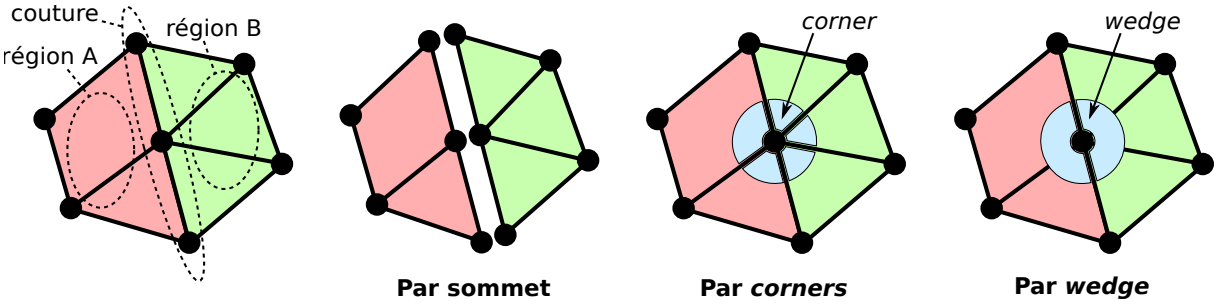


FIGURE 2.6 – Un patch associé à deux régions de texture différentes peut être modélisé de différentes façons. Les coordonnées UV par sommet obligent à opérer un changement topologique (dédoubler certains sommets). Les *corners* sont représentés par des couples (sommet,face). Les *wedges* représentent une concaténation des *corners* de même coordonnées UV, autour d'un sommet donné.

Par sommet

Lorsque la paramétrisation d'une texture est continue, attacher les coordonnées 2D aux sommets est direct. Par extension de la propriété de continuité, pour tout sommet s_0, s_1 du maillage et s'_0, s'_1 de la texture $s_0 \succ s_1 \Leftrightarrow s'_0 \succ s'_1$. Cette configuration est évidemment la moins coûteuse, et permet d'utiliser seulement n_s^p paires de coordonnées, où n_s^p est le nombre de sommets impliqués dans la paramétrisation.

Dans le cas d'une paramétrisation originellement discontinue, il est possible de dupliquer les sommets du maillage, créant ainsi une composante connexe par région de texture (voir figure 2.6). Cependant, une telle opération modifie fortement la connectivité du maillage et par conséquent sa topologie.

Par *corner*

Afin de prendre en charge une paramétrisation de texture discontinue, sans modifier la topologie du maillage, ses données peuvent être attachées à des couples (sommet, face)

appelés *corners* [Hop98] (voir figure 2.6). Une face du maillage possède autant de *corners* qu'elle a de sommets. Un sommet est associé à autant de *corners* qu'il a de faces incidentes. Si un *corner* est affecté par la paramétrisation, celui-ci sera lié à des coordonnées UV.

L'utilisation de *corners* permet de gérer correctement les coutures de la texture, quelle que soit la connectivité du maillage. Cela engendre néanmoins un surcoût car il faut stocker $\sum_{i=0}^{n_f^p} degré(f_i)$ *corners*, avec n_f^p le nombre de faces du maillage affectées par la paramétrisation.

Par *wedge*

Afin d'optimiser la gestion de la texture par *corners*, Hoppe [Hop98] définit un nouvel élément : le *wedge* (voir figure 2.6). Ce dernier représente une collection de *corners* adjacents ayant les mêmes coordonnées UV. Cette factorisation réduit le volume des données à conserver. Cela est d'autant plus efficace qu'il est rare de rencontrer un sommet dont le patch est associé à plus de deux régions.

2.1.5 Métriques de distance

Durant nos travaux, il nous a fallu mesurer les distorsions engendrées sur un maillage donné. Pour cela, il existe plusieurs métriques. Nous avons choisi de n'en retenir que deux : la distance MRMS [1] et MS-SSIM [2].

La métrique MRMS implémentée dans METRO[CRS98] détermine la distance géométrique entre deux maillages, $\mathcal{M}$ et $\mathcal{M}'$, de la façon suivante :

$$MRMS(\mathcal{M}, \mathcal{M}') = max(RMS(\mathcal{M}, \mathcal{M}'), RMS(\mathcal{M}', \mathcal{M})) \tag{2.4}$$

$$RMS(\mathcal{M}, \mathcal{M}') = \frac{1}{|\mathcal{M}|} \sum_{i=0}^{n_s} e(s_i, \mathcal{M}') \tag{2.5}$$

$$e(s, \mathcal{M}') = \min_{f' \in \mathcal{M}'} d(s, f') \tag{2.6}$$

où $|\mathcal{M}|$ et n_s sont respectivement la surface et le nombre de sommets d'un maillage $\mathcal{M}$, s est un sommet d'un maillage $\mathcal{M}$, f' est une face d'un maillage $\mathcal{M}'$ et $d(s, f')$ est la plus petite distance Euclidienne entre s et f'. Pour plus de précision dans la mesure de distance, il est possible d'échantillonner plus finement les deux surfaces à comparer.

Cette distance est pratique pour quantifier une distorsion géométrique entre deux versions d'un même maillage car elle peut être simple et rapide à calculer, si la stratégie d'échantillonnage n'est pas trop coûteuse. Néanmoins, c'est une métrique globale qui a ses limites.

MS-SSIM [WSB03a], quant à elle, est une métrique perceptuelle entre images 2D. Elle détecte les changements de structure dans l'image. Ces travaux ont montré que cette

1. MRMS :Maximum Root Mean Square
2. MS-SSIM :Multi-Scale Structural Similarity For IMage Quality Assessment

métrique donne d'excellents résultats (corrélés avec la vision humaine) pour l'évaluation de la fidélité visuelle entre images naturelles. Lavoué *et al.* [LLV16] ont également démontré que cette approche pouvait être employée efficacement pour la comparaison de maillages. Ainsi, nous utilisons cette métrique afin d'évaluer la fidélité visuelle d'un maillage texturé par rapport à un autre. Notre mise en œuvre de ces travaux détermine un score (entre 0 et 1) sur une série d'images de rendu des objets à comparer. Ces images sont prises, pour chacun des deux maillages, à partir de 42 points de vue régulièrement échantillonnés sur une sphère entourant le modèle. Cette métrique nous permet ainsi de déterminer la perte de qualité visuelle d'un maillage texturé par rapport à une référence. Le caractère multi-échelle de la métrique offre également une plus grande robustesse.

2.2 Structures de données appliquées aux maillages

Comme nous l'avons vu dans la Section 2.1, les données 3D sont généralement organisées en maillages. Chaque maillage à ses propriétés topologiques et sa connectivité. Ainsi, afin de stocker ces données, plusieurs structures existent. La structure utilisée est également dépendante du traitement que l'on souhaite effectuer sur le maillage (par exemple, une modification de la connectivité). Un choix judicieux de la structure de données permet d'optimiser l'efficacité des manipulations, tant en terme de temps d'exécution que de taille de stockage. Il existe différents types de structures de données. Chacune a ses avantages et ses limites.

2.2.1 Structures compactes

Les structures de données compactes sont étudiées depuis très longtemps. Le but premier de ces travaux est de déterminer quelle serait la meilleure représentation possible pour minimiser la taille des données, pour un type de maillage donné.

Gurung *et al.* [GLLR11a] se servent de la notion de *corner*, généralement utilisée dans la gestion de la texture, suivant les travaux de Rossignac *et al.* [Ros01], afin de définir les relations existant entre les différents éléments du maillage. Leur contribution réside dans le fait de regrouper, à chaque fois que cela est possible, les triangles d'un maillage 2-variété par paire. Chaque paire forme alors un quadrilatère sur lequel sont définis plusieurs opérateurs d'accès basés sur les *corners*. Les relations d'adjacence et d'incidence sont stockées en utilisant deux tableaux de correspondance entre les sommets et leurs *corners* et un tableau de correspondance entre les faces et leurs *corners*. L'appariement des triangles permet de diminuer le nombre de références stockées (autour de 2 par triangle) en optimisant les opérateurs d'accès. Cette approche a également l'avantage d'offrir une gestion directe des maillages quadrangulaires.

D'autres travaux de Gurung *et al.* [GLLR11b] proposent une représentation permettant de stocker efficacement des maillages triangulaires 2-variétés (voir figure 2.7). Ils recherchent un cycle Hamiltonien de faces en parcourant le maillage. Ce cycle leur permet de structurer de manière consistante les triangles concernés pour fournir une lecture en *random access* des éléments incidents et adjacents à un sommet ou à une face. L'utilisation de la même approche d'appariement de triangles rend la structure encore plus compacte (autour d'une référence par triangle) avec les mêmes performances d'accès.

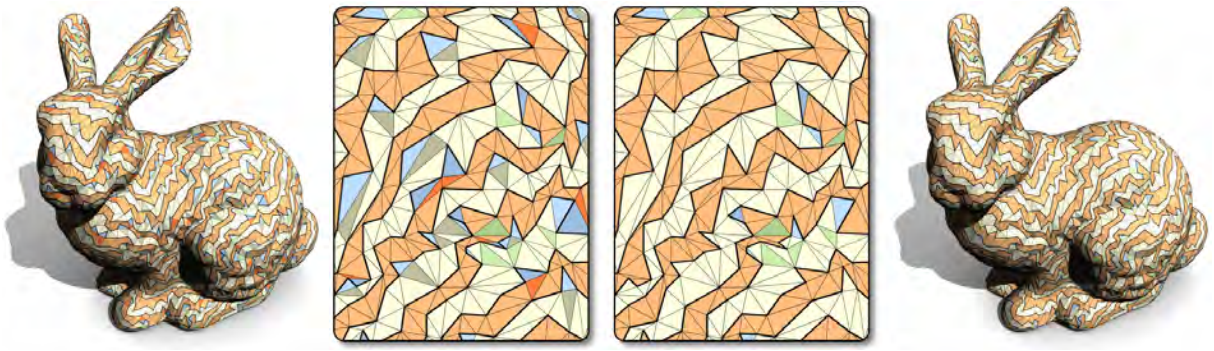


FIGURE 2.7 – Visualisation de la structure de Gurung *et al.* [GLLR11b].

Nous constatons que les structures compactes comme celles présentées ci-dessus ont la particularité de diminuer drastiquement la taille des données en conservant des temps d'accès compétitifs. Néanmoins, elles ont également leurs limites. En effet, tout changement topologique sur ce type de structure entraîne irrémédiablement une refonte partielle ou totale de la connectivité du maillage. Ces changements sont donc coûteux et contraignent l'usage de ces structures à des applications pour lesquelles le maillage conserve une connectivité fixe.

De plus, les travaux dans ce domaine ne se préoccupent généralement que des maillages 2-variétés. Ainsi, il est difficile de trouver une structure compacte permettant de stocker efficacement des surfaces, quelle que soit leur connectivité.

2.2.2 Structure pour maillages variétés et quasi-variétés

Dans la littérature, plusieurs travaux ont pris le parti de conserver plus d'information de connectivité afin d'augmenter les performances d'accès et d'obtenir une manipulation efficace des données, notamment en terme de modifications topologiques.

Différentes structures partagent ces propriétés. Baumgart [Bau72] se sert du concept de *winged edge* (voir figure 2.8) pour stocker la connectivité d'un maillage. Une *winged edge* est une arête conservant les références sur ses deux sommets et ses deux faces incidentes, ainsi que sur l'arête précédente et suivante de chacune de ces faces. Cet agencement permet un parcours immédiat de la structure de manière locale. Les opérations topologiques sur cette structure ne nécessitent qu'une modification locale de la connectivité, ce qui la rend plus efficace, en ce sens, qu'une structure compacte. Le coût mémoire est logiquement plus élevé. Néanmoins, le concept même de *winged edge* induit la manipulation de surfaces 2-variétés orientées, i.e. toutes les faces du maillage ont un sens de parcours de leurs sommets défini et opposé à celui de leurs faces adjacentes.

Plusieurs autres structures offrent une modélisation efficace des maillages surfaciques 2-variétés orientés. Les structures basées demi-arêtes sont souvent privilégiées. Celles-ci représentent les arêtes d'un maillage par deux demi-arêtes (voir figure 2.8). Chaque demi-arête conservant une référence sur sa face incidente, sur son sommet de départ et sur la demi-arête précédente et suivante autour de la face en question, elle stocke moins d'information qu'une *winged edge*. Il est possible de trouver plusieurs implémentations de ce type de structure. *Polyhedral Surface* [Ket99] et *Surface Mesh* de CGAL [SB11] ou encore OpenMesh [BBS+02] sont parmi les plus connues. Elles sont également accompagnées de

nombreux algorithmes de traitements géométriques et topologiques performants.

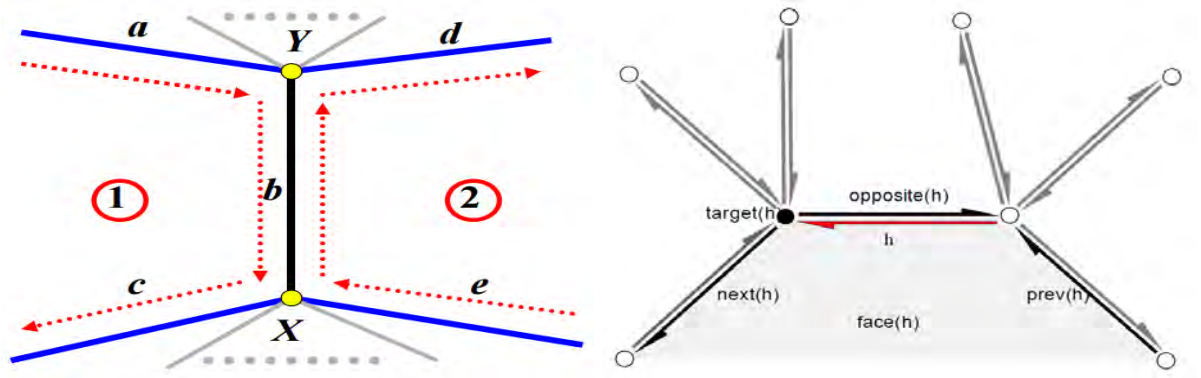


FIGURE 2.8 – À gauche, description d'une *winged edge*. À droite, description d'une demi-arête.

Les cartes combinatoires sont également un moyen de modéliser un maillage orienté. Leur particularité est qu'elles étendent le concept de demi-arêtes en nD. Ici, on parlera alors de *dart*, stockant une référence sur un élément incident de chaque dimension, de sorte que le modèle soit orienté. Ce principe est notamment utilisé par le Linear Cell Complex de CGAL [Dam16]. Plus largement, les cartes généralisées (ou G-cartes) possèdent les mêmes caractéristiques à la différence qu'elles ne sont pas orientées. Ces dernières permettent ainsi de modéliser tous les maillages, quelle que soit leur dimension, à l'exception des maillages non-variétés. Le modeleur Moka [3] utilise ces G-cartes et plusieurs opérations topologiques associées.

2.2.3 Structure pour maillages non-variétés

Si la majorité des structures de données sont dédiées aux maillages variétés, il existe quelques travaux significatifs sur le stockage et la manipulation de maillages non-variétés. En effet, certaines approches comme [LL01] s'intéressent à la gestion de maillages non-variétés. Dans ces travaux, Lee *et al.* cherchent à corriger la connectivité localement non-variété en introduisant des éléments intermédiaires pour effectuer la transition.

De Floriani *et al.*, dans [CDFW11, DFH05, DFMPS04], proposent de prendre en charge ces maillages complexes, en 3D ou en nD, à travers différentes stratégies. L'approche étant, pour les deux premiers, de traiter les éléments créant la non-variété de façon séparée, et dans le dernier, de mettre en place des mécanismes permettant un parcours générique et cohérent du maillage.

Q-Complex [ZLLY12] est une autre alternative. Cette structure hiérarchique utilise le concept de *dart* pour faire face à la difficulté de la non-variété. Ainsi, les configurations localement non-variétés sont traitées, que ce soit pour des éléments isolés (par exemple, des sommets isolés, des arêtes isolées) ou pour des incidences complexes (par exemple, sur des arêtes complexes).

Enfin, les travaux de Silva et Gomez [SG03] présentent une représentation basée sur les relations d'incidence, la structure AIF. Dans une telle structure, un élément de dimension n conserve une référence sur l'ensemble de ses éléments incidents, de dimension $n-1$

3. http://moka-modeller.sourceforge.net/index.php

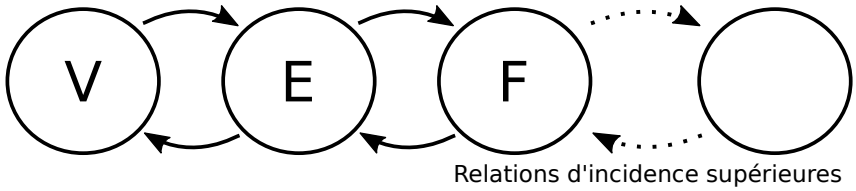


FIGURE 2.9 – Représentation de la structure AIF [SG03]. Ce schéma montre les relations entre les différents éléments géométriques du maillage (V pour les sommets, E pour les arêtes et F pour les faces).

et/ou $n+1$ (voir figure 2.9). Cela permet de manipuler un maillage de façon générique, qu'il soit 2-variété ou non-variété.

2.2.4 Structure hiérarchique

En marge de ce cet état de l'art, il existe également des structures de données hiérarchiques. Ces dernières permettent de stocker un maillage de base $\mathcal{M}^0$, une version simplifiée du maillage final $\mathcal{M}^n$, puis les différents éléments permettant de passer de $\mathcal{M}^0$ à $\mathcal{M}^n$ à volonté, dans un sens ou dans l'autre. Ces structures, notamment [DFMPS04], sont utilisées pour obtenir une version adaptative du maillage en fonction de certains critères. En visualisation, par exemple, la distance à la caméra joue un rôle dans la résolution du maillage à afficher [DFMMP00, P+96]. L'approche hiérarchique des maillages est également utilisée dans le domaine de la compression progressive [GTLH98](voir Section 2.5). Cela permet de produire différents niveaux de détails lors de la décompression. Pour plus d'information sur ce type de structures, il est possible de se reporter à l'état de l'art réalisé par De Floriani *et al.* [DFKP05].

2.2.5 Conclusion

	2-variété	Non-variété	Polygonal	Changements topologiques
SQuad [GLLR11a]	✓	✗	✗	✗
LR [GLLR11b]	✓	✗	✗	✗
Winged Edge [Bau72]	✓	✗	✓	✓
OpenMesh [BBS+02]	✓	✗	✓	✓
SurfaceMesh [SB11]	✓	✗	✓	✓
Polyhedral Surface [Ket99]	✓	✗	✓	✓
LCC [Dam16]	✓	✗	✓	✓
PES [LL01]	✓	✓	✓	✓
IA* [CDFW11]	✓	✓	✗	✓
NMIA [DFH05]	✓	✓	✗	✓
NMT [DFMPS04]	✓	✓	✗	✓
Q-Complex [ZLLY12]	✓	✓	✓	✓
AIF [SG03]	✓	✓	✓	✓

Tableau 2.1 – Tableau comparatif de différentes structures de données étudiées.

Dans le contexte de nos travaux, nous souhaitons mettre en œuvre un algorithme de compression progressive de maillages surfaciques. Une partie de notre approche consistant à simplifier le maillage (voir la Section 2.5), il est impératif de pouvoir réaliser des modifications topologiques sur sa connectivité. De plus, afin de permettre une gestion de l'ensemble des maillages surfaciques, 2-variétés ou non-variétés, il est impératif de choisir une structure adaptée. Cet ensemble comprend également les maillages polygonaux, dont les faces peuvent être de degré $n > 2$. Peu de structures possèdent les caractéristiques souhaitées (voir tableau 2.1). Nous avons choisi, pour nos travaux d'utiliser la structure AIF, de Silva et Gomez, pour sa simplicité d'implémentation.

2.3 Compression de données

Un ensemble de données peut être représenté comme une suite de symboles appartenant à un alphabet $\mathcal{A}$. Par exemple, une nombre est une suite de symboles, appelés chiffres, de l'alphabet $\mathcal{A} : \{i | 0 \leq i \leq 9\}$. La compression de données a pour but de diminuer la taille de cette suite en réduisant son entropie. Pour réaliser cette compression, il est obligatoire de procéder à une transformation réversible des données afin de pouvoir les décompresser après coup.

Nous parlerons ici de l'entropie de Shannon qui exprime la quantité d'information contenue dans un ensemble de données. Lorsque l'on parcourt une source d'information $\mathcal{SI}$, chaque symbole sym_i $(0 \leq i < Card(\mathcal{SI}))$ possède une probabilité d'apparition P_i directement liée à son nombre total d'occurrences. En informatique, où un symbole est représenté par un nombre fixe de bits, l'entropie H de $\mathcal{SI}$ représente le nombre bits minimum moyen qu'il est nécessaire d'utiliser pour un symbole. Elle est calculée de la façon suivante :

$$H(\mathcal{SI}) = - \sum_{i=0}^{Card(\mathcal{SI})-1} P_i \times log_2(P_i) \tag{2.7}$$

Plus l'entropie est faible, plus l'information est structurée et régulière. Au contraire, une entropie élevée décrira un ensemble de données chaotique. Il est généralement nécessaire de transformer des données pour s'approcher de cette limite théorique. Cette conversion s'appelle le codage. En utilisant des connaissances *a priori* sur les données ou non, le codage consiste à changer un symbole ou une combinaison de symboles de l'alphabet de départ en un symbole ou une combinaison de symboles de l'alphabet d'arrivée. Cette démarche est réalisée de telle sorte qu'il soit possible, avec ou sans informations supplémentaires, de restituer intégralement ou partiellement les données d'origine. Nous avons choisi de traiter, dans la partie suivante, uniquement le codage permettant la restitution intégrale des données, appelé également codage sans perte.

Le codage peut prendre plusieurs formes mais c'est lui qui permet de compresser les données. Il est possible de faire subir différents codages successifs à un ensemble de données.

2.3.1 Exemple de codage simple

Le codage se sert des connaissances acquises sur les données afin de les transformer au mieux. On prend souvent l'exemple des documents scannés. Dans ce cadre précis, nous représentons un document par des lignes de 64 pixels. Chaque pixel est soit blanc (B), soit noir (N). Un premier codage du document, pour sa numérisation, utilisera donc l'alphabet $\mathcal{A} : \{B, N\}$. Prenons l'exemple suivant du codage d'une ligne de pixels :

$$BBBBBBBBBBBBBBBBBBBBBBNNNNNNNNBBBBBBBBBBBBBBBBBBBBBBNNNNBBBBBBB$$

La structure même d'un document, s'il s'agit de texte, crée de longues séquences de symboles similaires (généralement B). Grâce à ce constat, il devient intéressant d'utiliser un codage par plages (ou *run-length encoding* en anglais). Au lieu de conserver chaque symbole, nous indiquons, pour chaque séquence de symboles similaires, sa longueur puis sa couleur (B ou N). Le nouveau code est le suivant :

$$22B8N22B4N8B$$

Le codage par plages a visiblement réduit l'information. De plus, si les symboles B et N sont codés sur 1 bit et les longueurs sur 6 bits ($2^6 = 64$), la taille du premier code est de 64 bits alors que celle du deuxième est de 28 bits. Nous avons donc effectivement réduit la taille de nos données. Toutefois, il faut noter que cette réduction n'est pas obligatoire et dépend essentiellement de la longueur des séquences de symboles similaires au sein d'une ligne.

2.3.2 Codage de Huffman

Le principe du codage de Huffman [H+52] est basé sur l'utilisation d'un arbre binaire. Il existe plusieurs variantes de ce codage. Nous nous intéresserons au codage semi-adaptatif et au codage adaptatif qui sont les plus performants. Le but est de compter le nombre d'occurrences de chaque symbole de la source de données. Les symboles deviendront les feuilles de l'arbre (voir figure 2.10). Chaque nœud possède un poids correspondant à la somme des poids de ses deux fils, ou le nombre d'occurrences du symbole correspondant dans le cas d'une feuille. On réunit ensuite, à chaque itération, les nœuds ayant le plus faible poids en un nouveau nœud. Le nœud final est appelé la racine. En adoptant un parcours de la racine jusqu'à une feuille donnée, on construit ainsi un code binaire associé au symbole correspondant. Pour cela il suffit de concaténer un 0 si l'on empreinte la branche de gauche et un 1 si l'on empreinte la branche de droite. Cela permet aux symboles avec une grande fréquence d'apparition d'être positionnés relativement haut dans l'arbre et donc d'avoir un code court. Une fois l'information $\mathcal{SI}$ codée, la taille moyenne d'un symbole $L(\mathcal{SI}')$, en bit/symbole, est bornée par l'entropie de $\mathcal{SI}$:

$$H(\mathcal{SI}) \leq L(\mathcal{SI}') \leq H(\mathcal{SI}) + 1 \tag{2.8}$$

Le codage semi-adaptatif calcule les occurrences de l'ensemble des données avant de construire l'arbre. Cela procure un gain en terme de bit/symbole car l'ensemble des probabilités d'apparition est connu au moment du codage. Cependant, il est nécessaire de transmettre la liste des symboles et leur nombre d'occurrences avec l'information codée pour permettre le décodage.

Le codage adaptatif, quant à lui, construit l'arbre binaire au fur et à mesure de la lecture des données. Les symboles sont alors codés à la volée en fonction de l'arbre courant. De cette façon, le décodeur n'a pas à recevoir d'informations complémentaires ou très peu.

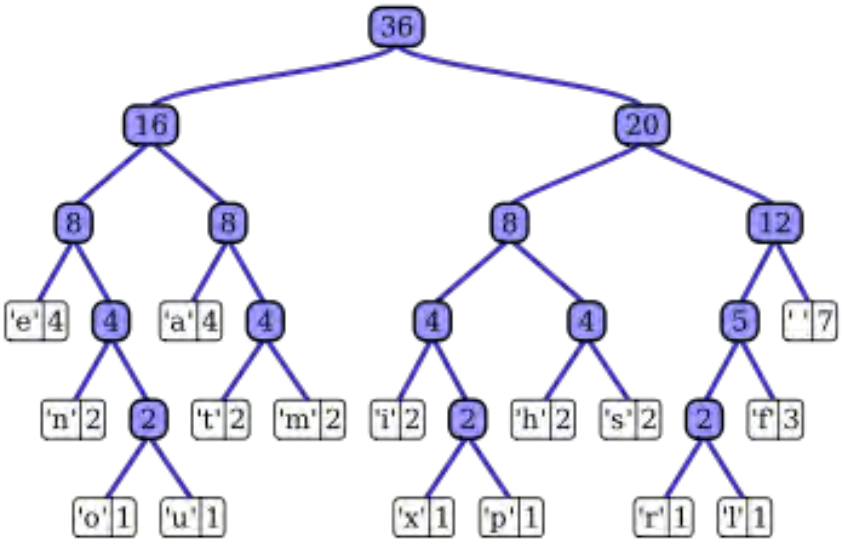


FIGURE 2.10 – Un exemple d'arbre de Huffman correspondant à la phrase « "this is an example of a huffman tree" ».

2.3.3 Codage arithmétique

Le codage arithmétique [WNC87, RL79] est un codage entropique à l'instar du codage de Huffman. Il reprend donc le principe d'exploiter la probabilité d'apparition des symboles. Le codage arithmétique représente une série de symboles par un nombre flottant. La construction de ce nombre commence, comme nous le montre la figure 2.11 par la définition d'un intervalle de départ, généralement $[0, 1[$. En connaissant la probabilité d'apparition des symboles d'un alphabet $\mathcal{A}$, pour chaque symbole sym_i $(0 \leq i < Card(\mathcal{A}))$, une borne inférieure Inf_i et une borne supérieure Sup_i sont déterminées telles que $Inf_i = \sum_{j=0}^{i-1} P_j$ et $Sup_i = Inf_i + P_i$, où P_k est la probabilité du symbole sym_k. Ensuite, durant la lecture des données, la borne inférieure Inf et la borne supérieure Sup de l'intervalle de départ sont progressivement mises à jour telles que $Inf = Inf + (Sup - Inf) \times Inf_i$ et $Sup = Inf + (Sup - Inf) \times Sup_i$, où Inf_i et Sup_i sont les bornes du symbole courant. Lorsque l'intervalle a convergé vers un nombre flottant déterminé, ce nombre est concaténé à l'information déjà codée et le processus redémarre à partir de l'intervalle de départ pour coder les symboles suivants. À la fin du codage, un symbole terminal est ajouté afin d'informer le décodeur. Pour le décodage d'un nombre flottant q, il faut connaître les bornes de chaque symbole. Pour chaque itération, il faut déterminer quel symbole sym_i borne q. sym_i est concaténé aux données décodées et q est mis à jour tel que $q = (q - Inf_i)/P_i$. Quand $q = 0$, le nombre a fini d'être décodé.

Contrairement au codage d'Huffman qui code un symbole avec un minimum de 1 bit, le codage arithmétique donne la possibilité de le représenter avec moins de 1 bit. Cela a pour avantage d'obtenir un nombre de bits moyen par symbole encore plus proche de l'entropie de Shannon.

Le codage arithmétique nécessite également de connaître, à la fois pour le codeur et le décodeur, l'ensemble des symboles et leur probabilité. Mais tout comme le codage de Huffman,

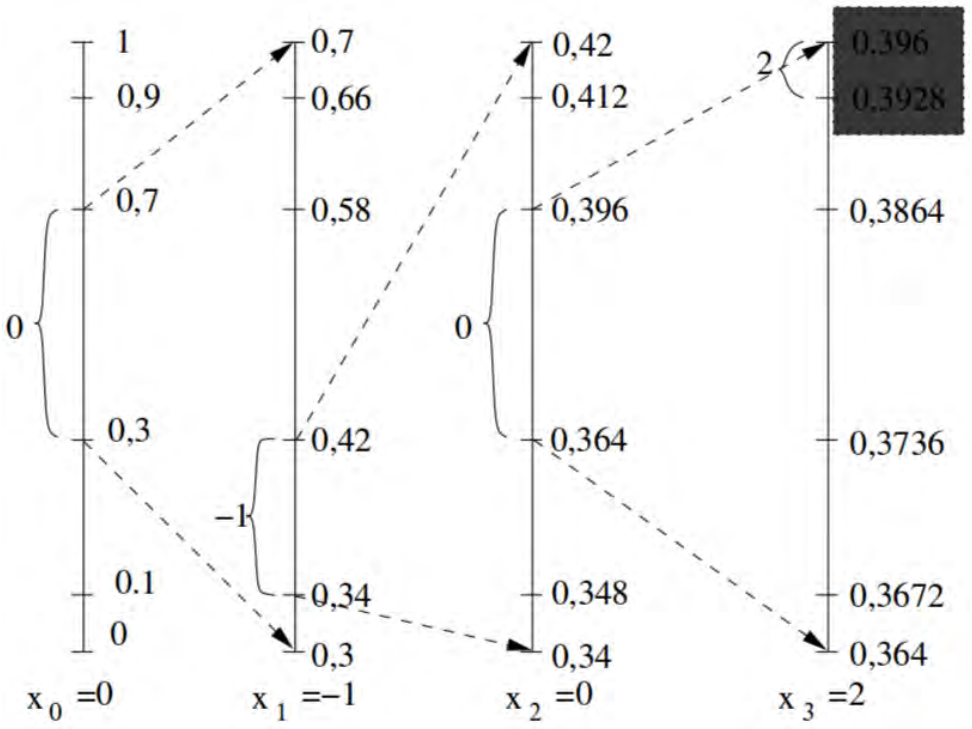


FIGURE 2.11 – Un exemple de codage arithmétique d'un signal d'alphabet $\mathcal{A} : \{-2, -1, 0, 1, 2\}$ avec les fréquences d'apparition respectives de 0.1, 0.2, 0.4, 0.2, 0.1 pour définir les intervalles. Symbole par symbole, le signal (0, -1, 0, 2) permet de construire le nombre flottant compris dans l'intervalle $[0.3928, 0.396]$, i.e. le nombre 0.3945 en binaire.

il est possible de n'utiliser que les informations déjà connues lors du parcours des données à coder.

2.3.4 Quantification

Afin de compresser des données, il est utile de connaître leur nature pour obtenir un résultat convaincant. Dans le cas de données 3D, l'information de géométrie peut, par exemple, être complexe à coder. En effet, les coordonnées des sommets d'un maillage sont représentées par des nombres flottants, avec une précision simple (32 bits) ou double (64 bits). Quand les premières décimales des ces coordonnées restent cohérentes d'un sommet à son voisinage, les dernières décimales sont totalement chaotiques. Si ces décimales ne sont pas nulles, l'entropie pour ce type de données sera donc assez élevée et la compression sera peu performante.

Pour contourner cette difficulté, la plupart des travaux de recherche sur le sujet utilisent une quantification uniforme. Ce traitement correspond à la conversion des coordonnées en entiers dans un intervalle $[0, 2^Q]$. Q est le nombre de bits utilisés pour représenter une coordonnée. Il exprime la précision voulue pour la géométrie du maillage. Cela implique une perte d'information car les dernières décimales des coordonnées originelles se trouvent tronquées. Néanmoins, sur un maillage normalisé, la quantification des coordonnées à 12 bits ($Q = 12$), qui est communément utilisée dans ce domaine, induit une erreur maximale, pour chaque coordonnée, de 1.2×10^{-4} unités de distance, ce qui revient à une erreur maximale de 2.1×10^{-4} unités de distance pour chaque sommet. Dans la grande majorité des applications cette erreur est négligeable. En contrepartie, le gain est substantiel car la position d'un sommet est maintenant de 3×12 bits, au lieu de 3×32 bits ou dans le pire des cas 3×64 bits, tout en diminuant l'entropie des données géomé-

triques. Comme justification supplémentaire, Michael Deering [Dee95] explique que l'exposant de 8 bits contenu dans un nombre flottant de 32 bits (IEEE) permet de décrire des distances allant de 15 milliards d'années lumière au rayon d'une particule subatomique. Pour maîtriser raisonnablement la précision, il décide d'utiliser une quantification à 16 bits par coordonnées. Son choix est alors fréquemment repris dans les travaux suivants. Toutefois, une quantification à 12 bits est aujourd'hui plus généralement utilisée.

Cette opération peut être réalisée sur d'autres attributs associés au maillages. Par exemple, les coordonnées de texture ou les normales liées aux sommets sont des informations de nature assez similaire aux données de géométrie. Elles sont donc généralement quantifiées de la même façon, avec une précision dépendant du contexte.

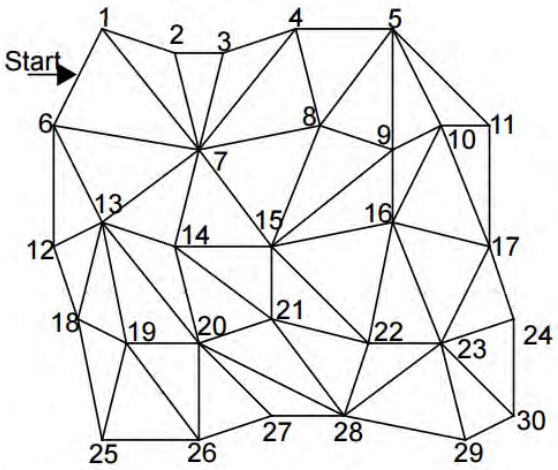
2.4 Compression mono-résolution de maillages 3D

La compression de maillages est un cas particulier de la compression de données. Il est nécessaire, à la fois, de coder la connectivité du maillage et ses différents attributs de manière efficace. Il a été prouvé par Tutte [Tut62], qu'un graphe planaire, représentant la connectivité d'un maillage triangulaire 2-variété, peut être codé avec un minimum de 3.24 bits par sommet. Cette borne théorique est ainsi la référence en ce qui concerne la compression de la connectivité du maillage.

Une première méthode de compression a été introduite par Deering dans [Dee95]. Cette approche propose de coder la connectivité en transformant l'information topologique en une série de triangles adjacents, cette bande de faces est appelée *triangle strip*. En ne conservant que la façon dont on passe d'un triangle à un autre, on réduit fortement l'entropie de la connectivité initiale. L'auteur ne définit alors que quelques codes et, en commençant par trois sommets formant un premier triangle, il explique comment il remplace un sommet par un nouveau pour trouver le prochain triangle (voir figure 2.12). Cette compression prend également en charge la géométrie, les couleurs et les normales liées aux sommets. Chacun de ces attributs sont préalablement quantifiés à une précision jugée suffisante. La position et la couleur d'un sommet étant des données locales, elles sont simplement codées en les remplaçant par la différence obtenue vis à vis du sommet précédent. Cette stratégie est estimée efficace sur des maillages uniformément échantillonnés car l'entropie de ces différences est faible contrairement à la donnée en elle-même. En définissant une sphère unité compartimentée en différents secteurs, les normales peuvent être codées par un indice unique, sur 17 bits, identifiant sa position exacte autour de la sphère.

Cette compression est appelée compression mono-résolution. Elle consiste à proposer un algorithme, généralement basé sur un parcours déterministe du maillage, pour coder les informations efficacement. Toutes ces données sont codées en un bloc. L'intégralité de ce bloc doit être lu et décompressé par le décodeur pour restituer le maillage dans sa globalité, au sens géométrique du terme. Plusieurs autres méthodes de compression mono-résolution ont vu le jour depuis, dans le but d'améliorer les performances ou d'élargir l'approche à d'autres catégories de maillages surfaciques. Mamou *et al.* [MZP09] présentent une optimisation de cette compression par *triangle strips*. Ils se basent sur une série de 10 configurations possibles pour diminuer l'entropie de la connectivité.

Certains travaux se sont plus particulièrement intéressés à la propriété de valence des sommets d'un maillage. Touma et Gotsman [TG98] cherchent ainsi, en partant d'un sommet donné, à concaténer sa valence correspondant au nombre de sommets adjacents. Ces sommets adjacents sont stockés dans une pile. Le processus continue en choisissant le prochain sommet de la pile. Un label est ajouté pour distinguer les trois situations suivantes du prochain sommet s (voir



Generalized Triangle Strip:
R6, O1, O7, O2, O3, M4, M8, O5, O9, O10, M11,
M17, M16, M9, O15, O8, O7, M14, O13, M6,
O12, M18, M19, M20, M14, O21, O15, O22, O16,
O23, O17, O24, M30, M29, M28, M22, O21, M20,
M27, O26, M19, O25, O18

Generalized Triangle Mesh:
R6p, O1, O7p, O2, O3, M4, M8p, O5, O9p, O10, M11,
M17p, M16p, M-3, O15p, O-5, O6, M14p, O13p, M-9,
O12, M18p, M19p, M20p, M-5, O21p, O-7, O22p, O-9,
O23, O-10, O-7, M30, M29, M28, M-1, O-2, M-3,
M27, O26, M-4, O25, O-5

Legend:
First letter: R = Restart, O = Replace Oldest, M = Replace Mi
Trailing "p" = push into mesh buffer
Number is vertex number, -number is mesh buffer reference
where -1 is most recent pushed vertex.

FIGURE 2.12 – Description de la compression par *triangle strips* de Mickael Deering [Dee95].

figure 2.13) : s est le dernier sommet traité d'un patch (complétude), le traitement de s permet de rencontrer une zone déjà traitée du maillage (séparation), tous les autres cas (ajout). Alliez et Desbrun [AD01b] améliorent cette compression par valence en minimisant les *séparations* par un meilleur choix du parcours. Ces travaux apportent également une meilleure gestion des bords du maillage. Cette approche permet de s'approcher très près de la borne définie par Tutte. La méthode a été étendue aux maillages polygonaux 2-variétés [KADS02, Ise02] en ajoutant à la valence des sommets le degré des faces. Un maillage constitué d'un seul type de polygone sera alors codé sans surcoût.

L'*EdgeBreaker* de Rossignac [Ros99] adopte un autre type de parcours pour compresser des maillages triangulaires 2-variétés. Il se sert d'un élément particulier : la porte. Une porte est le point d'entrée à une face par une de ses arêtes. En partant d'une porte donnée, la configuration du sommet opposé s_o offre cinq possibilités (voir figure 2.14). (1) Le patch de s_o est complet, un C est codé, on choisi alors la porte de droite ; (2) la porte de gauche ne mène à aucune face (inexistante ou déjà traitée), un L est codé, on choisi la porte de droite ; (3) la porte de droite ne mène à aucune face, un R est codé, on choisit la porte de gauche ; (4) Le patch de s_o n'est pas complet mais les deux autres portes mènent à une face, un S est codé, la porte de gauche et ajoutée à la pile, on choisi la porte de droite ; (5) les deux autres portes ne mènent à

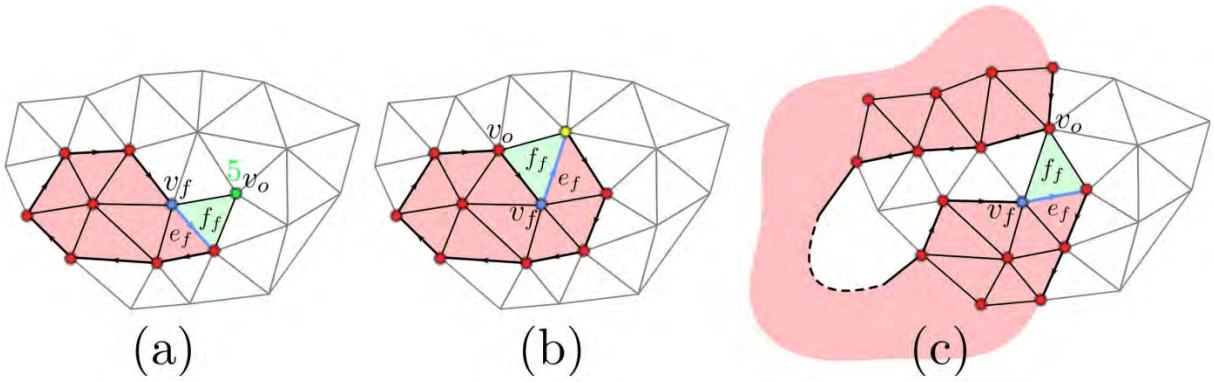


FIGURE 2.13 – Description des différentes configurations du codage par valence : (a) ajout, (b) complétion du patch et (c) rencontre d'une zone préalablement traitée.

aucune face, un E est codé, on choisi la dernière porte de la pile. Ce parcours permet de coder la connectivité avec seulement 4 bits par sommet. King *et al.* étendent l'*EdgeBreaker* aux maillages quadrangulaires. Ils atteignent 2.67 bits par sommet en profitant de l'avantage que procurent des faces de plus haut degré en réduisant *de facto* l'information de connectivité.

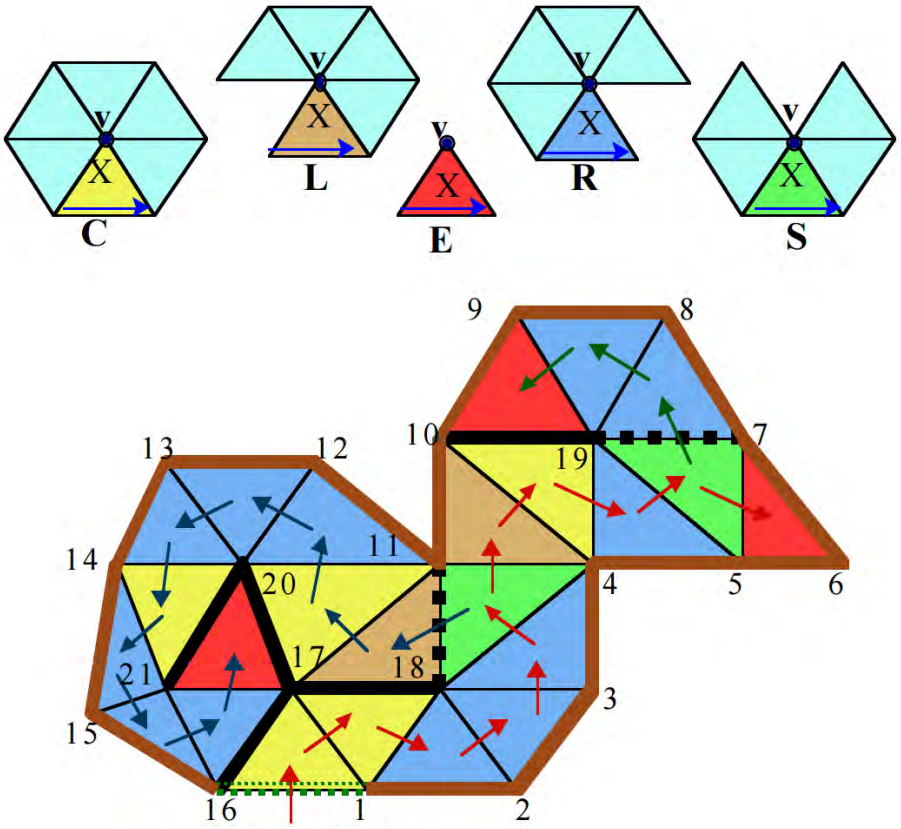


FIGURE 2.14 – Description du codage du *EdgeBreaker* de Rossignac [Ros99]. En haut, nous montrons les différentes configurations pour une porte donnée. L'algorithme est décrit sur un exemple de maillage triangulaire 2-variété, en bas.

Dans l'ensemble, les méthodes de compression mono-résolution permettent un codage quasi optimal d'un maillage. Les parcours utilisés offrent une compression et une décompression rapide des données car ils s'appuient sur des stratégies simples à mettre en œuvre. Malheureusement, la compression mono-résolution ne permet pas au décodeur d'obtenir un maillage global avant la fin

de la transmission et de la décompression des données. Dans un certain nombre d'applications, notamment la visualisation d'objets 3D sur le Web, cela entraîne un manque de fluidité dès lors que la décompression ne s'effectue pas en temps réel.

2.5 Compression multi-résolution de maillages 3D

Si la compression mono-résolution est efficace pour réduire la taille des données 3D, elle peut être mal adaptée à certains cas d'utilisation. En effet, dans le domaine de la visualisation de maillages, il peut être désagréable, voir complètement bloquant en fonction de la quantité de données à transmettre, de devoir attendre l'intégralité de la décompression pour interagir avec l'objet en question.

En 1996, Hoppe [Hop96] définit une nouvelle façon de voir la compression de maillages. Le principe de base est de proposer une décompression progressive des données, de telle sorte qu'il soit possible de visualiser immédiatement une version simplifiée de l'objet, dès le début de la réception des données. Puis, au fur et à mesure de la décompression, des versions (appelées niveaux de détails ou LoDs) de plus en plus détaillées sont reconstruites, jusqu'à restituer le maillage initial (voir figure 2.15). Afin d'obtenir ce résultat, il est nécessaire, à la compression, de simplifier peu à peu le maillage. À chaque opération de simplification, les informations permettant de revenir en arrière sont conservées. Cela permet, en partant du LoD le moins détaillé $\mathcal{M}^0$, de retrouver itérativement le maillage initial $\mathcal{M}^n$. Les informations de raffinement sont alors codées afin que le décodeur puisse également reconstruire le maillage à l'aide de $\mathcal{M}^0$, fourni dès le début de la transmission. $\mathcal{M}^0$ ne contenant que très peu d'informations, il est donc possible, avec une très faible quantité de bits, de visualiser l'objet 3D sous sa forme simplifiée.

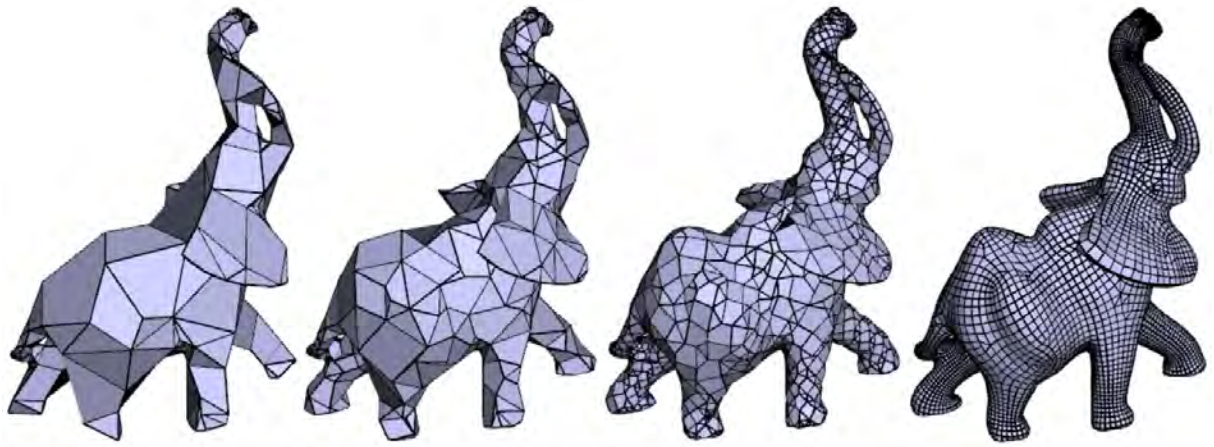


FIGURE 2.15 – Décompression progressive du modèle Elephant par la méthode de Maglo *et al.* [MCAH12]. La figure montre plusieurs LoD du maillage du moins raffiné, à gauche, au plus raffiné, à droite.

La compression progressive est particulièrement adaptée dans le contexte Web défini dans la Section 1. Dans cette situation, la rapidité et la fluidité de la transmission sont essentielles. Cette compression augmente significativement la qualité de l'expérience utilisateur, d'autant plus lorsqu'il s'agit de quantités importantes de données 3D. La progressivité de l'approche offre également une adaptation transparente aux périphériques de réception car il est possible, en fonction des caractéristiques matérielles, de stopper la décompression à tout moment, sans porter atteinte à la visualisation de l'objet.

Afin de comprendre clairement les méthodes existantes de compression progressive de maillages, nous commençons par présenter un état de l'art sur la simplification de maillages surfaciques (voir Section 2.5.1). Ensuite, nous évoquons les principales méthodes de compression progressive proposées de 1996 à ce jour, en fonction des types de maillages qu'elles traitent : 2-variétés triangulaires (Section 2.5.2), 2-variétés polygonaux (Section 2.5.3), non-variétés triangulaires (Section 2.5.4) et non-variétés polygonaux (Section 2.5.5).

2.5.1 Simplification d'objets 3D

Il existe un certain nombre d'opérateurs de simplification dans l'état de l'art. Néanmoins, dans le cadre de la compression progressive, la simplification doit être facilement réversible pour permettre au décodeur de reconstruire le maillage durant la décompression. L'étude de Heckbert et Garland [HG97] classifie les méthodes de simplification de maillages surfaciques en quatre grandes catégories : la décimation de sommets, la décimation d'arêtes ou l'agrégation de sommets, la décimation de faces et enfin la décimation de collections de faces adjacentes.

La décimation de sommets consiste à supprimer itérativement des sommets en remaillant l'ensemble de leurs faces incidentes (voir figure 2.16). Différents travaux comme ceux de [SZL92, SL96] se sont penchés sur cette approche. Le remaillage du patch en question peut suivre diverses stratégies. En pratique, l'algorithme de Delaunay contraint est couramment employé. Pour inverser la simplification d'un patch, il est nécessaire d'insérer un nouveau sommet au sein du patch, généralement au barycentre. La position du nouveau sommet est alors corrigée pour correspondre à la position originelle. Dans le cas d'un maillage triangulaire, la nouvelle connectivité est évidente, cependant, pour un maillage polygonal, il est nécessaire de fournir des informations complémentaires afin de retrouver la configuration initiale du patch. Malheureusement, cette technique de simplification n'est applicable que sur des maillages 2-variétés, de part sa nature.

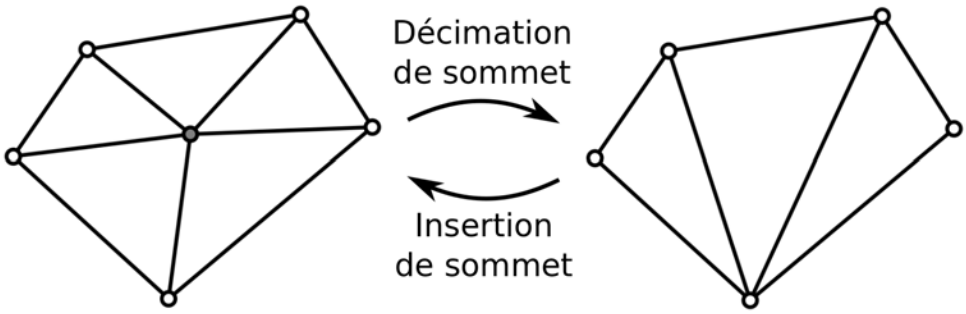


FIGURE 2.16 – Description de l'opérateur de décimation de sommets et son dual, l'opérateur d'insertion de sommets.

La catégorie de simplification suivante est décomposée en deux parties, la décimation d'arête (ou contraction) et l'agrégation de sommets (ou fusion), la deuxième étant la généralisation de la première (voir figure 2.17). La contraction d'arête est largement utilisée [HDD+93, Gue95, AS96, RR96] dans le domaine de la simplification de maillages surfaciques. Son avantage est qu'elle peut être utilisée aussi bien sur des surfaces 2-variétés que non-variétés (comme nous le montrons dans la Section 3.4), qu'il s'agisse d'une triangulation ou d'un ensemble de polygones. Une contraction consiste en une fusion des sommets d'une arête donnée et la suppression des faces incidentes à l'arête en question. L'agrégation de sommets suit le même principe mais ne nécessite pas forcement que deux sommets à fusionner soit adjacents. Elle est notamment appliquée dans [RB93, GH97] pour la simplification de maillages non-variétés. Les opérations duales sont respectivement l'expansion et la duplication de sommet. Encore une fois des informations

géométriques et topologiques complémentaires sont obligatoires afin d'inverser la simplification. Par exemple, dans le cas d'une configuration triangulaire, localement 2-variété sans bord, autour d'une arête a à contracter, il sera nécessaire de connaître les deux arêtes pivot et de conserver cette information. Une arête pivot est l'arête résultant de la fusion des deux arêtes d'une face incidente à a lors de sa suppression.

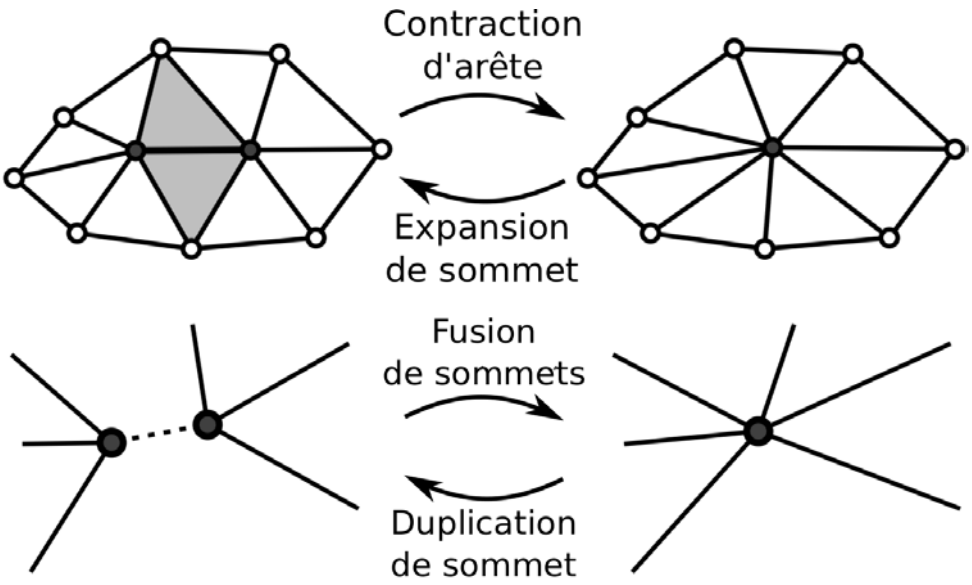


FIGURE 2.17 – En haut, description de l'opérateur de contraction d'arêtes et son dual, l'expansion de sommets. En bas, description de l'opérateur de fusion de sommets et son dual, la duplication de sommets.

La simplification de maillages par décimation de faces reste assez marginale dans la littérature. Cette méthode de simplification est présentée, entre autres, dans [Ham94, WHTS01] exclusivement pour des maillages 2-variétés triangulaires. Il s'agit de supprimer la face ainsi que toutes ces arêtes incidentes puis de fusionner l'ensemble de ses sommets incidents (voir figure 2.18). Il faut également fournir un certain nombre d'informations, comme la position initiale des sommets, pour pouvoir revenir en arrière.

Enfin, la décimation de collection de faces adjacentes (ou groupement de faces) a été étudiée afin d'offrir une plus grosse granularité à la simplification de maillages. Le principe reste le même que la décimation de faces. Le fait d'appliquer l'opération de décimation sur un ensemble de faces augmente la rapidité de la simplification. Toutefois, la qualité visuelle de la simplification peut être moins bonne que pour les méthodes précédentes, surtout sur des objets complexes.

Quelle que soit la technique de simplification, elle est généralement guidée par une métrique permettant à l'algorithme de minimiser l'erreur engendrée. La construction d'une file de priorité sur les éléments à décimer en fonction de leur poids respectifs est un schéma couramment utilisé. La nature de ces poids est variée. La courbure, la qualité géométrique des faces, ou la saillance des arêtes sont des exemples de mesures utilisées dans le calcul de ces poids. Le choix du poids, en fonction de la méthode de simplification et des caractéristiques du maillage traité, affecte très largement la distorsion engendrée, c'est à dire la différence entre le maillage initial et le maillage simplifié. Il est impératif de minimiser cette distorsion pour conserver la meilleure régularité topologique et/ou qualité visuelle pour ce maillage.

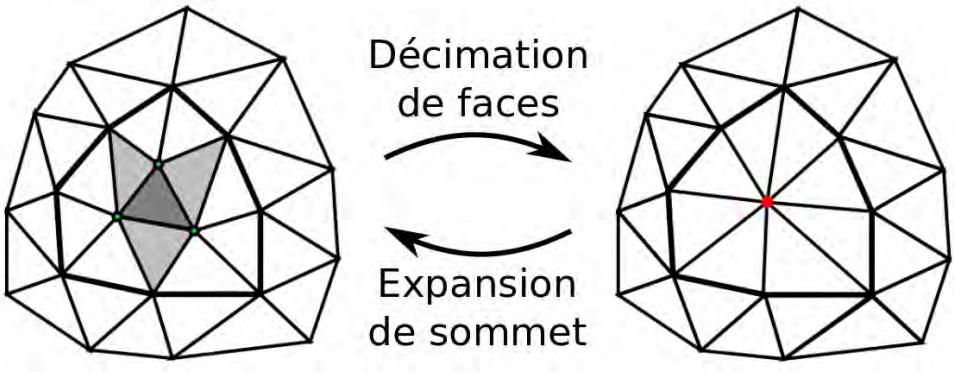


FIGURE 2.18 – Description de l'opérateur de décimation de faces et son dual, l'expansion de sommets en faces.

2.5.2 Compression progressive de maillages 2-variétés triangulaires

Les premiers travaux de Hoppe sur le sujet [Hop96] présentent une méthode permettant la compression de maillages triangulaires 2-variétés. Il utilise une file de priorité sur les arêtes dont le poids est une énergie tenant compte, à la fois, de facteurs géométriques mais aussi des valeurs de possibles attributs liées aux sommets. La minimisation de cette énergie tend, dans ce cas, à conserver les saillances, limiter la distorsion et optimiser l'aspect des faces en contraignant la longueur des arêtes par la simulation d'un système à ressort. À chaque itération, Hoppe applique l'opérateur de contraction pour décimer la première arête de la file de priorité. La file est mise à jour à chaque nouvelle décimation. Cela permet de contrôler localement l'erreur induite par la simplification. Cela transforme peu à peu le maillage $\mathcal{M}^n$ initial en $\mathcal{M}^0$ sa version la plus simplifiée. Au delà de la simplification, Hoppe donne la possibilité de retrouver le maillage initial en conservant plusieurs informations, I^i (i allant de n à 1), à chaque contraction d'arête. La première partie est constituée, pour une arête s_0s_1, des indices de trois sommets : s_{res} résultant de la contraction, s_l et s_r les sommets pivot, c'est à dire que $s_{res}s_l$ et $s_{res}s_r$ sont les arêtes pivot de la contraction. Ensuite, il stocke les attributs de s_0 et de s_1. Il exploite la cohérence locale de l'opération de simplification pour coder ces données. Les sommets pivot sont alors décrits sur 5 bits en partant du principe qu'un sommet a, en moyenne six sommets adjacents. Les attributs de s_0 et s_1 sont codés relativement aux nouveaux attributs de s_{res}. Dans ce cadre, $\mathcal{M}^n$ peut alors être exprimé par $\mathcal{M}^0$ suivi des informations I^1 à I^n.

La méthode de Hoppe [Hop96] devait spécifier explicitement l'indice des sommets résultants afin de savoir sur quel élément l'expansion devait avoir lieu. Alliez et Desbrun [AD01a] préfèrent s'appuyer sur un parcours déterministe du maillage pour définir implicitement un ordre de raffinement à la décompression. En parcourant le maillage de face en face, cette méthode permet de simplifier peu à peu la géométrie grâce à un opérateur de décimation de sommets (voir figure 2.19). De la même manière que dans [AD01b], Alliez et Desbrun utilisent la valence des sommets pour coder chacune des simplifications. Après chaque phase (ou vague) de décimation, les auteurs proposent une étape de nettoyage consistant à régulariser la connectivité du maillage, et ainsi d'homogénéiser les valences par sommet. Cette phase est également encodée pour rester en accord avec le décodeur. Cette approche propose également un codage plus efficace de l'information géométrique. En effet, les vecteurs de déplacements de sommets sont représentés au sein d'un repère de Frenet, construit localement en fonction de la surface, avant leur compression. Cette manipulation permet de réduire significativement l'entropie des vecteurs en question.

Plusieurs autres méthodes utilisent la décimation par vagues. [TGHL98] réalise une simplification de patch. Dans ce cas particulier, un patch est défini comme un "polygone simple" (i.e.

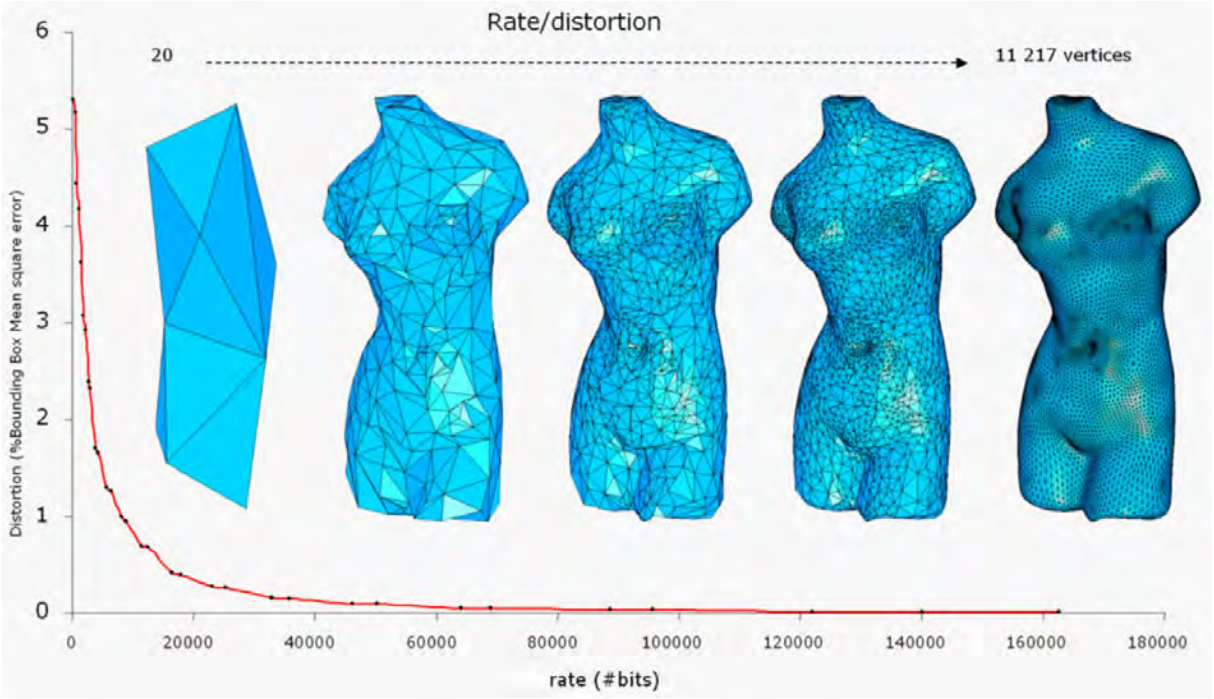


FIGURE 2.19 – Décompression du modèle *Venus* (de 20 à 11,362 sommets) par la méthode [AD01a] et sa courbe de débit-distorsion.

une triangulation sans arête intérieure). Les informations à encoder sont alors les arêtes supprimées de ce patch et les déplacements de sommets associés. [PR00] présente une simplification progressive du maillage à l'aide de contractions d'arêtes. Afin de remédier aux limitations de [Hop96], ces contractions sont ordonnées le long d'un parcours déterministe représenté par un arbre couvrant. En plus d'un codage de la connectivité standard, la méthode propose une prédiction géométrique basée sur la position des sommets voisins afin d'encoder plus efficacement les déplacements des sommets lors de la décompression. Outre le choix de la structure d'arbre couvrant pour l'ordonnancement des contractions, il est également possible de s'appuyer sur une séquence Hamiltonienne d'arête, comme nous le montre [KBG02]. Pour leur part, Cohen *et al.* [COLR99] utilisent une coloration du maillage pour déterminer les différents patchs du maillage et leur ordre. Ces patchs sont ensuite simplifiés par la suppression de leur sommet central.

Les méthodes de compression progressive présentées jusqu'à maintenant encodaient le maillage pendant sa simplification (comme nous l'avons défini au début de cette section). Cela équivaut à une approche *fine to coarse*, où l'on part du maillage le plus détaillé pour arriver à la version la plus grossière. En réalité, il existe également l'approche inverse ou *coarse to fine*. Ainsi [VCP09] propose un algorithme de compression par reconstruction. Cette méthode débute à partir d'un maillage $\mathcal{M}^0$ (la version la plus grossière), obtenue par une simplification initiale. En utilisant le même algorithme de reconstruction à la compression comme à la décompression (le codeur écrit les données et le décodeur les lit), le modèle original est restitué. Cette algorithme de reconstruction ne garantit pas de retrouver la même connectivité. Il est donc nécessaire de coder une série de bascules d'arêtes (*edge flip* en anglais).

Dans cette catégorie de méthodes *coarse to fine*, on retrouve également les travaux de Karni et Gotsman [KG00]. Il s'agit ici d'une compression spectrale de la géométrie de maillages 2-variétés triangulaires. Les auteurs transforment la matrice d'adjacence de sommets en une base orthogonale, grâce à un opérateur Laplacien. La projection des coordonnées 3D sur cette base forme le spectre géométrique du maillage. Ces coefficients peuvent enfin être transmis progres-

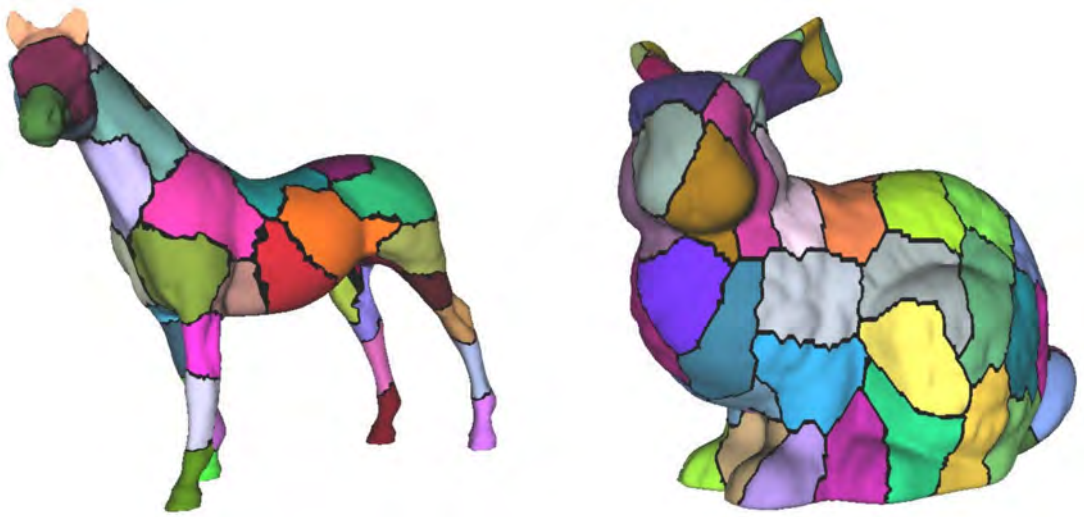


FIGURE 2.20 – Partitionnement des modèles *Horse* et *Bunny* par [KG00] pour assurer la rapidité et la stabilité de la méthode de compression spectrale.

sivement en envoyant les bits significatifs en premiers. Malheureusement, à cause de limitations dues à la complexité et à la stabilité de la méthode, il est impossible de définir ce spectre sur le maillage entier. La méthode nécessite donc un partitionnement du maillage en plusieurs parties (voir figure 2.20).

2.5.3 Compression progressive de maillages 2-variétés polygonaux

Comme nous l'avons vu, un certain nombre de travaux ont d'ores et déjà été menés sur la compression progressive d'objets 3D. Néanmoins, pour le moment, ces avancées se sont toutes limitées au traitement de maillages 2-variétés triangulaires. Ces surfaces ont l'avantage de posséder des contraintes topologiques fortes qu'il est aisé d'exploiter afin d'obtenir un encodage efficace et ainsi parvenir à de remarquables taux de compression.

La méthode proposée par Maglo *et al.* [MCAH12] permet cependant d'étendre ces algorithmes pour permettre la compression de maillages 2-variétés polygonaux. Cette approche simplifie également la surface par vagues successives. Or, elle utilise un opérateur de simplification de patch, adapté aux faces polygonales, pour décimer le maillage. Ce type de simplification supprime le sommet central du patch et ajoute des arêtes pour remailler le trou créé. Ce sont ces informations que Maglo *et al.* encodent par des booléens. Le parcours mis en place pour ordonner les décimations, et ainsi assurer une harmonie entre le codeur et le décodeur, est similaire à celui employé dans [AD01a]. La méthode se sert également d'un repère de Frenet dans le but d'améliorer le taux de compression relatif à la géométrie.

À notre connaissance, [MCAH12] est la seule approche ciblant les maillages 2-variétés polygonaux. Néanmoins, il existe quelques rares méthodes permettant la compression progressive de maillages non-variétés, comme détaillé ci-dessous.

2.5.4 Compression progressive de maillages non-variétés triangulaires

Popović et Hoppe [PH97] sont les premiers à proposer une méthode de compression progressive destinée aux maillages non-variétés triangulaires. Celle-ci est une extension de [Hop96],

avec ses avantages et ses défauts. En effet, ils ont généralisé leurs opérateurs de simplification et de raffinement aux configurations non-variétés triangulaires. L'encodage de la connectivité a également été modifié pour inclure les nouvelles possibilités. Pour éviter de pénaliser le taux de compression avec des codes nettement plus nombreux que dans [Hop96], Popović et Hoppe s'appuient sur des contraintes topologiques évidentes afin de réduire la redondance de l'encodage. De plus, ils introduisent une stratégie de propagation de la valeur d'aire initiale des faces durant la simplification. Cette technique permet d'approximer les sommets esseulés par des sphères et les arêtes esseulées ou pendantes par des cylindres. Les dimensions de ces sphères et de ces cylindres sont définies par les valeurs d'aires propagées jusqu'aux sommets et arêtes en question. La qualité visuelle de maillage complexe est ainsi significativement augmentée, notamment pour les LoDs les plus grossiers (voir figure 2.21).

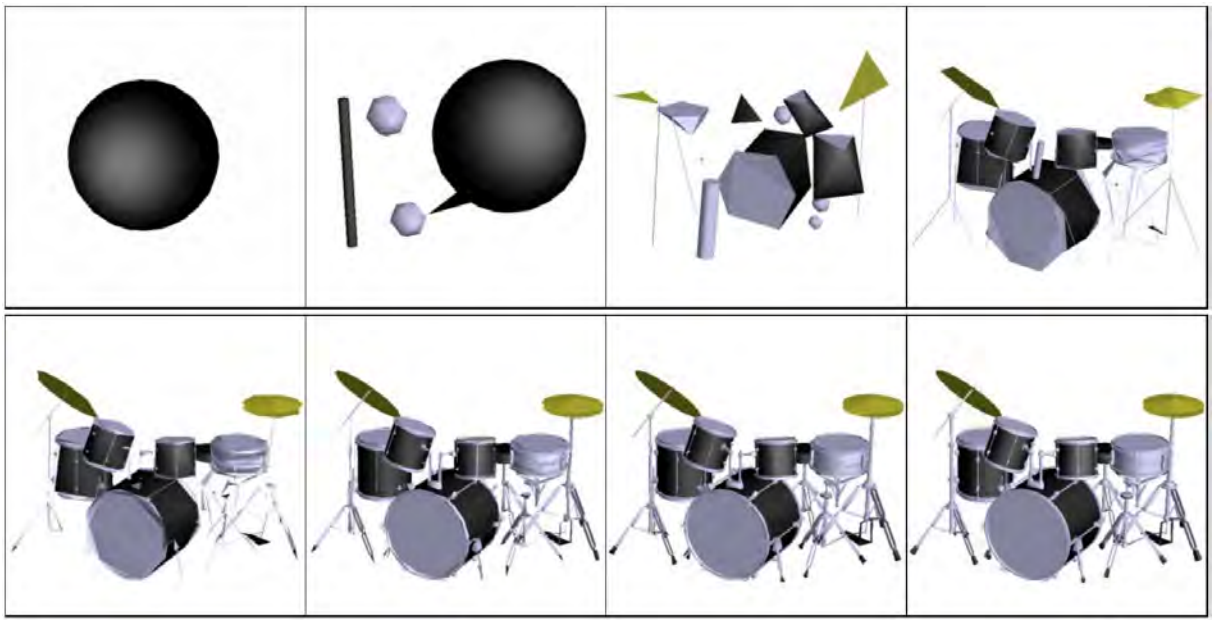


FIGURE 2.21 – Exemple d'une décompression opérée par [PH97] sur le modèle *Drumset*. Les sommets isolés et les arêtes isolées ou pendantes sont représentés visuellement par des sphères et des cylindres, respectivement, pour améliorer la qualité visuelle des premiers LoDs.

La plupart des méthodes de compression progressive, comme [PH97], se penche principalement sur l'encodage de la connectivité, afin de le rendre le plus efficace possible. La géométrie du maillage est alors encodée assez simplement, même si plusieurs approches prédictives réduisent son impact. Pourtant, Gandoin et Devillers [DG00] estiment que l'information de géométrie représente, en taille, la plus grosse partie des données décrivant un maillage. Partant de ce constat, ils vont proposer un tout nouveau type de compression progressive : la compression basée géométrie. Avec [DG00], puis [GD02], Gandoin et Devillers se servent d'un octree, construit sur le maillage, avec une approche *coarse to fine* pour encoder efficacement sa géométrie. En partant de la boite englobante du maillage (i.e. un octree à une cellule représentant le premier niveau), l'algorithme va itérativement découper toutes les cellules du niveau courant. Pour chaque cellule découpée, les auteurs encodent le nombre de sommets présents dans chaque cellule fille. Cela permet au décodeur de construire un maillage avec le centre de chaque cellule non-vide du niveau courant, durant la décompression. Afin de maintenir la connectivité, il est nécessaire de spécifier au décodeur la relation entre les différentes cellules. Ainsi, si un des sommets d'une cellule est adjacent à un des sommets d'une autre, les deux cellules seront reliées par une arête. Les opérations topologiques sur cette connectivité relative aux cellules et son encodage sont réalisés de la même manière que dans [PH97]. Tous ces éléments font de [GD02] une méthode de compression très efficace en terme de taux de compression. Néanmoins, cette approche ne

possède pas de contrôle local de la distorsion et donc l'utilisation d'un octree entraîne un fort effet de quantification visible à bas et moyen débit.

Plusieurs autres travaux ont repris la méthode de [DG00] afin de l'améliorer. Tian *et al.* [TJL+12] proposent une optimisation du codage de l'octree, ainsi qu'une métrique locale permettant de diminuer la distorsion. Peng et Kuo [PK05] ajoutent à cela une prédiction basée géométrie dans le but de réduire l'entropie de la connectivité. Malgré tout, même si ces méthodes limitent l'effet de quantification, il est encore présent.

Peng *et al.* [PHK+10] vont alors décider de se séparer de l'octree tout en gardant la même philosophie de compression. Il s'agit toujours d'une approche *coarse to fine*, mais les cellules sont remplacées par des groupements de sommets. À chaque niveau, le *Generalized Lloyd Algorithm* (GLA) couplé à une métrique géométrique décompose le groupement de sommets courant en un maximum de K groupements fils, K étant fixé par l'utilisateur. Un sommet représentant est choisi au sein d'un groupement pour qu'il soit à la fois proche de la moyenne des sommets et soumis localement à une assez forte courbure. Ce choix permet de respecter les caractéristiques géométriques du maillage, comme les saillances. Lors de la décomposition d'un groupement, les auteurs codent le déplacement des sommets représentant des groupements fils par rapport à celui du groupement père. Le changement de connectivité engendré par cette décomposition est également codé. Le choix des groupements à décomposer est guidé par une prédiction, basée sur la géométrie du maillage, connue du codeur et du décodeur. L'affranchissement de l'octree et le soin apporté au contrôle de la distorsion permettent à Peng *et al.* de proposer une méthode de compression efficace, offrant de bons résultats en terme de débit-distorsion (voir figure 2.22). Cependant, il est important de noter que cette approche ne garantit pas la restitution sans perte du maillage à la fin de la décompression.

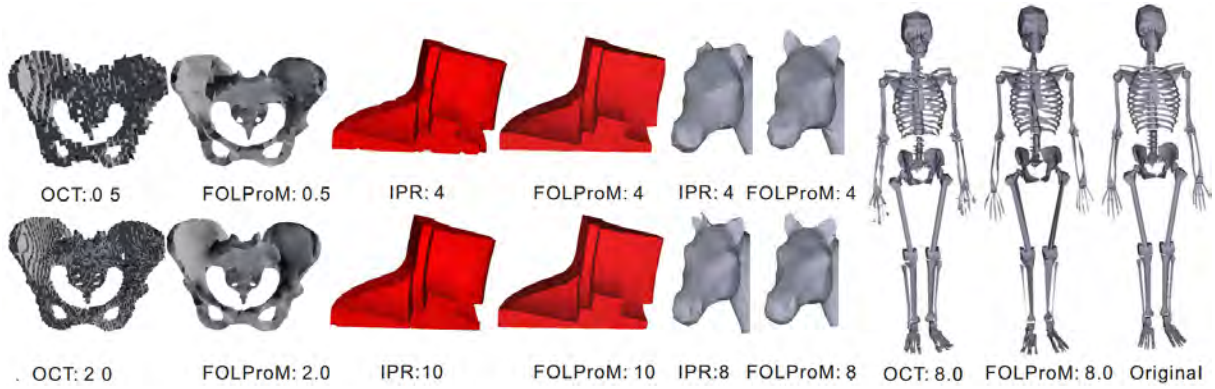


FIGURE 2.22 – Différents LoDs des modèles *Donna*, *Fandisk*, *Horse* et *Skeleton*, de gauche à droite. Les modèles *Donna* et *Skeleton* sont non-variétés. IPR, OCT et FOLProM correspondent respectivement aux méthodes présentées dans [VCP09], [PK05] et [PHK+10]. Les auteurs fournissent également le nombre de bits par sommet utilisés pour la reconstruction de chaque LoD.

2.5.5 Compression progressive de maillages non-variétés polygonaux

Comme le montrent les sections précédentes, il existe un certain nombre de méthodes dédiées à la compression progressive de maillages surfaciques. Néanmoins, très peu permettent de traiter des maillages triangulaires non-variétés et, à notre connaissance, aucune ne gère la compression de maillage polygonaux non-variétés. Guéziec *et al.* [GBTS99] proposent pourtant une manière d'outrepasser cette limitation. Leur approche consiste à dupliquer les éléments non-variétés d'un maillage pour obtenir un ensemble de sous-maillages 2-variétés. Ces sous-maillages sont ensuite compressés à l'aide de [TGHL98] et les informations nécessaires pour retrouver, à la fin de la

décompression, le maillage non-variété sont ajoutées. Malheureusement, aucun exemple clair de compression de maillages polygonaux n'est donné.

2.6 Compression de l'attribut de texture

Si la compression de maillages 3D a été traitée assez largement, les différents travaux du domaine se sont principalement focalisés sur la compression de l'information de connectivité et de géométrie. Il est vrai que ces données représentent l'information minimum à transmettre pour être en mesure de restituer un modèle 3D. Néanmoins, pour un grand nombre d'applications, cela ne permet pas d'obtenir une qualité visuelle satisfaisante. En effet, pour être réaliste, un maillage est souvent enrichi de différents types d'attributs. Il peut s'agir, par exemple, de couleurs, de textures ou encore de normales pour détailler d'avantage la matière ou la géométrie de la surface. Comme nous l'avons vu dans la Section 2.1, ces données peuvent être attachées de plusieurs façons. Isenburg et Snoeyink [IS01] proposent une stratégie d'encodage des attributs attachés par *corner* à un maillage 2-variété. L'utilisation de *corners* rend possible un gestion simple des coutures présentes dans la paramétrisation. En se servant des notions de *smooth corner* (*corner* similaire au précédent dans l'ordre de rotation autour d'un sommet) et *crease corner* (*corner* différent du précédent dans l'ordre de rotation) et de quelques règles sur leur disposition, dont certaines prédictives, ils proposent un encodage efficace permettant de traiter n'importe quel type d'attribut durant la compression mono-résolution d'un maillage.

Les travaux que nous évoquons dans la suite de cette section se penchent plus particulièrement sur la gestion de la texture. Nous rappelons que les textures retranscrivent généralement une information de couleur, mais elles peuvent également être utilisées pour exprimer des normales, des déplacements de positions, etc. C'est donc un attribut polyvalent.

2.6.1 Compression de la paramétrisation

Isenburg et Snoeyink constatent dans [IS03] que les rares méthodes de compression prenant en compte l'attribut de texture encodent les coordonnées UV à l'aide de la même technique de prédiction que pour les coordonnées des sommets du maillage. Selon eux, cette approche n'est pas correcte car les coutures éventuelles rendent ces prédictions totalement erronées. Ils présentent donc dans cet article un opérateur de prédiction, certes moins efficace, mais cohérent et adapté à ce type de données pour la compression mono-résolution de maillages 2-variétés. Pour la détermination des *smooth corners* et des *crease corners*, ils appliquent la méthode exposée dans [IS01]. Puis une prédiction dite *parallélogramme* est utilisée pour optimiser le codage des coordonnées UV.

De leur côté, Váša et Brunnett tentent également d'améliorer le codage de l'attribut de texture dans le cadre d'une compression mono-résolution de maillages 2-variétés. Pour ce faire, ils proposent successivement deux méthodes. D'abord, dans [VB13], les auteurs proposent un opérateur de prédiction basée sur un parallélogramme pondéré. En fonction des données déjà décodées, ils construisent un quadrilatère $(ss_r s_o s_l)$ basé sur une face triangulaire $(ss_r s_l)$ déjà décodée de la paramétrisation, le sommet s_o étant la prédiction. Les angles encore inconnus du quadrilatère sont estimés en fonction du degré des sommets s_r et s_l. La compression de la paramétrisation en elle-même est réalisée par l'algorithme *Edgebreaker* [Ros99].

Dans un second temps, Váša et Brunnett [VB14] mettent en place une amélioration de leur prédiction en tirant avantage de la géométrie du maillage. Deux éléments permettent cette avan-

cée. Lors de la décompression, la connectivité et la géométrie sont entièrement décodées avant les données relatives à la texture. De plus, les auteurs mettent en lumière la similarité existant entre un triangle du maillage et le triangle correspondant au sein de la paramétrisation. L'article explique que cette similarité est d'autant plus grande si la paramétrisation a été réalisée automatiquement ou semi-automatiquement. Une paramétrisation manuelle (par des designers) engendre une similarité, en moyenne, plus faible. Ces deux éléments permettent de construire une prédiction des coordonnées UV en fonction de la position des sommets du maillage. Váša et Brunnett proposent alors plusieurs méthodes de prédiction allant dans ce sens (voir figure 2.23). Il s'agit d'une amélioration de la prédiction basée sur un parallélogramme pondéré (la pondération dépend alors également de la géométrie du maillage) et de plusieurs variantes utilisant un opérateur Laplacien.

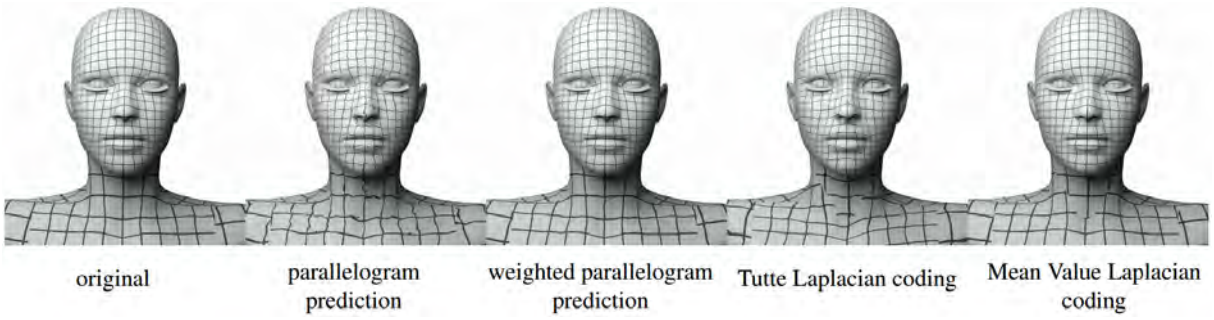


FIGURE 2.23 – Comparaison visuelle entre le modèle *Victoria* original, à gauche, et les versions reconstruites avec un débit de 5.25 bps par les différentes méthodes proposées dans [VB14].

En marge de ces méthodes, Sander *et al.* [SSGH01] décident d'améliorer la qualité visuelle des LoDs générés par [Hop96]. Pour cela, ils choisissent d'optimiser la paramétrisation initiale de la texture, à la compression, afin qu'elle soit mieux adaptée aux versions successives simplifiées. Une métrique d'étirement de la texture permet de modifier la paramétrisation en conséquence. Un découpage et un nouvel agencement de la texture est également proposé. Cette méthode limite ainsi les distorsions liées à la texture durant la simplification du maillage.

2.6.2 Compression progressive de l'image de texture

La question de la compression de la paramétrisation de la texture a été abordée. Cependant, il est nécessaire de joindre aux informations du maillage l'image de texture pour que les différents LoDs puissent être affichés dans leur intégralité, durant la décompression. Or la taille de ces images est non-négligeable et souvent même plus importante que celle des données compressées relatives au maillage. Si cela ne pose aucun problème dans le cadre d'une compression mono-résolution, cela pénalise grandement les méthodes de compression progressive. En effet, afin de pouvoir afficher, à la décompression, les premiers LoDs texturés, l'image doit être transmise en amont. Or cela nuit au rapport débit-distorsion.

Plusieurs méthodes, comme [TA04, YSK04, YLK04], proposent d'appliquer la progressivité à l'image de texture. Ces approches commencent par compresser progressivement la texture en utilisant, par exemple, le format JPEG 2000. L'image est ainsi décomposée en plusieurs niveaux de détails $\mathcal{T}^0 ... \mathcal{T}^m$, où $m+1$ est le nombre de niveaux. $\mathcal{T}^0$ est le premier LoD (le plus grossier) de la texture. En concaténant $\mathcal{T}^0$ et $\mathcal{T}^1$, on obtient le deuxième LoD et ainsi de suite jusqu'au $m+1^{\text{ème}}$. Ensuite, les $\mathcal{T}^i$ sont multiplexés avec les différentes vagues d'informations relatives au maillage. Pour guider ce multiplexage, toutes ces méthodes utilisent une métrique, généralement basée image, pour obtenir le meilleur rapport débit-distorsion tout au long de la décompression.

Le multiplexage donne alors un réel caractère progressif à la compression de maillages texturés. Toutefois, les métriques employées peuvent ne pas être les plus adaptées pour mesurer la qualité visuelle de modèles 3D.

2.7 Conclusion

À la vue de l'état de l'art, nous constatons que le domaine de la compression progressive de maillages 2-variétés a déjà été largement étudié. Cependant, aucune approche ne permet encore de traiter l'ensemble des maillages surfaciques, i.e. polygonaux non-variétés, de manière générique. Ce manque nous a conduit à proposer une méthode de compression progressive sans perte adaptée à ces maillages. Grâce aux travaux précédemment réalisés, nous avons également acquis la conviction de l'importance d'un contrôle local de la décimation pour une nette amélioration de la qualité visuelle. Cette conviction nous a guidée dans notre objectif d'obtenir d'excellents ratios débit-distorsion lors de la décompression, notamment à bas et moyen débit.

À la différence de la majorité des méthodes existantes, nous prenons en compte l'attribut de texture potentiellement lié au maillage. Notre gestion de ces données par *corner* permet la prise en charge des coutures, fréquemment présentes au sein de la paramétrisation. Pour finir, afin de disposer d'une méthode de compression entièrement progressive, nous avons proposé un mécanisme pour multiplexer les informations relatives à l'image de texture et celles relatives au maillage. Ce multiplexage est guidé par une métrique perceptuelle afin d'optimiser le rapport débit-distorsion, durant toute la décompression.

Chapitre 3

Compression progressive générique

3.1 Résumé

L'objectif de nos recherches est de mettre en œuvre une méthode de compression progressive adaptée aux maillages surfaciques non-variétés polygonaux. Cette classe d'objets 3D représente l'ensemble des maillages surfaciques. La compression que nous présentons se situe donc comme une approche générique permettant de traiter, de la même manière, tout maillage surfacique quelle que soit sa connectivité. De plus, les avantages apportés par la compression progressive (détaillée dans la Section 2.5) permettent de lever un certain nombre de verrous comme l'interactivité des transmissions ainsi que l'adaptation aux périphériques de réception et aux données traitées. Ces atouts font de notre méthode une alternative intéressante pour le stockage, la transmission et surtout la visualisation interactive d'objets 3D dans un contexte Web.

Comme expliqué dans l'état de l'art (Section 2.2), la compression progressive de maillages nécessite l'utilisation de structures de données adaptées. Afin que les algorithmes en jeu soient efficaces, l'accès aux éléments du maillage, mais également la modification topologique de ce dernier doivent être réalisables en temps raisonnable, voir en temps interactif ou réel en ce qui concerne la partie décompression de la méthode. Nous avons donc choisi de nous appuyer sur la structure AIF [SG03] pour stocker et manipuler nos données 3D (voir Section 3.2).

Notre méthode de compression simplifie le maillage traité par vagues successives à l'aide d'un opérateur de contraction d'arête (voir Section 3.4). Cela veut dire que, à chaque itération, k contractions sont réalisées sur le maillage, supprimant ainsi k sommets. Chaque itération ou

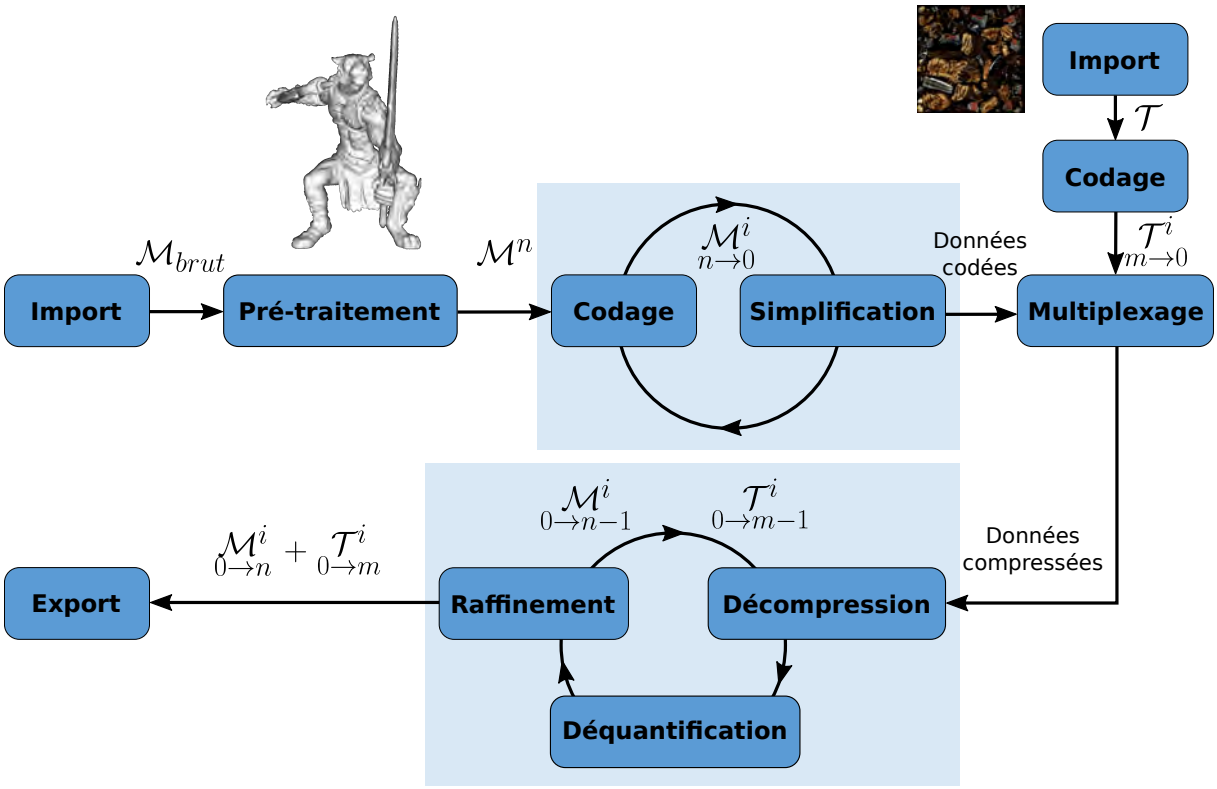


FIGURE 3.1 – Pipeline de notre méthode de compression. Notre approche commence par charger le maillage dans la structure AIF. Après avoir pré-traité le maillage, celui-ci est progressivement simplifié et encodé. En parallèle, l'image de texture est compressée progressivement. Nous procédons alors au multiplexage des paquets de données relatifs au maillage et à la texture. Durant la décompression, le décodeur détermine si le paquet courant raffine le maillage ou la texture. Dans les deux cas, l'utilisation de ces données permet d'obtenir un LoD de meilleure qualité visuelle. L'opération se répète pour tendre vers le maillage original.

vague de décimation produit un nouveau LoD transformant le maillage initial $\mathcal{M}^n$ en $\mathcal{M}^{n-1}$ puis $\mathcal{M}^{n-2}$ jusqu'à $\mathcal{M}^0$ (voir Section 3.6).

La décimation des arêtes est entièrement guidée par une métrique locale (voir Section 3.5). Cette métrique permet d'attribuer un poids pour chaque arête du maillage et donc de construire une file de priorité sur ses éléments. Durant une vague de décimation, les arêtes sont contractées dans l'ordre de la file. Cette dernière est mise à jour à chaque simplification afin d'optimiser la gestion de la distorsion induite. Une vague est terminée lorsque le poids de l'arête courante, au sein de la file de priorité, dépasse un certain seuil, établi dynamiquement. La décimation se finit lorsqu'il ne reste plus d'arête dans la file de priorité.

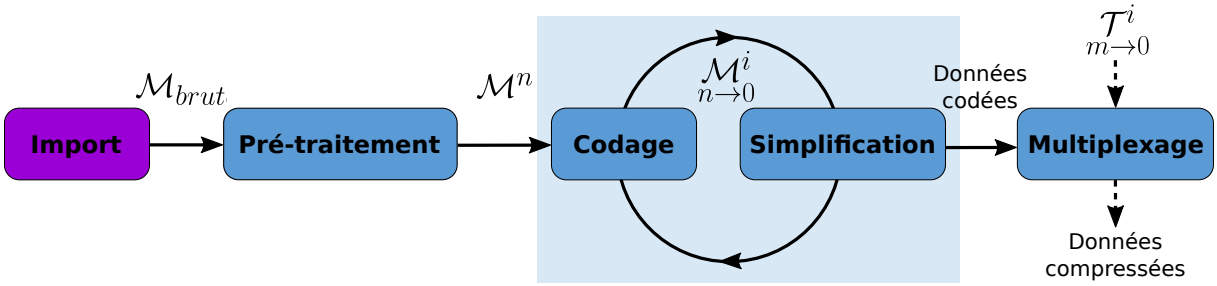
Afin de coder efficacement le maillage et permettre au décodeur de restituer le maillage initial, nous utilisons certaines stratégies destinées à réduire l'entropie des données 3D traitées (voir Section 3.7). Durant la simplification, avant chaque contraction, les informations nécessaires à la restitution du voisinage de l'arête simplifiée sont récupérées puis codées. Cela implique des données de connectivité et de géométrie mais également les attributs de texture associés. À la fin de la vague, un parcours déterministe est construit sur le maillage sous la forme d'un arbre couvrant. Cela permet de regrouper les informations de contraction de façon à ce que le décodeur retrouve implicitement les sommets à expandre. Pendant la construction de l'arbre, la position des sommets résultant d'une contraction lors de la vague courante est codée. Cela détermine un ordre implicite des informations de contraction à transmettre. Cet ordre est connu à la fois du

codeur et du décodeur.

Une fois la simplification totale du maillage effectuée, l'ensemble des informations est alors comprimé par un codeur arithmétique. Cela donne un flux de données précédé par une entête et le codage de $\mathcal{M}^0$ (composé uniquement de sommets isolés). Ce flux peut ainsi être transmis et décodé afin de reconstruire peu à peu, à partir de $\mathcal{M}^0$, les LoD $\mathcal{M}^1$ puis $\mathcal{M}^2$ jusqu'à $\mathcal{M}^n$ (voir Section 3.8).

Dans le cas de maillages texturés, la paramétrisation de la texture est codée au même moment que la géométrie, elle est donc transmise de façon progressive. En ce qui concerne la texture en elle-même, nous souhaitons éviter de la transmettre dans son intégralité au début du flux. Afin d'optimiser la qualité visuelle des LoDs pour un débit donné, nous compressons progressivement la texture grâce au format JPEG 2000. Ces données sont ensuite multiplexées aux informations compressées relatives au maillages (voir Section 3.8). Le multiplexage est guidé par une métrique de distance perceptuelle basée image afin de conserver le meilleur rapport débit-distorsion durant toute la décompression.

3.2 Structure de données choisie



Comme indiqué dans la Section 2.2, nous avons choisi d'utiliser la structure de données AIF pour manipuler le plus facilement possible des données 3D surfaciques. La première étape de notre méthode est donc, logiquement, de charger les données 3D à comprimer dans la structure AIF.

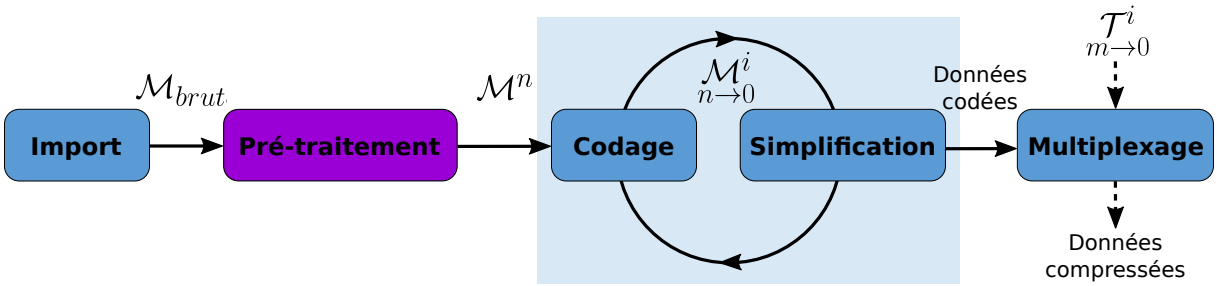
Pour obtenir une structure réellement adaptée à notre approche, nous avons procédé à quelques modifications de la structure de base. En premier lieu, afin de pouvoir effectuer efficacement des traitements topologiques sur le maillage, il a été nécessaire de mettre en place un système de cache concernant certaines relations d'incidence et d'adjacence non natives. Par exemple, AIF permet d'accéder à l'ensemble des arêtes incidentes à une face de manière directe. Cela n'est pas le cas pour l'ensemble des sommets incidents à une face. Lorsqu'une telle requête est formulée, cet ensemble est calculé en s'appuyant sur les relations d'incidence déjà connues. L'ensemble est alors conservé, permettant d'accélérer les requêtes similaires suivantes. Tout changement topologique affectant cet ensemble notifie automatiquement à la face son invalidation. À la prochaine requête, l'ensemble sera recalculé. Ce système augmente significativement les performances des algorithmes de simplification et de raffinement réalisés sur le maillage, dans le cadre de notre méthode.

Dans un second temps, il a fallu prendre en charge l'attribut de texture potentiellement lié au maillage. Alors que certaines propriétés peuvent être attachées directement aux sommets, arêtes ou faces d'un maillage (par exemple, la couleur d'une face), la gestion de la texture nécessite une modélisation plus complexe. Nous avons donc ajouté, à la structure AIF, une liste d'éléments

corners, discutés dans la Section 2.1. Ici, un *corner* est une paire (sommet, face) à laquelle on associe des coordonnées 2D décrivant un point sur la texture associée.

Il est important de rappeler que la suite de notre méthode de compression n'est pas dépendante d'une structure de données particulière tant que cette dernière est capable de gérer naturellement tous les cas de non-variétés. Le choix d'AIF a été motivé par sa simplicité, sa flexibilité et sa généricité en dimension (dans l'éventualité d'une extension de la méthode aux maillages volumiques).

3.3 Pré-conditions et pré-traitements



Si la méthode de compression progressive permet de traiter tous les maillages surfaciques, il existe cependant certaines pré-conditions qu'ils doivent valider afin que le processus se déroule correctement (voir Section 3.3.2). De plus, des pré-traitements, obligatoires ou optionnels, sont effectués sur le maillage pour optimiser le codage (voir Section 3.3.1 et 3.3.3). Dans la suite de ces travaux, nous considérons que le maillage initial $\mathcal{M}^n$, en entrée de notre méthode, est valide de notre point de vue.

3.3.1 Quantification et déquantification

Dans la Section 2.3.4, il est expliqué que la quantification des attributs scalaires (par exemple, la géométrie, les coordonnées de texture) est appliquée afin d'obtenir un codage efficace de ces valeurs, sans être pénalisé par le bruit souvent inutile de leurs dernières décimales. Nous choisissons donc, après le chargement du maillage dans la structure AIF, de quantifier les coordonnées de position des sommets à Q_G bits et, s'il existe une texture associée, les coordonnées de texture à Q_T bits. La quantification $\tilde{v}$ d'une valeur $v \in \mathcal{V} : \{v_0, ..., v_n\}$ est réalisée comme suit :

$$\tilde{v} = arrondi(\frac{v - min(\mathcal{V})}{max(\mathcal{V}) - min(\mathcal{V})} \times (2^Q - 1)) \tag{3.1}$$

Q est le nombre de bits sur lequel est exprimé $\tilde{v}$. $min(\mathcal{V})$ et $max(\mathcal{V})$ sont respectivement la valeur minimale et la valeur maximale de l'ensemble $\mathcal{V}$. L'arrondi utilisé est un arrondi au plus proche entier vers les infinis (pour les valeurs intermédiaires comme -5.5 et 12.5, respectivement en -6 et 13). Afin de quantifier de manière homogène les valeurs données, un arrondi au plus proche pair (ou impair) serait plus judicieux. Cependant, en pratique, la distribution des valeurs quantifiées ne change pas, ou très peu.

La déquantification des valeurs ne permet évidemment pas d'inverser l'arrondi. Cela engendre donc une erreur e_i pour chaque valeur v_i. Cette erreur est tout de même bornée en fonction de

Q :

$$0 \leq e_i \leq 0.5 \times \frac{max(\mathcal{V}) - min(\mathcal{V})}{2^Q - 1} \tag{3.2}$$

Notre méthode procède à la déquantification des valeurs dès leur décodage, afin d'obtenir la position des sommets et les coordonnées de texture dans leurs espaces initiaux respectifs. Cette opération transforme $\tilde{v}$ en v' de la façon suivante :

$$v' = \frac{\tilde{v}(max(\mathcal{V}) - min(\mathcal{V}))}{2^Q - 1} + min(\mathcal{V}) \tag{3.3}$$

À la compression, Q est fixé par l'utilisateur pour chaque type de données et $max(\mathcal{V})$ ainsi que $min(\mathcal{V})$ sont déterminés en fonction des données. Durant la décompression, le décodeur n'a aucun moyen de connaître ces informations. Q_G et G_T sont donc écrits sur 5 bits (de 0 à 32) dans l'entête du flux de données compressées. Ces valeurs sont suivies de la coordonnée minimale puis maximale des sommets du maillage. Il n'est pas nécessaire de spécifier les bornes de toutes les dimensions car nous effectuons une quantification uniforme sur les trois axes afin de ne pas perturber les prédictions géométriques décrites dans la Section 3.7.1. Les bornes de la paramétrisation de texture ne sont pas spécifiées car les coordonnées sont comprises entre 0 et 1 et qu'il est rare que l'on puisse réduire significativement cet intervalle.

3.3.2 Intégrité géométrique

Le deuxième pré-traitement obligatoire effectué sur le maillage avant sa compression sert à valider son intégrité topologique pour le bon fonctionnement de la méthode par la suite. Les critères d'intégrité ne sont pas respectés lorsque deux sommets du maillage partagent exactement la même position, i.e. ils ont les mêmes coordonnées. Cette situation arrive généralement à la suite de deux types de manipulations. La première est le repli manuel ou automatique d'un bord de la surface sur un autre, sans les fusionner, lors de la création ou l'édition du maillage. Un exemple simpliste serait la modification, par un designer, d'un plan triangulé afin d'obtenir un cylindre ouvert. La deuxième manipulation envisagée est la quantification des positions réalisée en début de pipeline. En effet, en fonction de la valeur de Q choisie, deux sommets relativement proches peuvent se retrouver exactement aux mêmes coordonnées entières. Une telle unicité est nécessaire pour garantir un ordre naturel dans l'espace 3D entre les différents éléments du maillage. Les coordonnées de texture, si elles existent, ne sont pas soumises à cette validation car aucun de nos algorithmes ne nécessite une hypothèse forte d'unicité concernant ces valeurs, contrairement à ceux concernant les positions des sommets.

La superposition de deux sommets peut également entraîner la superposition de deux arêtes ou même de deux faces. Nous assumons que ces configurations sont géométriquement non valides. Afin de rendre au maillage son intégrité géométrique, la position de tous les sommets est vérifiée et les cas de superposition sont détectés. Chaque groupe de sommets géométriquement identiques sont fusionnés par un opérateur d'agrégation de sommets très similaire à l'opérateur de contraction générique d'arêtes présenté en Section 3.4. À la fin de ce pré-traitement, le nombre de sommets du maillage courant est légèrement inférieur ou égal au nombre de sommets du maillage précédemment chargé dans la structure AIF. Cela certifie l'unicité des positions de sommets dont nous tirons parti plus loin dans la Section 3.7.1. Une amélioration possible serait de définir une valeur Q, utilisée pour la quantification, de telle sorte que cela n'engendre pas de fusion de sommets supplémentaire à celles potentiellement exigées par le maillage initial.

La méthode de compression proposée ne permet pas, pour le moment, d'inverser les fusions opérées durant cette phase. La connectivité peut alors, dans certaines situations, être légèrement modifiée. Cependant, la qualité visuelle du maillage n'est pas impactée. En effet, si des sommets étaient initialement à la même position, leur fusion n'aura pas modifié la qualité visuelle. De plus, il est possible de modifier la valeur de Q utilisée de telle sorte que cela n'engendre pas de fusions dues à la quantification.

3.3.3 Fusion des composantes connexes

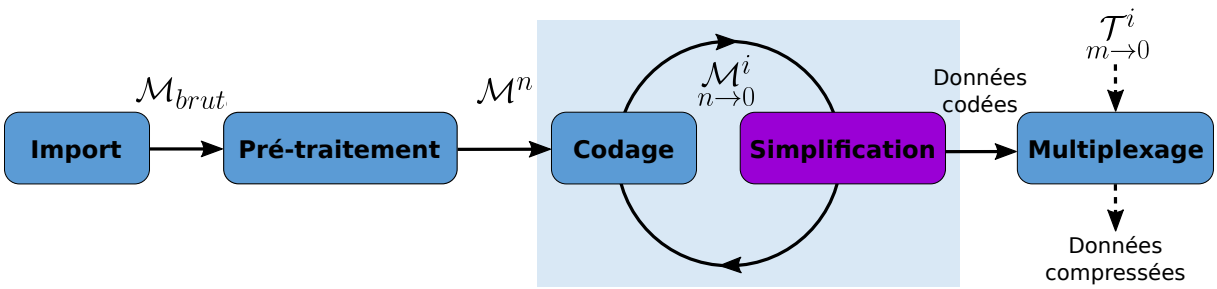
Un maillage peut être constitué d'une ou plusieurs composantes connexes. Le terme composante connexe définit une sous-partie $\mathcal{M}^-$ d'un maillage $\mathcal{M}$ pour laquelle, pour tout sommet s_i et $s_j \in \mathcal{M}^-$ $(s_i \neq s_j)$, il existe une suite d'arêtes adjacentes allant de s_i à s_j.

Notre méthode traite correctement les maillages comportant plusieurs composantes connexes. Toutefois, il arrive de rencontrer des assemblages de plusieurs centaines de composantes. Comme expliqué dans la Section 3.7.1, dans notre cas, un nombre élevé de composantes peut pénaliser la compression. Pour remédier à cela, nous avons décidé de proposer un traitement optionnel reliant les différentes composantes par une arête isolée afin de produire un maillage constitué d'une seule composante connexe.

L'idée n'étant pas de relier chaque partie de façon naïve, nous utilisons la répartition spatiale des composantes pour guider les liens. Après avoir détecté toutes les composantes connexes, nous calculons leurs barycentres qui seront leurs points respectifs de référence. Nous apparions les barycentres les plus proches jusqu'à ce que chacun ait été sélectionné au moins une fois. Cela certifie que toutes les composantes seront bien connectées à un tout. Ensuite, pour chaque paire de barycentres b_i et b_j, représentant respectivement les composantes $\mathcal{M}_i^-$ et $\mathcal{M}_j^-$, nous lions les sommets $s_k \in \mathcal{M}_i^-$ et $s_l \in \mathcal{M}_j^-$ par une nouvelle arête $s_k s_l$. s_k et s_l sont choisis tels que la longueur $|s_k s_l|$ soit minimale.

Ce traitement améliore significativement le taux de compression. Parfois la qualité visuelle du maillage est également augmentée car, en se regroupant, les composantes connexes offrent une visualisation plus cohérente. Cependant cet effet est difficilement contrôlable et dépend de la métrique utilisée pour la décimation des arêtes (voir Section 3.5). Cette modification topologique est aisément réversible. Les arêtes à supprimer sont définies explicitement en fonction du numéro de la contraction qui l'a décimée à la compression.

3.4 Opérateurs de simplification



Dans le cadre de notre méthode de compression progressive, il est nécessaire de simplifier itérativement le maillage $\mathcal{M}^n$. Pour cette décimation, il est important de choisir un opérateur

de simplification adapté à nos exigences. Nous nous sommes appliqués à minimiser au mieux la distorsion engendrée par cette simplification, même sur les maillages les plus complexes. Pour cela, il était primordial d'avoir un contrôle fort sur l'erreur engendrée. Nous avons donc décidé de nous appuyer sur un opérateur possédant une granularité fine. La manipulation de surfaces non-variétés polygonales a également été une contrainte incontournable. En effet, des opérateurs comme le remaillage de patch ne permettent pas, à eux seuls, de simplifier de tels maillages. Nous avons ainsi choisi assez naturellement, la contraction d'arête, pour la simplification, et l'expansion de sommet (voir Section 3.9), pour le raffinement. Leur généralisation, à savoir l'agrégation et la duplication, ne semble pas nécessaire. Ces derniers permettent, entre autres, de fusionner naturellement deux composantes connexes, cependant, l'information d'adjacence entre deux sommets est une chose supplémentaire à coder. Ce surcoût ne paraît pas justifié. En outre, notre mécanisme de fusion de composantes connexes, décrit dans la Section 3.3.3, permet déjà, si besoin est, de simuler ce comportement.

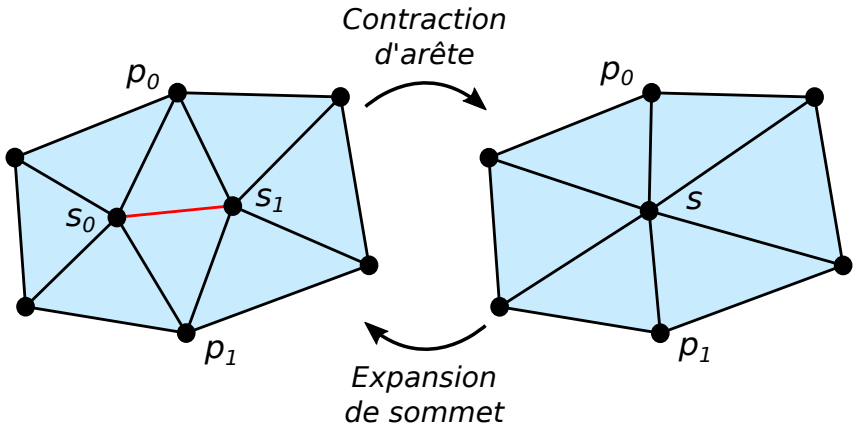


FIGURE 3.2 – Schéma représentant la contraction d'une arête s_0s_1 sur une configuration 2-variété sans bord, transformant le patch de gauche vers le patch droit. L'opération inverse, l'expansion du sommet s_0 est également décrite. Celle-ci s'appuie sur les deux sommets pivots p_0 et p_1.

La contraction d'arête sur un patch localement 2-variété est une opération bien maîtrisée à ce jour (voir figure 3.2). Hoppe l'a introduite dans [Hop96] pour sa méthode de compression progressive. Il explique que ce seul opérateur permet de simplifier entièrement un maillage surfacique. Cette contraction (comme l'expansion de sommet) est généralisée aux configurations localement non-variétés triangulaires dans [PH97]. Il est alors possible de contracter une arête esseulée ou complexe. Cependant, nous avons besoin d'opérateurs traitant des patchs non-variétés polygonaux. Nous avons donc développé nos propres opérateurs génériques de simplification et de raffinement en partant des travaux de Popović et Hoppe.

Notre algorithme de contraction d'une arête s_0s_1 consiste en plusieurs phases (voir figure 3.3). Premièrement, nous supprimons l'ensemble des faces triangulaires incidentes à s_0s_1 et les relations d'incidence et d'adjacence qui les lient aux éléments voisins (figure 3.3-b). À ce stade, les *corners* associés à ces faces sont également supprimés.

Ensuite, les relations d'incidence de s_0 et de s_1 sont reportées sur le nouveau sommet s, en incluant les *corners*. Cette dernière opération occasionne généralement des doublons d'arêtes et de faces. Une phase de nettoyage est alors obligatoire pour supprimer les éléments en question et corriger les relations d'incidence et d'adjacence impliquées (figure 3.3-c). Cette opération ne peut pas supprimer d'élément déjà volontairement dupliqué car la validité du maillage a été vérifiée lors du pré-traitement (voir Section 3.3.2). s_0s_1 est alors supprimée ainsi que ses relations avec les éléments voisins, notamment ses faces incidentes non triangulaires. Ces faces perdent alors un degré. De façon générique, toutes les faces incidentes à s_0s_1 perdent un degré, seulement, les

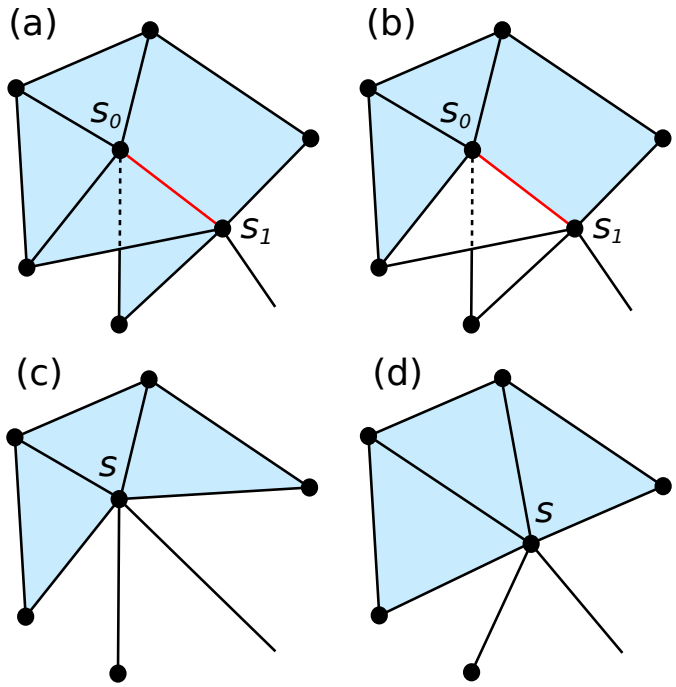


FIGURE 3.3 – Schéma représentant la contraction d'une arête s_0s_1 sur une configuration non-variété polygonale (a). (b) L'ensemble des faces triangulaires incidentes à s_0s_1 sont supprimées. (c) s_0 et s_1 sont fusionnés en résolvant tous les doublons d'arêtes et de faces créés. Dans le même temps s_0s_1 est supprimée. (d) Le nouveau sommet résultant s est placé au milieu de l'ancienne arête s_0s_1.

faces triangulaires deviennent invalides ($degré < 3$) et doivent être supprimées.

Pour finir, il est nécessaire de mettre à jour les informations de géométrie et de texture autour de s. Cette mise à jour dépend de la nouvelle position de s. La plupart des méthodes de simplification par contraction d'arête choisissent l'emplacement de s en fonction de divers critères locaux comme la courbure ou la position des sommets adjacents. Ce choix est réalisé pour minimiser la distorsion et/ou conserver la régularité du maillage. Toutefois, dans le cadre de la compression progressive, déterminer une position variable de ce type obligerait à fournir, lors de l'expansion de s durant la décompression, la position originelle de s_0 ET de s_1. Ceci étant coûteux, nous avons décidé de définir la nouvelle position de s comme étant le milieu de s_0s_1, de sorte qu'une seule position soit nécessaire pour retrouver l'emplacement de s_0 et de s_1 (figure 3.3-d). Une amélioration possible de cette stratégie, que nous n'avons pas explorée, serait de proposer plutôt trois positions potentielles pour s : s_0, s_1 ou le milieu de s_0s_1. Cette proposition impliquerait des informations supplémentaires à coder pour l'expansion de sommet. Néanmoins, cela augmenterait la qualité visuelle des différents LoD, tout particulièrement le long des arêtes vives, qui peuvent être malmenées lors de la simplification.

Une fois le sommet résultant de la simplification, s, correctement placé, il est possible de mettre à jour les coordonnées de texture des *corners* qui lui sont associés. Pour cela, il est indispensable de définir une nouvelle notion : l'appariement de *corners*. Avant la contraction de l'arête, nous détectons les différentes régions de la texture présentes sur le patch. En d'autres termes, nous décelons les coutures potentielles. Chaque *corner* du patch est associé à sa région (dans la figure 3.4, c_1 est associé à la première région, en rose). Un *corner* c de s_0 (resp. s_1) est dit *appariable* s'il existe un *corner* c' de s_1 (resp. s_0) associé à la même région (par exemple, c_1 et c_2 sont respectivement incidents à s_0 et s_1 et sont associés à la même région). Il est dit non appariable dans le cas contraire (par exemple, il n'existe pas de *corner* associé à la région verte

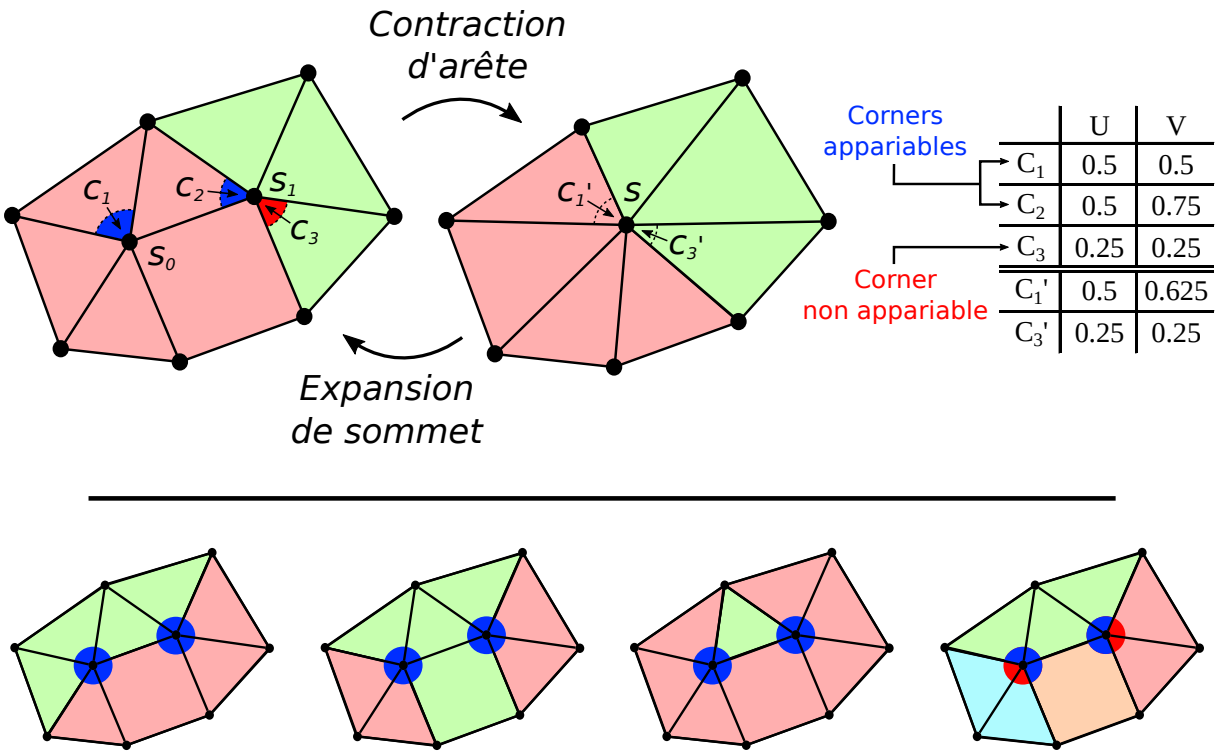
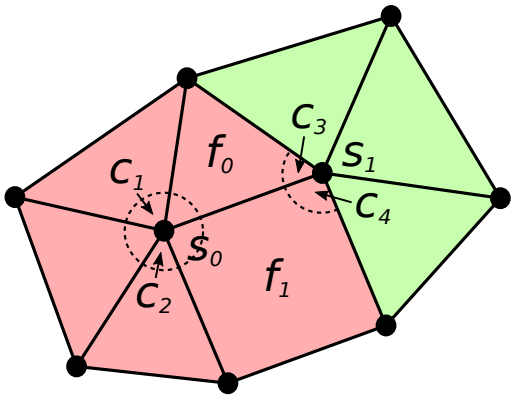


FIGURE 3.4 – En haut à gauche : Schéma expliquant la gestion des *corners* durant la contraction de l'arête s_0s_1. En haut à droite, tableau des valeurs UV de certains *corners*, avant et après la contraction. En bas, différentes configurations de coutures au sein d'un patch avant contraction.

autour de s_0, c_3 est donc non appariable). Cette notion est essentielle car pour mettre à jour les coordonnées UV des *corners* de s après contraction, nous utilisons une interpolation linéaire entre deux *corners* appariables. Suite à la détection des régions, nous associons à chaque *corner* c_i de s_0 et de s_1 leur *corner* c'_i. Les *corners* non appariables sont associés à eux-même ($c'_i = c_i$). Ainsi, après la contraction, il est possible de définir les nouvelles coordonnées des *corners* c_j de s comme le milieu des coordonnées UV de c_j et de c'_j.

Les limites de cette technique sont de deux natures. Tout d'abord, les *corners* non appariables restent inchangés. Comme le sommet résultant de la contraction change nécessairement de position, cela introduit un déplacement de la couture. Nous tentons de minimiser cette erreur dans la Section 3.5. Ensuite, il est préférable que les régions de texture soient convexes. Une région concave engendrerait des artefacts dus à l'interpolation, au niveau des coutures, de la paramétrisation sur les faces du maillage lors de sa visualisation. Nous n'avons pas étudié cette difficulté dans nos travaux.

Enfin, il existe un cas particulier qui n'est pas géré naturellement par cette technique (voir figure 3.5). En effet, il est possible pour deux faces f_0 et f_1, incidentes à une arête s_0s_1, d'être associées, entre autres, à des *corners* c_{00} (f_0, s_0), c_{01} (f_0, s_1), c_{10} (f_1, s_0) et c_{11} (f_1, s_1) de telle sorte que $c_{00} = c_{10}$ mais $c_{01} \neq c_{11}$. s_1 est un sommet dit divisé, alors que s_0 est dit uni. Grossièrement, cela veut dire que f_0 et f_1 sont liées à la même région de texture, sur le patch courant, en étant cependant séparées par une couture. Cette configuration apparait, par exemple, lorsque l'on plaque sur un cube une texture en forme de patron. Pour éviter toute ambigüité, les *corners* incidents au sommet divisé et associés à la région en question ne peuvent être appariés. Ainsi, dans la figure 3.5, c_1 et c_2 ne pourrons être appariés avec c_3 ou c_4 car ces derniers auront été invalidés. Si c_1 et c_2 sont non appariables, leurs coordonnées ne seront pas interpolées. Ce mécanisme engendre donc également un déplacement de la couture.



	U	V
C_1	0.5	0.5
C_2	0.5	0.5
C_3	0.25	0.5
C_4	0.75	0.5

FIGURE 3.5 – Exemple du cas particulier de repli de la texture. Les faces f_0 et f_1 sont associées à la même région de texture, cependant, les *corners* c_3 et c_4, pourtant associés au même sommet et à la même région n'ont pas les mêmes coordonnées UV.

Un tel opérateur de simplification peut ainsi gérer toutes les configurations possibles de patchs surfaciques, tant au niveau de la connectivité que de la paramétrisation de la texture. Nous proposons donc une contraction d'arête générique permettant de traiter des maillages non-variétés polygonaux texturés avec ou sans couture.

3.5 Sélection des sites de simplification

L'opérateur de contraction d'arête étant défini, il est possible de simplifier itérativement le maillage en vue de sa compression progressive. Néanmoins, nous avons souhaité attacher une attention particulière à la qualité de la simplification. C'est pourquoi les arêtes à contracter durant la décimation ne sont pas choisies au hasard. Nous avons mis en place une stratégie de sélection des sites de simplification visant à minimiser la distorsion engendrée. Nous avons également décidé de ne pas axer cette stratégie sur un seul critère, mais au contraire de profiter du maximum d'informations disponibles (connectivité, géométrie, attribut de texture) afin de guider au mieux les contractions d'arêtes.

Nous utilisons une file de priorité sur les arêtes du maillage afin de déterminer leur ordre de contraction. Nous associons à chaque arête a un poids $p(a)$. C'est sur ces poids qu'est défini l'ordre de la file de priorité. Dans notre implémentation, ces poids peuvent être de natures diverses. Il est d'ailleurs possible pour l'utilisateur de proposer assez directement son propre calcul de poids. Pour notre part, nous avons retenu au long de nos expérimentations quatre mesures principales grâce à leurs propriétés et leurs apports.

Le critère de longueur d'arête est une mesure extrêmement simple permettant un tri de la file de priorité cohérent. En effet, utiliser cette mesure implique de contracter en priorité les plus petites arêtes du maillage sans tenir compte de la courbure. Non seulement cette stratégie élimine les plus petits détails en préservant la forme générale de l'objet 3D, mais en plus, elle homogénéise la résolution du maillage. Les zones de forte résolution, parfois sur-échantillonnées ou bruitées, vont être décimées. De plus, la simplicité de ce critère le rend facile et rapide à calculer. C'est un bénéfice de temps non négligeable apporté durant la compression, notamment lorsqu'il s'agit de trier plusieurs centaines de milliers d'arêtes, voire des millions. Le choix du critère de longueur d'arête p_{long} offre des résultats convaincants sur les objets 3D peu complexes et globalement lisses. Cependant, il faut noter plusieurs limitations. En premier lieu, ce critère priorise les zones du maillage les plus résolues. Afin de conserver les caractéristiques principales

du maillage, comme les arêtes vives, ce n'est pas forcément la bonne stratégie. Ensuite, l'ensemble des informations locales ou globales, qui pourrait aider à mieux comprendre le maillage et ainsi mieux le simplifier, ne sont pas utilisées. Par exemple, la paramétrisation de la texture est totalement ignorée. Cela peut engendrer de graves distorsions qui dégraderaient fortement la qualité visuelle du maillage, autant sinon plus que d'importantes distorsions géométriques.

Afin d'obtenir une meilleure approximation de l'erreur géométrique, il est possible d'utiliser la distance de Hausdorff. Dans l'idéal, il serait intéressant de déterminer la distance de Hausdorff globale entre le maillage original et le maillage courant, après la contraction simulée d'une arête a, pour obtenir le poids $p_{haus}(a)$. En pratique, seule la différence géométrique locale entre le patch original de a et le patch courant, après la contraction simulée de a, est nécessaire. Cela permet un calcul plus efficace du poids. Malheureusement, la conservation du maillage original et la maintenance de la correspondance entre les patchs courants et initiaux est coûteuse en terme d'espace mémoire. Notre solution consiste à simplifier le calcul de l'erreur en définissant $p_{haus}(a)$ comme la distance de Hausdorff $d_{haus}(a)$ entre le patch de a avant et après la simulation de contraction. Cette mesure permet une bonne approximation de la distorsion géométrique provoquée par la contraction d'une arête donnée. Nous avons toutefois souhaité inclure des informations de connectivité dans le calcul de ce critère. Dans le cadre du traitement de maillages non-variétés, certaines contractions d'arêtes ne sont pas souhaitables. Elles auraient des effets négatifs sur la forme générale de l'objet traité qui serait difficilement mesurable avec la distance de Hausdorff. Durant nos travaux, nous avons répertorié trois groupes d'arêtes.

1. Les arêtes pendantes, durant leur contraction, peuvent entraîner une réduction significative de la boite englobante du maillage. La décimation de ces arêtes peut aussi provoquer la fusion de deux parties distinctes du maillage, ce qui conduirait à perdre visuellement la cohérence spatiale du modèle.
2. Les arêtes dont la contraction produirait des arêtes pendantes dans leur voisinage sont également particulières. Simplifier ce type d'arêtes revient à une perte, parfois importante, d'aire sur le maillage. En effet, la création d'une arête pendante est le résultat de la suppression d'une face incidente qui ne serait pas forcément compensée par la croissance d'autres faces du patch (contrairement à la contraction d'une arête dans une configuration localement 2-variété).
3. L'ensemble des arêtes restantes constituent le troisième groupe.

Pour résumer, en identifiant les arêtes du maillage, nous souhaitons minimiser les pertes d'aires et de volume pour rester fidèle le plus longtemps possible à l'aspect général du modèle original. Pour cela, $p_{haus}(a)$ sera calculé différemment selon le groupe de a :

$$p_{haus}(a) = \begin{cases} diago + p_{long}(a) & \text{si } a \text{ est une arête pendante} \\ 2 \times diago + \sqrt{\Delta aire(a)} & \text{si au moins une arête pendante est produite} \\ d_{haus}(a) & \text{sinon} \end{cases} \tag{3.4}$$

$diago$ est la diagonale de la boite englobante et $\Delta aire(a)$ est la différence d'aire, entraînée par la contraction de a, sur le patch. La diagonale de la boite englobante est utilisée ici comme un terme discriminant car c'est une valeur assez stable au long de la décimation et qu'il n'est pas possible pour une longueur d'arête ou une différence d'aire d'être supérieure à cette valeur. L'objectif de cette mesure est de prioriser le troisième groupe d'arêtes sur lequel le critère de distance d'Hausdorff est calculable de façon certaine. Ensuite, on retrouve dans la file de priorité les arêtes pendantes, triées en fonction de leur longueur, c'est à dire, grossièrement, de l'impact

qu'aura leur potentielle contraction. Enfin, les arêtes dont la contraction produit des arêtes pendantes sont classées en dernier. Cet ordre a été choisi car il nous paraissait évident que la suppression d'arêtes pendantes était grandement préférable à leur création, durant le processus de simplification. De plus, il faut noter qu'une arête pendante ne sera pas prise en compte lors du rendu (à part traitement spécifique). Cela veut dire qu'elles n'apportent rien visuellement. Il est donc tout à fait contreproductif d'en produire de nouvelles. Cette mesure est plus lourde à calculer que p_{long} mais donne de meilleurs résultats, que cela soit sur des maillages 2-variétés lisses, comme le montre la figure 3.6, ou non-variétés beaucoup plus complexes. Par contre, comme le critère précédent, elle ne prend pas en compte les distorsions possibles liées à la texture.

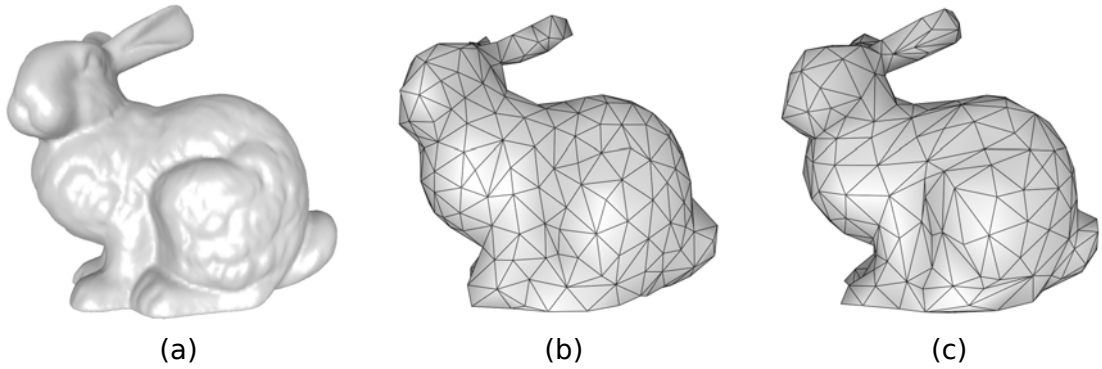


FIGURE 3.6 – Comparaison visuelle de la simplification du modèle *Bunny* (35,947 sommets), dont l'original est montré en (a), à l'aide du critère (b) p_{long} (1,416 sommets) et (c) p_{haus} (1,430 sommets). On note qu'à plusieurs endroits, le LoD obtenu grâce à p_{long} ne respecte plus la géométrie du maillage original. p_{haus} offre une meilleure fidélité au maillage l'original.

Afin de minimiser l'erreur produite au niveau de la paramétrisation de la texture durant la simplification, nous proposons un critère adapté. Une telle distorsion peut avoir deux causes. La première est une interpolation erronée de la paramétrisation due à la présence d'une région de texture concave. Une piste intéressante permettant de limiter l'apparition de cette situation serait de s'appuyer sur une analyse des régions de texture pour détecter ces cas. Malheureusement, nous n'avons pas approfondi cette voie. Cela reste donc une piste d'amélioration. La deuxième cause est l'impossibilité ou la non-volonté d'interpoler certaines coordonnées UV au sein des *corners* durant certaines contractions d'arêtes lorsqu'une couture est présente sur le patch. Cette absence d'interpolation provoque visuellement un déplacement de la texture au niveau de ses coutures. Le principe de notre approche est de prioriser les arêtes provoquant le moins de déplacement de texture lors de leur contraction. Pour ce faire, nous calculons p_{text} de la manière suivante :

$$p_{text} = \frac{p_{long}(a)}{2} \times nbDéplacements \tag{3.5}$$

$nbDéplacements$ correspond au nombre de frontières de régions déplacées lors de l'éventuelle contraction. Une simple couture passant par s_0 et s_1 sur le patch de l'arête s_0s_1 n'entraînera pas de déplacement à la décimation de s_0s_1 et donc $nbDéplacements$ sera nul. Nous avons préféré utiliser le déplacement de sommet dans l'espace 3D plutôt que la distance de déplacement dans l'espace de la texture pour rester consistant avec les mesures précédentes. Cela apporte une certaine homogénéité. Le calcul de ce poids permet d'obtenir une préservation excellente des coutures appliquées sur le maillage. En d'autres mots, les arêtes contenues strictement à l'intérieur d'une région de texture ou dans l'alignement de sa frontière sont contractées en premier. Cependant, cela laisse la géométrie et la connectivité du maillage sans aucun contrôle,

ce qui n'est pas préférable à l'utilisation de p_{haus}.

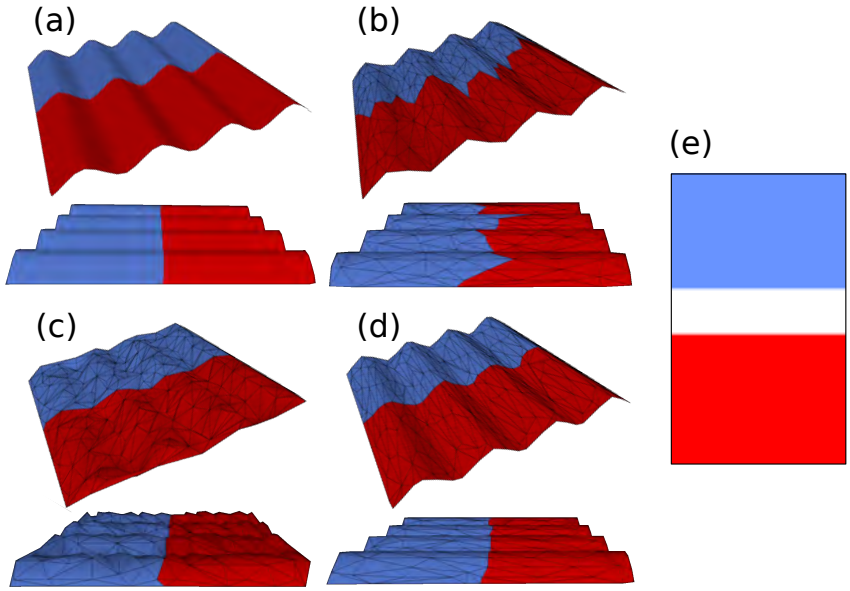


FIGURE 3.7 – (a) Présentation d'un maillage plan ondulé initial (1089 sommets), (b) le maillage simplifié avec p_{haus} (221 sommets), (c) le maillage simplifié avec p_{text} (264 sommets), (d) le maillage simplifié avec p_{comb} (234 sommets). Nous fournissons également, en (e), l'image de texture associée.

En faisant le constat que, seules, les mesures p_{haus} et p_{text} n'étaient pas suffisantes pour guider correctement la simplification de maillages non-variétés texturés, nous proposons une combinaison de ces deux critères. Cette nouvelle solution p_{comb} est une simple addition pondérée des deux mesures en question :

$$p_{comb}(a) = \alpha \times p_{haus}(a) + (1 - \alpha) \times p_{text}(a), 0 \leq \alpha \leq 1 \tag{3.6}$$

Ainsi, il est possible d'influencer le calcul des poids avec la variable α pour privilégier la préservation de la géométrie et de la connectivité ou, au contraire, de mettre l'accent sur la conservation de la paramétrisation de la texture. Nous avons volontairement choisi de fixer α à $1/2$ en estimant que les deux propriétés étaient d'égale importance pour la qualité visuelle du rendu. Ce compromis est permis par l'homogénéité appréciable des deux critères. Le résultat de cette sélection est visible sur un simple exemple de plan ondulé (voir figure 3.7). Dans cet exemple, p_{haus} (figure 3.7-b) ne limite pas le déplacement de la couture et p_{text} préserve très mal les caractéristiques géométriques du maillage (figure 3.7-c). p_{comb}, en revanche, tient compte de ces deux aspects afin de maintenir une qualité visuelle satisfaisante pour ce LoD (figure 3.7-d).

La mesure utilisée pour le calcul du poids de chacune des arêtes permet de trier ces dernières en fonction du résultat attendu. Après le pré-traitement du maillage en entrée, nous affectons à toutes les arêtes de $\mathcal{M}^n$ son poids. Durant la simplification, après chaque contraction d'une arête s_0s_1, les poids de l'ensemble des arêtes du patch de s_0 sont mis à jour afin d'actualiser la file de priorité. Les arêtes supprimées en cours de simplification sont retirées de la file.

3.6 Vagues de décimation

Pour chaque contraction d'une arête s_0s_1, il est nécessaire de conserver les informations, notées $info(s)$, permettant de procéder au raffinement du sommet s correspondant, lors de la décompression (voir Section 3.9), pour tendre vers la restitution complète du maillage $\mathcal{M}^n$. Cependant, il est également important pour le décodeur de pouvoir retrouver le bon sommet s à traiter. Hoppe, dans [Hop96], code explicitement l'identifiant de ces sommets. Au regard des travaux suivants, comme [AD01a] ou [KBG02], il apparaît nettement que cette stratégie a peu d'avantages et reste extrêmement coûteuse. La mise en place de vagues de décimation, largement employées en compression progressive de maillage, améliore significativement le taux de compression. Le principe d'une vague de décimation v_i est de regrouper un nombre k_i d'opérations unitaires de simplification le long d'un parcours déterministe sur le maillage. Ce parcours, connu à la fois du codeur et du décodeur (voir Section 3.7.4), permet de savoir implicitement quel élément est à raffiner pendant la décompression. Il est possible que certaines informations doivent être codées concernant le parcours en lui-même. Quoi qu'il en soit, la stratégie reste très avantageuse. À l'instar de plusieurs travaux précédents, notamment [PR00], nous proposons une simplification du maillage $\mathcal{M}^n$ par vagues.

Chaque vague v_i aura pour but de simplifier le maillage courant $\mathcal{M}^i$ en $\mathcal{M}^{i-1}$. Le processus s'arrête lorsque le maillage $\mathcal{M}^0$ est atteint. Ce dernier ne contient plus ni arête, ni face (toutes décimées) mais seulement m sommets. m est le nombre de composantes connexes de $\mathcal{M}^n$ (après pré-traitement du maillage en entrée). Pour chaque vague v_i, notre méthode procède en deux phases : une série de k_i contractions d'arêtes s_ps_q $(p \neq q)$ sur le maillage $\mathcal{M}^i$ et la construction d'un parcours déterministe sur $\mathcal{M}^{i-1}$ permettant d'ordonner les $info(s_p)$ prélevées au fur et à mesure des contractions. La récupération des $info(s_p)$ ainsi que la seconde phase sont décrites respectivement dans les Sections 3.7.2, 3.7.3 et 3.7.1.

En ce qui concerne la série de contractions, elle fait appel à la file de priorité des arêtes du maillage. À chaque itération, la première arête contractable est retirée de la file. Elle est ensuite décimée par notre opérateur générique de simplification, puis la file de priorité est mise à jour. Comme suggéré plus haut, toutes les arêtes ne sont pas contractables. Certains critères sont à respecter. L'utilisateur peut, par exemple, vouloir éviter un changement topologique ou une inversion brutale de la normale à une face voisine. Ces limitations sont optionnelles. Cependant, des contraintes obligatoires sont posées pour le bon déroulement de la méthode.

Deux configurations rendent une arête s_0s_1 non contractable (voir figure 3.8). La première, généralement bien identifiée dans la littérature, est la situation dans laquelle un triangle f_0, tel que $f_0 \prec s_0s_1$, serait soit adjacent à deux triangles, f_1 et f_2, tels que $f_1 \prec s_0$ et $f_2 \prec s_1$, soit adjacent par deux arêtes à une face f_3, telle que $degré(f_3) > 3$ et $f_3 \not\prec s_0s_1$ (figure 3.8-a et b). Cette configuration crée, par définition et dans tous les cas, des faces dégénérées lors de la contraction de s_0s_1. Deuxièmement, à cause de la gestion des faces polygonales, un recouvrement de deux faces f_0 et f_1 est possible, avec $degré(f_0) < degré(f_1)$ (figure 3.8-c). Une configuration similaire peut être obtenue si f_0 est remplacée par un trou (figure 3.8-d). Ce recouvrement n'invalide pas géométriquement ou topologiquement le maillage. Mais la contraction d'une arête a, telle que $a \prec f_0$ et $a \not\prec f_1$, produit des faces dégénérées. Le recouvrement potentiel de faces est plus facile à détecter que la configuration dérivée. De plus, il est souvent générateur d'artefacts nuisibles à la visualisation d'un maillage. Nous avons donc préféré interdire ce recouvrement. Il est important de noter que dans l'absolu, même si le résultat d'une contraction sur de telles configurations n'est pas valide, l'utilisation de notre opérateur de simplification, dans ces cas précis, fonctionne. Les situations sont interdites car elles propageraient une connectivité non valide qui serait alors problématique pour la suite de la simplification. Cela serait également bien trop coûteux de coder la résolution de ces cas. Il faut noter que les configurations (c) et

(d) de la figure 3.8 sont détectées seulement sur des maillages polygonaux et ont une fréquence d'apparition extrêmement faible. De plus, ces situations ne sont pas définitives. La contraction d'une des arêtes voisines permet, souvent assez rapidement, à s_0s_1 de redevenir contractable.

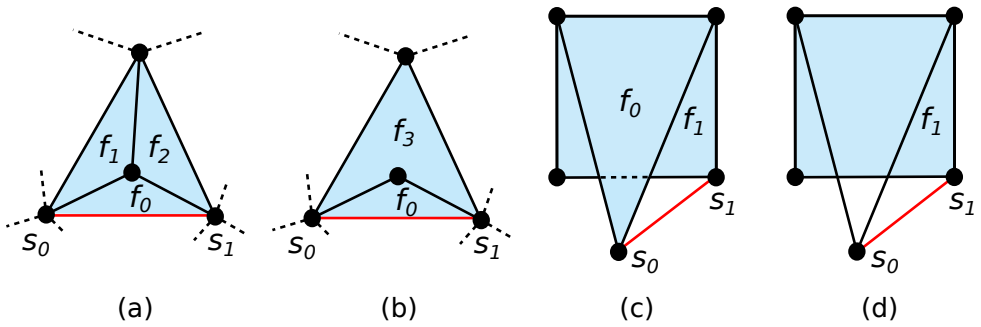


FIGURE 3.8 – Quatre configurations dans lesquelles l'arête rouge est dite non contractable.

Une autre restriction est appliquée sur les arêtes à contracter. En effet, étant donné que l'ordre des contractions réalisées au cours d'une vague dépend de la file de priorité, cela ne correspond pas à l'ordre du parcours établi par la suite. Afin d'éviter tout conflit entre contraction lors du raffinement, il est impératif que chaque simplification soit indépendante. Pour garantir cela, une arête ne peut être décimée si un des sommets de son patch est le résultat d'une contraction précédente. Cette contrainte est plus largement discutée dans la Section 3.7.1.

Une vague de décimation se termine lorsque le poids de l'arête courante, au sein de la file de priorité, dépasse un certain seuil. La difficulté réside dans le fait que dépendre d'un seuil statique ne permet pas de simplifier totalement le maillage. Un seuil sur le nombre d'arêtes à décimer par vague, même progressif, s'adapterait à tous les maillages de la même manière, ce qui intuitivement ne paraît pas la bonne solution. Certains maillages, constitués de surfaces peu courbées mais très résolues, devraient permettre un grand nombre de simplification, au moins pour les premières vagues. Au contraire, des maillages complexes avec beaucoup de courbures nécessiteraient des vagues plus courtes pour minimiser la distorsion.

Dans ce contexte, nous proposons un seuil dynamique appliqué aux poids de la file de priorité. Pour ne pas trop diminuer les performances de la simplification, nous avons choisi de retenir la moyenne des poids comme valeur seuil, recalculée au début de chaque vague. Lors de nos expérimentations, ce choix simple s'est avéré efficace dans la mesure où le nombre de contractions k_i par vague v_i restait suffisamment grand tout en respectant les propriétés géométriques des maillages traités. Nous pouvons d'ailleurs remarquer la cohérence de ce seuil avec la figure 3.9 qui montre la courbe des poids mesurés sur les arêtes avant la première vague de décimation du modèle *Tractor*. Il est malgré tout possible de s'appuyer sur une analyse plus complexe de la courbe représentant tous les poids de la file de priorité pour déterminer ce seuil, pour chacune des vagues. Une étude des variations de la courbe permettrait éventuellement d'optimiser la répartition des vagues.

Dans notre approche, nous avons constaté l'importance du compromis entre la maximisation du nombre de contractions par vague et la minimisation de la distorsion. L'idéal, en terme de distorsion, est de ne réaliser qu'une contraction par vague, de façon similaire à [Hop96]. La distorsion serait alors minimale au regard du critère employé pour le calcul du poids des arêtes. Malheureusement, cela entraînerait inévitablement n_s vagues de décimation. Nous rappelons que pour chaque vague, un parcours est construit et devra être codé. Ce choix pénalisera donc fortement le taux de compression. À l'inverse, procéder au maximum de contractions durant chaque itération réduirait considérablement le nombre de vagues nécessaires. Il en découlerait une baisse importante de l'information à coder concernant les différents parcours. Mais cela

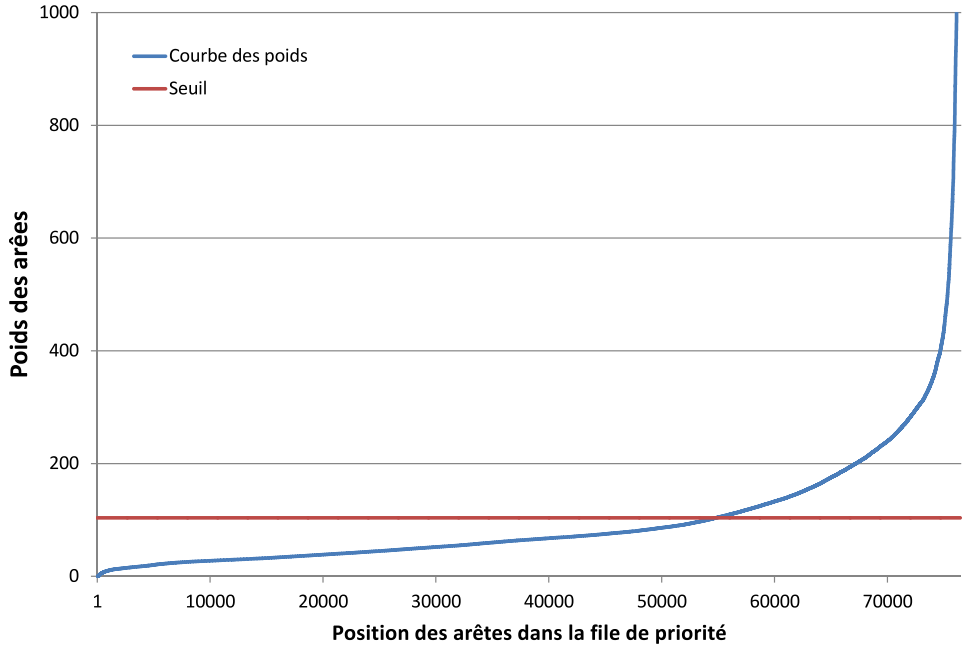
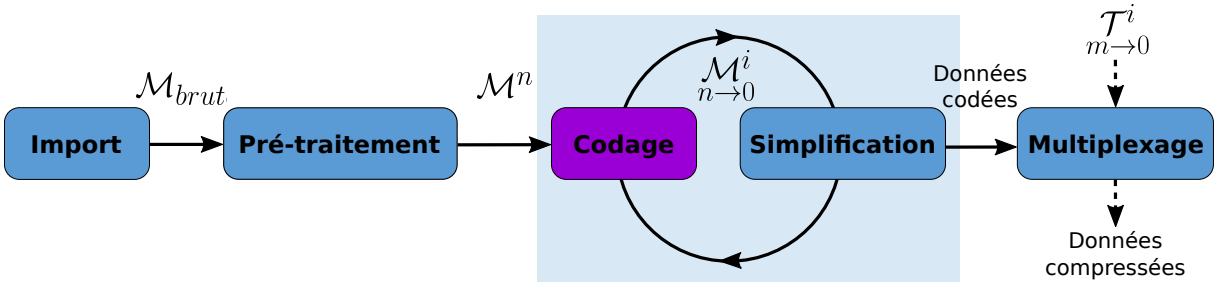


FIGURE 3.9 – Courbe des poids mesurés grâce à *pcomb* avant la première vague de décimation du modèle *Tractor* (en bleu). La ligne rouge représente le seuil calculé par la moyenne des poids. Seules les arêtes dont le poids est inférieur au seuil peuvent être contractées durant la vague courante.

aurait pour conséquence de supprimer totalement le contrôle de la distorsion.

Le seuil dynamique proposé plus haut nous semble adapté pour répondre à ce compromis, différent pour chaque maillage. Les performances de cette approche sont d'ailleurs évaluées dans la Section 4.2. Pour les maillages que nous avons utilisés lors de nos travaux, le nombre de vagues varie approximativement entre 30 et 60.

3.7 Encodage



La simplification progressive du maillage décrite dans les sections précédentes ne constitue que la première étape de notre méthode de compression. Il est également nécessaire de coder les informations qui permettent d'effectuer le raffinement du maillage afin de restituer $\mathcal{M}^n$. Au cours d'une vague, cette opération se déroule en deux temps. Tout d'abord nous stockons, pour chaque arête s_0s_1 contractée, les informations $info(s_0)$. Ensuite, nous concaténons ces données codées dans l'ordre de construction d'un parcours sur le maillage courant (voir Section 3.7.4).

Cette partie a pour but de décrire, d'une part, le codage des $info(s_i)$ permettant de reconstruire un patch en fonction du patch de s_i. $info(s_i)$ est composé de plusieurs types d'informations pour assurer une restitution exhaustive : la géométrie, la connectivité et la paramétrisation de la texture, dans cet ordre précis. L'ensemble de ces informations, pour une arête donnée s_0s_1, sont récupérées sur le patch de s_0s_1 en simulant sa contraction, avant que la réelle contraction ne soit exécutée. D'autre part, nous expliquons la construction, pour chaque vague de décimation, du parcours ordonnant les informations collectées à l'itération courante. Nous dévoilons également les différentes stratégies et optimisations réalisées sur ce parcours pour améliorer son codage.

3.7.1 Connectivité

Pour le codage de la connectivité du patch d'une arête s_0s_1, nous nous sommes inspirés des travaux de Hoppe [Hop96] et de leur extension aux patchs triangulaires non-variétés [PH97]. Nous proposons ainsi une ultime extension pour les patchs polygonaux non-variétés. Cela permet d'appréhender la totalité des configurations surfaciques (à l'exception des cas discutés dans la Section 3.4).

Le principe central est d'associer, à chaque arête et face adjacente à s_0s_1, un code en fonction de sa connectivité au sein du patch. Ces codes serviront au décodeur pour reconstruire le patch initial. Le tableau de correspondance des codes est donné par la figure 3.10. La ligne (a) de la figure liste les différents codes possible (de 0 à 3).

La ligne (b) correspond aux configurations possibles relatives aux arêtes :
- Un code 0, appliqué à une arête a incidente à s avant l'expansion (pendant la décompression), signifie que a sera incidente à s_0.
- À l'inverse, un code 1 obligera a à devenir incidente à s_1.
- Le code 2 concerne les arêtes qu'il va falloir séparer en deux nouvelles arêtes sans création de face. Une arête devient incidente à s_0, elle est notée s_0s_2. L'autre arête sera incidente à s_1, elle est notée s_1s_2.
- Le code 3 concerne les arêtes qu'il va falloir séparer en deux nouvelles arêtes avec création de face. Une arête devient incidente à s_0, elle est notée s_0s_2. L'autre arête sera incidente à s_1, elle est notée s_1s_2. Le code 3 spécifie simplement que ces deux nouvelles arêtes et s_0s_1 forment une nouvelle face $s_0s_1s_2$.

La ligne (c) correspond aux configurations possibles relatives aux faces :
- Un code 0 est donné pour une face qui deviendra incidente à s_0.
- Un code 1 est donné si, au contraire, elle doit être incidente à s_1.
- Le code 2 représente également une séparation, mais de deux faces cette fois-ci.
- Enfin, nous avons modifié le code 3, qui est un apport de notre méthode par rapport à [PH97], afin de spécifier les faces polygonales incidentes à s_0s_1. Ce dernier code est affecté aux faces de degré supérieur à 3, les faces triangulaires étant déjà gérées dans les configurations précédentes (code 3 pour les arêtes).

L'ensemble des codes présentés permet de décrire des configurations polygonales non-variétés. Néanmoins, la concaténation de ces différents codes (d'abord pour les arêtes, puis pour les faces) est exagérément longue. [PH97] met en lumière des contraintes topologiques entre ces codes et donc des redondances inutiles. Par exemple, dans leur cas, si une arête a doit être incidente à s_0 (resp. s_1), alors une face f, telle que $f \prec a$, sera forcément incidente à s_0 (resp. s_1). Son code est donc inutile.

Ces déductions diminuent considérablement la longueur des codes décrivant la connectivité et sont donc bénéfiques, à terme, pour le taux de compression. Pour notre part, nous avons souhaité pousser ces contraintes plus loin encore, tout en les adaptant au contexte polygonal.

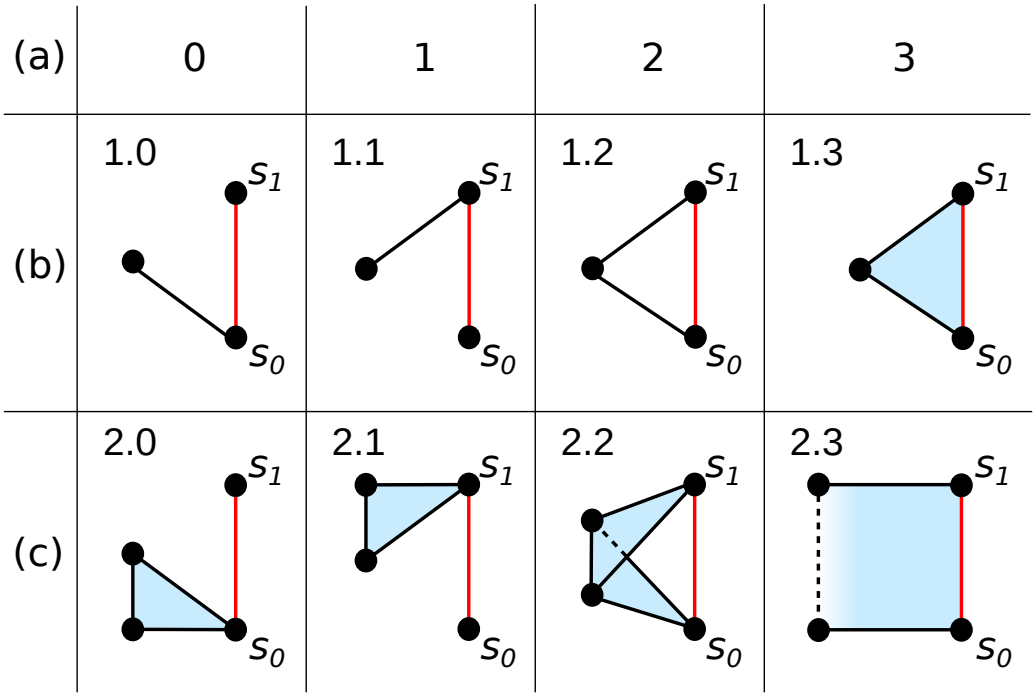


FIGURE 3.10 – Tableau des différentes configurations présentes autour d'une arête s_0s_1 à contracter. (a) Les codes générés lorsque la configuration est rencontrée, (b) les configurations concernant les arêtes adjacentes à s_0s_1, (c) les configurations concernant les faces incidentes à s_0s_1.

Cette optimisation se présente en deux parties (voir figure 3.11) : la déduction par contrainte et la prédiction de codes. La première partie se repose entièrement sur les codes associés aux arêtes, qui sont tous obligatoires. Après avoir pris connaissance de ces codes, le décodeur peut entièrement déduire les codes associés aux faces. Si deux arêtes d'une face sont de code 0 (resp. 1) alors la face est de code 0 (resp. 1). Si la face a une arête de code 0 (resp. 1) et une arête de code 2 ou 3, alors la face est de code 0 (resp. 1). Si deux arêtes d'une face sont de code 2 ou 3, la face a pour code 2. Et enfin, si une face a une arête de code 0 et une arête de code 1, elle est de code 3. Ces implications rendent les codes associés aux faces obsolètes et permettent donc de supprimer près de la moitié des codes générés pour décrire la connectivité du patch.

Dans un second temps, nous avons souhaité réduire l'entropie des codes restants (associés aux arêtes). Pour cela, nous nous appuyons sur un système de prédiction. Cette amélioration se base sur le constat que, sur un maillage, plus deux sommets sont proches, plus ils ont de chances d'être adjacents (i.e. liés par une arête). Cela est d'autant plus vrai sur un maillage régulier uniformément échantillonné. Ainsi, nous estimons que, sur le patch d'une arête s_0s_1, lorsqu'une arête a, adjacente à s_0s_1, est associée au code 0 (resp. 1), le sommet de a opposé à s_0 (resp. s_1) sera statistiquement plus proche de s_0 (resp. s_1) que de s_1 (resp. s_0). Au lieu de générer simplement un code 0 ou 1 pour a, comme présenté précédemment, nous prédisons avec quel sommet (s_0 ou s_1) elle devrait être incidente. Puis nous générons un 0 si la prédiction est effectivement juste et un 1 sinon, comme le montre la figure 3.11-c. Cette prédiction sera également faite par le décodeur. En fonction du code lu, ce dernier sera alors capable de confirmer ou d'infirmer la prédiction pour retrouver la bonne connectivité. Pour que la prédiction ait un sens, il faut que les positions originelles de s_0 et s_1 soient connues et donc que les informations géométriques (présentées dans la Section 3.7.2) soient lues et décodées avant les informations de connectivité, lors de la décompression.

L'utilisation de cette prédiction basée géométrie permet, après compression, de réduire la

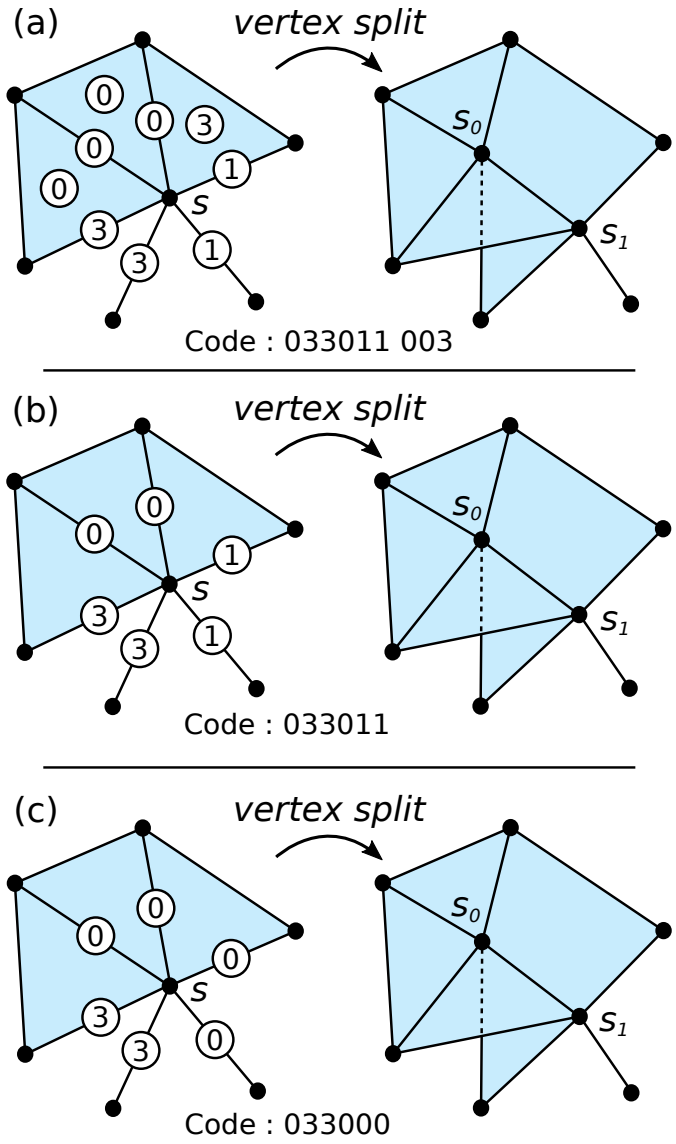


FIGURE 3.11 – Représentation du codage de la connectivité sur un patch polygonal non-variété. (a) Codage sans optimisation avec les codes associés aux arêtes et aux faces. (b) Codage par contraintes topologiques, seuls les codes associés aux arêtes sont générés, ceux associés aux faces sont déterminés implicitement. (c) Codage par contraintes topologiques et prédiction géométrique, similaire au (b) mais les codes 0 et 1 sont prédits en fonction de la position de s_0 et s_1.

taille des codes 0 et 1 associés aux arêtes de près de 90%, en moyenne, grâce à la diminution de l'entropie qu'elle apporte. Ce gain est d'autant plus intéressant que ces codes représentent une grande majorité des données liées à la connectivité.

Pour être certain de retrouver la connectivité exacte du maillage, il est important de fournir une information supplémentaire. Pour des modèles non-variétés, l'orientation des faces n'est pas toujours consistante sur la globalité du maillage. L'orientation des faces triangulaires nouvellement créées lors d'une expansion n'est donc pas toujours déterministe. Nous sommes donc forcés de prédire le comportement du décodeur. En effet, à la décompression, l'ensemble des nouvelles faces, notée $\mathcal{F}$, doit être orienté correctement. Pour cela, pour chaque face de $\mathcal{F}$, nous nous appuyons sur ses faces adjacentes (sauf les faces de $\mathcal{F}$ qui n'ont pas encore été traitées) pour prédire son orientation. Si aucune orientation consistante n'est trouvée ou s'il n'y a pas de face

adjacente, la face courante est orientée dans le sens $\overrightarrow{s_0 s_1}$. Le codeur réalise la même prédiction et va alors générer un 0 si elle est vérifiée et un 1 sinon. Le décodeur pourra alors déduire de ces codes la bonne orientation.

L'orientation correcte des faces n'est pas nécessaire pour toutes les applications, ainsi cette partie pourrait être optionnelle pour optimiser la compression, sans pour autant modifier le fonctionnement de nos algorithmes. Néanmoins, dans un contexte de visualisation de maillage sur le Web, la cohérence de l'orientation des faces est importante. Cela justifie le soin particulier apporté à cette information.

3.7.2 Géométrie

Afin que le décodeur puisse restituer la position initiale de s_0 et s_1 lors de l'expansion de s, il est nécessaire d'inclure cette information dans $info(s)$. Une solution naïve serait de transmettre directement les coordonnées originelles de s_0 et s_1. Cependant, la géométrie est généralement la donnée la plus coûteuse lors de la compression d'un maillage. Il est donc important d'optimiser son codage. Ainsi, nous utilisons des coordonnées relatives à s pour décrire la position de s_0 et s_1. De plus, comme évoqué à la Section 3.4, s est placé au milieu de $s_0 s_1$ durant la contraction. En se servant de la relation $\overrightarrow{ss_0} = -\overrightarrow{ss_1}$, il est possible de retrouver la position de s_0 et s_1 seulement à l'aide du vecteur $\overrightarrow{ss_0}$.

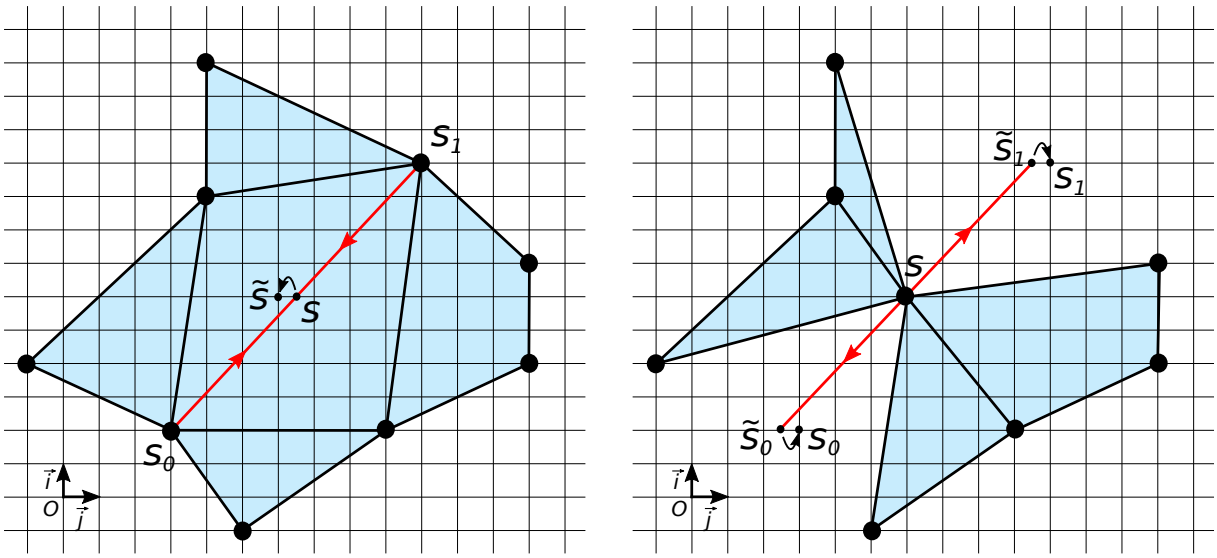


FIGURE 3.12 – À gauche, la contraction de $s_0 s_1$ place s sur des coordonnées non-entières (tout comme le vecteur à encoder), un arrondi au plus proche entier dans la direction de l'origine est effectué. À droite, lors de l'expansion de s, le vecteur fourni place logiquement s_0 et s_1 sur des coordonnées non-entières, de façon déterministe, un arrondi au plus proche dans la direction des infinis est effectué.

Les sommets s_0 et s_1, comme les autres, possèdent des coordonnées quantifiées, donc entières. Il est possible que les coordonnées de s ne soient pas entières (voir figure 3.12). s ne serait donc plus sur la grille tridimensionnelle implicitement tracée par la quantification. Les simplifications suivantes autour de s empireraient cet état. Pour résoudre ce problème, nous décidons de déplacer s, devenant $\tilde{s}$, aux coordonnées entières les plus proches (dans la direction de l'origine) tout en conservant le vecteur ss_0. Ainsi, durant la compression, le maillage ne contient que des sommets à coordonnées entières. Les vecteurs $\overrightarrow{ss_0}$ ont des coordonnées entières ou, dans le pire des cas, constituées de valeurs médianes. Utiliser $2\overrightarrow{ss_0}$ permet d'être certain de manipuler des

valeurs entières, sans pénaliser le taux de compression grâce au codage entropique final. À la décompression, l'utilisation de $2\overrightarrow{ss_0}$ pourra donner un sommet s_0 aux coordonnées non entières. Dans ce cas, elles seront arrondies au plus proche (en s'éloignant de l'origine). À partir de s_0, nous retrouvons s_1 avec la relation suivante : $s_1 = s_0 - 2\overrightarrow{ss_0}$.

Afin d'optimiser encore le codage de l'information géométrique, il est possible de déplacer $2\overrightarrow{ss_0}$ dans un repère de Frenet. Cette technique, déjà mise en place dans [AD01a] et [LLD09], consiste à construire un nouveau repère local dont l'origine correspond à l'origine du vecteur à transformer et les axes sont dirigés en fonction de la surface. L'intérêt d'un tel changement de repère est de réduire l'information d'au moins une des dimensions. Cette stratégie est très efficace pour des surfaces lisses, par exemple. Pour notre méthode, nous avons utilisé l'implémentation de [LLD09] pour transformer $2\overrightarrow{ss_0}$ en $2\overrightarrow{ss_0}$, à la compression, puis $2\overrightarrow{ss_0}$ en $2\overrightarrow{ss_0}$, à la décompression.

Si cette approche est fortement adaptée aux maillages 2-variétés, comme le montre [AD01a], elle est cependant moins naturelle sur des maillages non-variétés. La construction d'un repère de Frenet n'a pas de réel sens lorsqu'elle est réalisée au sein d'un patch localement non-variété. De plus, le placement du repère peut facilement être de mauvaise qualité. Malheureusement, sans information supplémentaire, il n'est pas possible pour le décodeur de déterminer, avant l'expansion de sommet, si un patch 2-variété deviendra non-variété ou non. Nous ne pouvons donc pas utiliser cette technique seulement dans des cas adaptés sans surcoût de données. Nous appliquons la transformation pour toutes les simplifications sans discernement en nous appuyant sur deux hypothèses. Premièrement, même s'il empêche d'améliorer le codage du vecteur, le mauvais placement du repère de Frenet, dû à la non-variété du patch, ne le pénalise pas particulièrement. Deuxièmement, la plupart des maillages non-variétés sont finalement constitués d'une majorité de patchs localement 2-variétés. Cela rend donc cette optimisation efficace, même sur des maillages complexes.

3.7.3 Paramétrisation de la texture

En plus de la connectivité et de la géométrie, nous codons également la paramétrisation de la texture sur le patch. Le principe est de restituer les coordonnées de texture initiales des *corners* associés à s_0 et s_1. ainsi que celles des *corners* supprimés lors de la contraction. Pour cela, nous codons deux types d'informations distinctes : des indices de région de texture et des déplacements de coordonnées (ou les coordonnées en elles-mêmes).

Tout d'abord, les régions de texture du patch sont indicées dans l'ordre de leur détection (voir figure 3.13). La formalisation de ces régions permet une meilleure compréhension de la paramétrisation locale du patch par rapport à une interprétation des coordonnées de texture seules. Cet ordre est le même pour le codeur et pour le décodeur. Les indices sont compris entre 0 et $n_r - 1$, n_r étant le nombre de régions détectées. Une région nouvelle sur le patch pour le décodeur aura l'indice n_r pour spécifier sa création. Dans le cas particulier énoncé dans la Section 3.4 et décrite par la figure 3.5, les faces impliquées seront associées à l'indice $n_r + i$ (i étant l'indice de leur région). Pour chaque face incidente à s_0s_1, nous générons l'indice de sa région. Si aucune couture n'est présente sur le patch, seuls des 0 sont générés, ce qui implique une entropie nulle. En pratique, il est rare de trouver plus d'une couture (i.e. deux régions) sur un patch. Ainsi, dans nos expérimentations, nous utilisons moins d'un bit par sommet, en moyenne, pour la compression de ces indices.

En ce qui concerne les coordonnées de texture, il existe trois cas de figure :

1. Lorsque la face possède un indice de région connue par le décodeur (compris entre 0 et $n_r - 1$), il est possible de ne transmettre qu'un vecteur de déplacement pour résoudre

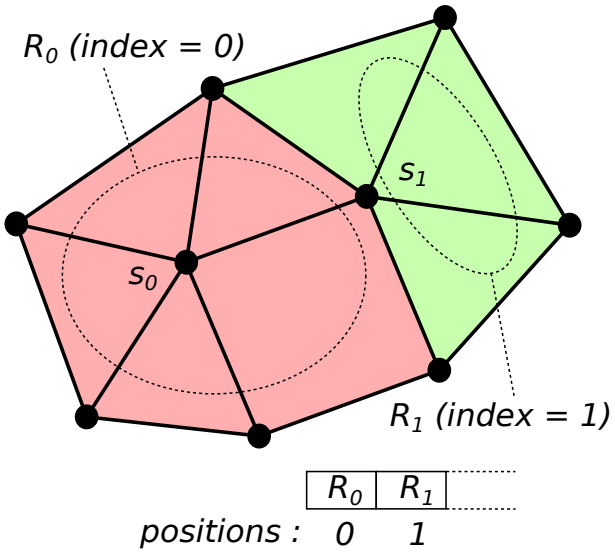


FIGURE 3.13 – Patch 3D montrant les différentes régions de texture détectées et leurs positions.

tous les *corners* de la région. En effet, les *corners* d'une même région autour d'un même sommet possèdent, dans ce cas, les mêmes coordonnées. Cette propriété permet également de ne fournir qu'un vecteur de déplacement par groupe de faces incidentes à s_0s_1 ayant le même indice de région.

2. La région est nouvelle pour le décodeur (indice n_r), il faut spécifier explicitement les coordonnées UV de tous les *corners* de la face.
3. Dans le cas d'une face d'indice de région $n_r + i$, elle sera traitée comme dans le cas (1). Cependant, il sera nécessaire de fournir un vecteur de déplacement pour chacune de ces faces.

Notre méthode code la paramétrisation de la texture avec un indice de région par face incidente à s_0s_1, un vecteur de déplacement par indice de région rencontré – sauf pour le cas (3) où chaque face nécessite un vecteur propre – et trois paires de coordonnées par région inconnue. La présence de régions inconnues est une situation exceptionnelle. Elle est directement liée au nombre total de régions sur la paramétrisation de $\mathcal{M}^n$.

Afin d'optimiser les codages des vecteurs de déplacement, nous avons décidé d'utiliser une nouvelle fois une prédiction basée géométrie, tout comme Váša et Brunnett [VB14]. Nous partons de la même hypothèse, cependant, notre stratégie est très différente. Le constat que font Váša et Brunnett est que, généralement, il existe une certaine similarité de forme entre les faces du maillage et les faces associées, formées par la paramétrisation (que nous appellerons les faces de texture). Il s'agit d'une hypothèse de paramétrisation conforme. En exploitant cette similarité, il est possible, pour le décodeur, une fois la géométrie originelle résolue, de s'en servir pour prédire les déplacements des coordonnées de texture.

Pour réaliser cette prédiction, nous choisissons une arête s_2s_3 (voir figure 3.14) du patch comme référence pour une région donnée. Cette arête est choisie de manière déterministe pour qu'elle soit connue du codeur et du décodeur. Nous calculons ensuite les coordonnées barycentriques de s_0 en fonction du repère ss_2s_3. En appliquant ces coordonnées à la face de texture $s's'_2s'_3$, nous approximons le sommet s'_0 par s''_0. Comme le décodeur est capable de réaliser la même prédiction durant l'expansion de sommet, il n'est plus nécessaire que de coder l'erreur effectuée $\overrightarrow{e} = \overrightarrow{s''_0s'_0}$. Le vecteur de déplacement $\overrightarrow{u} = \overrightarrow{s's'_0}$ pourra ainsi être retrouvé de la façon suivante : $\overrightarrow{u} = \overrightarrow{s's''_0} + \overrightarrow{e}$. Dans ce contexte, plus les triangles ss_2s_3 et $s's'_2s'_3$ sont similaires, meilleure sera la prédiction. Une prédiction précise entraînera des vecteurs erreurs petits et donc

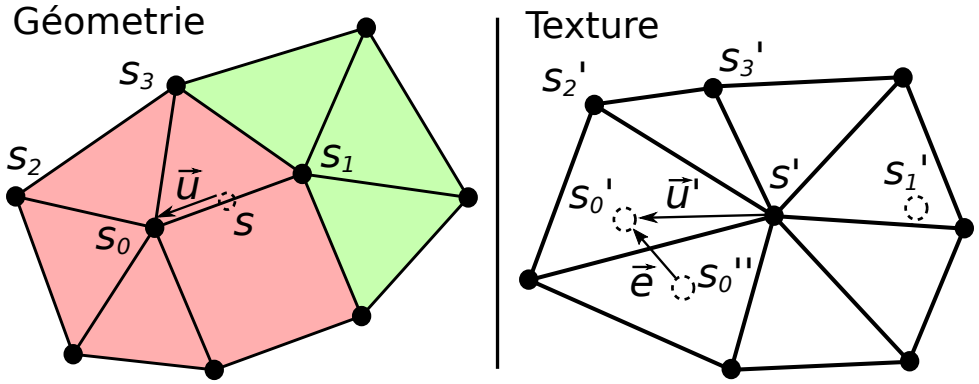


FIGURE 3.14 – Sur la gauche, un patch 3D montrant le déplacement opéré pour passer de s à s_0 lors de l'expansion de s. Sur la droite, la paramétrisation correspondante au patch 3D montrant la prédiction de s_0'' et l'erreur réalisée par cette dernière.

une faible entropie. Cela améliorera le taux de compression.

Finalement, pour toutes les faces triangulaires $s_0 s_1 s_i$ incidentes à $s_0 s_1$, qui sont supprimées lors de la contraction, nous devons restituer le *corner* c_i (s_i, $s_0 s_1 s_i$). Dans la grande majorité des cas, si $s_0 s_1 s_i$ possède une face adjacente f associée à la même région, alors c_i pourra être déduit de c_i' (s_i, f), sans surcoût. Dans le cas contraire, nous utilisons la similarité décrite précédemment pour prédire les coordonnées UV en fonction de $s_0 s_1 s_i$. Seul un vecteur erreur est fourni, permettant au décodeur de résoudre le *corner* c_i.

3.7.4 Vague de décimation

Si le codage de chaque simplification unitaire peut être réalisé comme nous le décrivons dans les sections précédentes, cela ne suffit pas. En effet, le décodeur aura beau recevoir des informations de raffinement, il ne saura pas à quel sommet les appliquer, lors de la décompression.

Pour remédier à cela, nous proposons d'ordonner, au sein d'une vague de décimation, les différentes contractions d'arêtes réalisées. Afin de maintenir cet ordre, nous nous appuyons sur une structure particulière : l'arbre couvrant. Appliqué à un maillage $\mathcal{M}$, un arbre couvrant est un ensemble d'arêtes connexes construit pour parcourir tous les sommets de $\mathcal{M}$. L'utilisation d'un arbre couvrant garantit que tous les sommets du maillage seront traités.

Pajarola et Rossignac [PR00] ont déjà présenté l'efficacité des arbres couvrants dans le processus de codage d'un maillage. Pour notre part, nous les manipulerons différemment tout en cherchant les mêmes avantages. Effectivement, pouvoir ordonner les informations de raffinement suivant un ordre déterministe compréhensible par le décodeur permet de ne pas transmettre explicitement les références de l'ensemble des sommets à expandre. Le choix des sites d'expansion devient alors implicite, ce qui produit un gain significatif en terme de taux de compression.

Avant tout, il est important de préciser que la construction d'un nouvel arbre couvrant est réalisée à la fin de chaque vague de décimation. Il est nécessaire que chaque vague ait un arbre propre à cause de l'ordre dans lequel celui-ci est construit, comme nous l'expliquons plus loin.

Nous nous appuyons également sur un point essentiel. Lors de la compression, le maillage courant $\mathcal{M}^i$ sera décimé en $\mathcal{M}^{i-1}$. Au moment de la construction de l'arbre couvrant, la connectivité et la géométrie à disposition sera donc celle de $\mathcal{M}^{i-1}$. Ce sera donc exactement la même configuration disponible pour le décodeur, à la décompression, avant qu'il n'applique les infor-

mations de raffinement permettant de restituer $\mathcal{M}^i$.

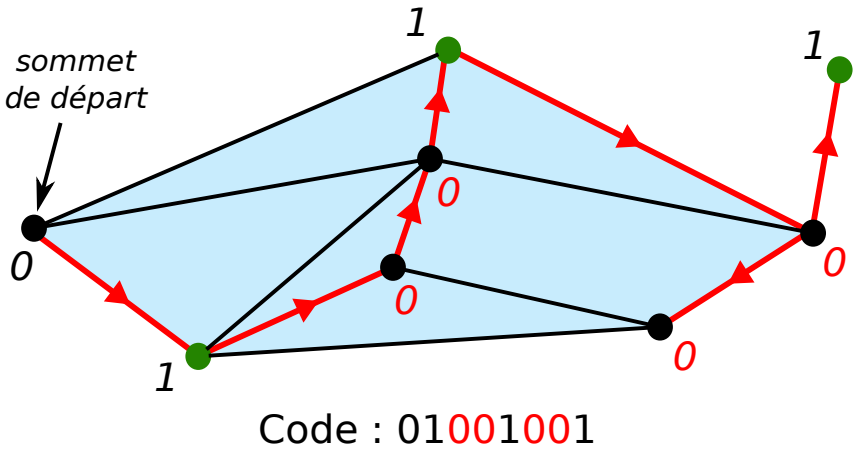


FIGURE 3.15 – Exemple de construction d'un arbre couvrant. Les arêtes rouges orientées décrivent le sens de construction de l'arbre, du sommet de départ jusqu'aux feuilles. Les sommets verts représentent des sommets résultants d'une contraction d'arête. Les codes rouges ne sont pas générés lorsque l'optimisation du codage de l'arbre est appliquée.

Comme nous l'avons déjà vu, un maillage peut contenir plusieurs composantes connexes. Dans ce cas, pour chaque vague, nous construisons un arbre couvrant par composante connexe. La construction de tels arbres commence par un sommet. Ce sommet est nommé *sommet de départ* de l'arbre couvrant (voir figure 3.15). Pour une composante connexe donnée, le sommet de départ est choisi totalement arbitrairement mais demeurera le même durant toute la compression. Le choix de ce sommet peut influencer, de la même manière que la méthode de construction employée, le codage de l'arbre. Néanmoins, il serait très coûteux de déterminer le meilleur sommet de départ, pour un gain très faible. De plus, le choix d'un sommet différent pour chaque vague, afin d'optimiser le codage de chaque arbre individuellement pourrait servir notre méthode. Malheureusement, cela implique de fournir l'ensemble des références des sommets de départ de chaque arbre couvrant, ce qui diminue le gain potentiel en terme de compression.

Une fois le sommet de départ choisi, il doit demeurer présent tout au long de la compression, jusqu'à atteindre $\mathcal{M}^0$. Cela signifie qu'il faut empêcher sa suppression. Interdire la contraction de ses arêtes incidentes biaiserait la simplification. Nous proposons donc un mécanisme simple. Lorsque nous procédons à la contraction d'une arête s_0s_1, nous vérifions si l'un de ses sommets est un sommet de départ. Si c'est le cas, nous nous assurons que le sommet de départ est s_0. Pour ce faire, il suffit simplement d'inverser l'ordre des sommets passés à notre algorithme de contraction d'arête, si nécessaire. Ce mécanisme permet au codeur, mais également au décodeur, de maintenir à jour le sommet de départ de manière déterministe.

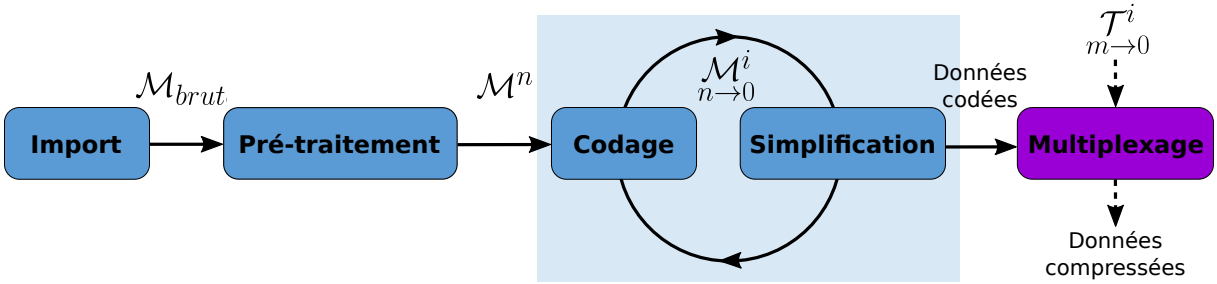
À partir d'un sommet de départ donné, la construction de l'arbre couvrant peut commencer. Pendant le processus, chaque sommet appartenant à l'arbre sera marqué spécifiquement. Nous nous servons également d'une pile de sommet, notée P. Pour commencer, nous empilons le sommet de départ. Puis, en prenant le premier sommet de la pile, nous empilons ses sommets adjacents qui n'appartiennent pas encore à l'arbre dans la pile de construction dans un ordre précis. Cet ordre est l'ordre naturel 3D, c'est à dire qu'ils sont triés par leurs coordonnées. Nous dépilons ensuite le sommet courant et recommençons jusqu'à ce que la pile soit à nouveau vide. Cette opération est répétée pour chaque composante connexe.

Il n'est pas nécessaire de construire physiquement l'arbre couvrant. Seul l'ordre des sommets importe. Il est donc possible de coder l'arbre à la volée sans avoir recours à une structure particulière. Ainsi, pour chaque sommet s dépilé, lors de la construction, nous détectons deux cas.

Si s n'est pas un sommet résultant d'une contraction survenue durant la vague précédente, nous concaténons un code 0 à la description de l'arbre couvrant. Dans le cas contraire, nous concaténons un code 1 à la description de l'arbre, puis nous concaténons les $inf(s)$ à la description de la vague de décimation. Les deux descriptions sont séparées. La Section 3.8 explique l'ordre de compression des différents codes générés.

L'ordre de décimation des arêtes n'est pas le même que l'ordre de codage de ces simplifications. Pour cette raison, il est nécessaire que chaque contraction soit indépendante, afin d'éviter tout conflit. En pratique, cela signifie que deux sommets résultant ne peuvent pas être adjacents. Cette contrainte pénalise mécaniquement le taux de compression en limitant le nombre de contractions par vague mais nuit également à la qualité visuelle du maillage en invalidant des arêtes dont la décimation serait bénéfique. En contrepartie, il est possible d'en tirer des bénéfices. Cette contrainte permet au décodeur de déterminer implicitement le code de certains sommets. Ainsi, durant la décompression, lorsqu'un sommet s est adjacent à un sommet s', déjà détecté comme résultant d'une contraction (code 1), alors s est obligatoirement non résultant (code 0). Ainsi, à la compression, si cette situation apparaît, nous ne concaténons pas de code 0. Cette optimisation permet de réduire le nombre de codes 0 générés pour la description des différents arbres de près de 50%. En ce qui concerne nos expérimentations, cela représente une réduction de 0.8 bits par sommet, en moyenne, au niveau du flux compressé.

3.8 Multiplexage des informations



Chaque information récupérée pendant la simplification, qu'elle porte sur la géométrie, sur la connectivité ou sur la texture, est concaténée dans un ordre défini et stockée, par vague, jusqu'à la fin du processus. La simplification se termine lorsque le maillage courant n'est plus constitué que par des sommets isolés. Ces sommets sont les sommets de départ utilisés pour la construction des différents arbres couvrants vus précédemment. Par définition, le nombre de sommets restants est donc égal au nombre de composantes connexes du maillage $\mathcal{M}^n$. À ce moment, la phase de codage arithmétique des données encodées peut commencer. Nous commençons donc la construction d'un fichier comprimé qui peut être vu comme un flux à transmettre au décodeur afin qu'il reconstitue le maillage (voir figure 3.16). Tout d'abord, nous écrivons une entête contenant certaines informations essentielles.

Un bit est utilisé pour spécifier si le maillage est texturé ou non. Si ce n'est pas le cas, toutes les informations relatives à la texture mentionnées par la suite sont omises. Puis, nous indiquons le nombre de bits sur lesquels sont quantifiées les coordonnées des sommets du maillage, ainsi que le nombre de bits sur lesquels sont quantifiées les coordonnées de texture. Nous écrivons également les coordonnées du sommet minimum et du sommet maximum de la boîte englobante du maillage. Les coordonnées de texture étant naturellement bornées entre 0 et 1, il n'est pas nécessaire de décrire leur boîte englobante. Ces informations permettront au décodeur de restituer le maillage dans l'espace cartésien d'origine et de restituer les bonnes coordonnées de texture.

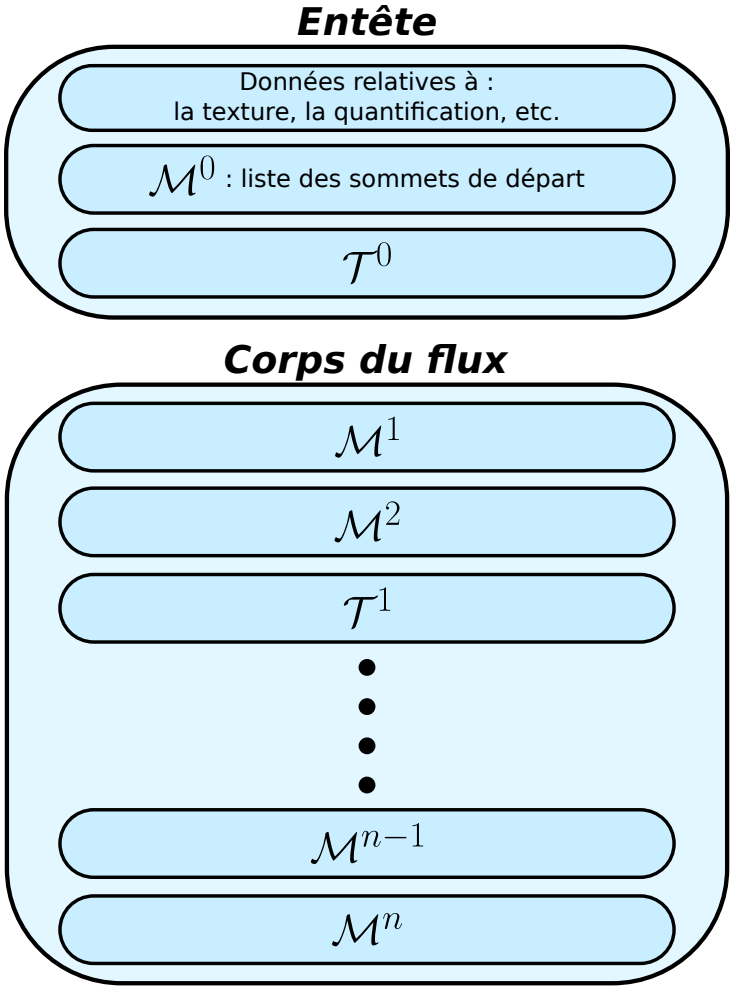


FIGURE 3.16 – Schéma représentant les différentes parties du flux de données reçu par le décodeur. Le corps est composé d'un multiplexage de paquets, relatifs au maillage et à l'image de texture, guidé par la métrique perceptuelle MS-SSIM.

Ensuite, le nombre de sommets restants est transmis. Celui-ci permettra de connaître le nombre de coordonnées à lire pour reconstruire $\mathcal{M}^0$. Ainsi, la suite de l'entête est composée des trois coordonnées (x, y et z) de chaque sommet de $\mathcal{M}^0$. Afin de réduire au maximum la taille de ces coordonnées, elles sont compressées à l'aide d'un codeur arithmétique. En pratique, le nombre de sommets restants est souvent faible, mais lorsque le nombre de composantes connexes est important, une telle compression est justifiée.

La suite du fichier, ou corps, est composée de paquets. Un paquet représentant l'ensemble des informations nécessaires pour reconstruire le LoD suivant. Pour chaque paquet, nous utilisons le même codeur arithmétique que pour l'entête. Cependant, nous utilisons un contexte différent pour chaque type de données. En effet, au début d'un paquet permettant de reconstruire un LoD supplémentaire du maillage, nous compressons les bits de construction du(des) arbre(s) couvrant(s). Un contexte est dédié à cette information. Ensuite, nous compressons l'ensemble des informations de contraction de la vague correspondante. Pour les informations d'une contraction, quatre contextes sont employés : un pour la géométrie (en première position), un pour la connectivité et l'orientation (en deuxième position) et deux pour l'attribut de texture associé (un pour les indices de régions et un pour les vecteurs de déplacement). L'utilisation de plusieurs contextes permet de traiter au mieux le flux car chaque type de données a une structure et des valeurs différentes. Les contextes du codeur arithmétique sont, ici, adaptatifs et évoluent donc

tout au long de la compression.

Pour le moment, nous n'avons pas décrit par quels moyens nous transmettions l'image de texture dans ce flux. De façon naïve, il serait possible de compresser l'image, dans un format standard (par exemple, JPG), puis de l'insérer juste après l'entête. De cette façon, les tous premiers LoDs pourraient déjà bénéficier de cette texture. Cependant, la texture représente malheureusement souvent plus de données que l'ensemble des paquets compressés qui la succèdent. Cela signifie qu'il faudrait, dans ces cas là, transmettre plus de la moitié du flux avant de pouvoir obtenir le premier LoD. Il ne serait alors plus vraiment question de compression *progressive*.

Afin de remédier à ce problème, nous avons décidé de traiter l'image de texture de manière progressive également. Plusieurs travaux existants [TA04, YSK04, YLK04] proposent de compresser progressivement la texture, ce qui revient à produire différents LoDs, tout comme pour le maillage lui-même. Les données représentant chaque LoDs de texture sont alors intercalées entre les données représentant chaque LoDs du maillage. Ce multiplexage est guidé afin d'optimiser le rapport débit-distorsion durant la décompression. En étudiant les différentes approches déjà proposées, nous avons estimé que, dans le cadre de notre compression progressive, nous pouvons améliorer le guidage de ce multiplexage. La suite de cette partie explique la stratégie mise en place.

Dans un premier temps, nous compressons progressivement l'image de texture à l'aide du format JPG-progressif. Ce format nous permet d'utiliser un script afin de définir les différents découpages de la texture en LoDs. Chaque LoD est alors représenté par un paquet de données. Nous avons volontairement choisi de décomposer la texture en une trentaine de paquets de façon à ce que leurs tailles soient régulièrement croissantes.

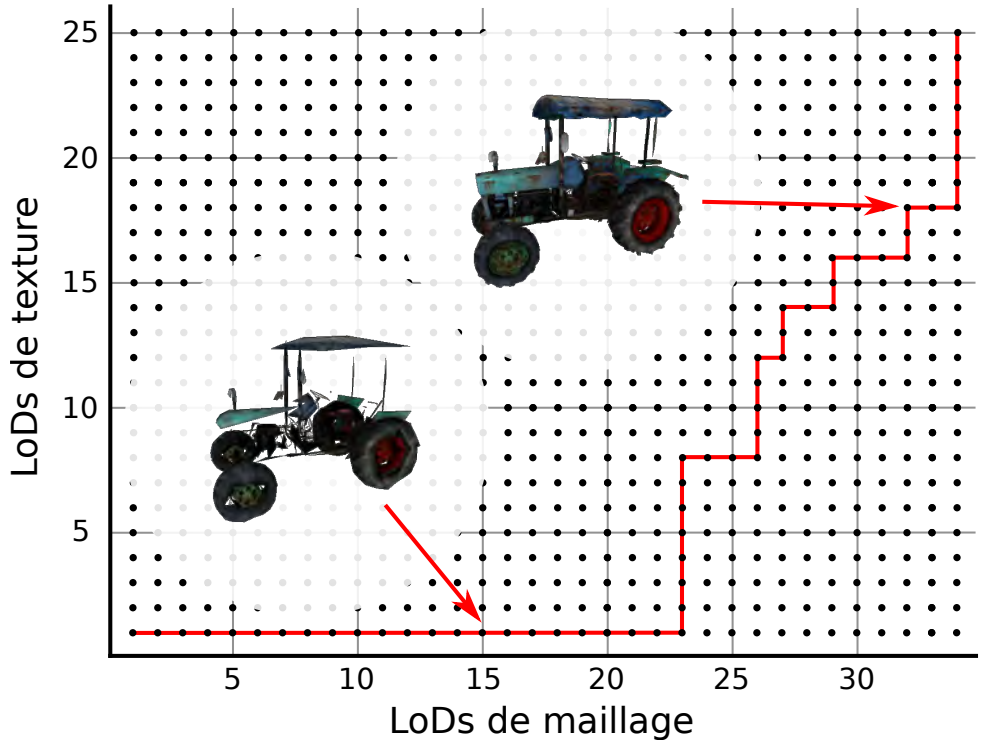


FIGURE 3.17 – Construction du chemin de multiplexage (en rouge) pour le modèle *Tractor*. Chaque point du graphique représente une combinaison entre un LoD de maillage et un LoD de texture.

La suite consiste à déterminer, après l'écriture de l'entête, s'il faut ajouter un paquet relatif au maillage ou un paquet relatif à la texture. Pour cela, nous utilisons la métrique perceptuelle

MS-SSIM [WSB03b]. Cette métrique a démontré qu'elle offrait une bonne prédiction de la qualité visuelle d'une image 2D par rapport à une autre. [LLV16] a également montré qu'elle était une des métriques les plus adaptées pour le calcul de la distorsion d'une maillage. MS-SSIM nous permet, pour un LoDs de maillage associé à un LoDs de texture, de déterminer un score en fonction du maillage d'origine. Pour ce calcul, nous avons repris l'approche de [LLV16]. Elle consiste à prendre un nombre de 42 vues 2D autour de la combinaison en question, puis du maillage $\mathcal{M}^n$. Chaque point de vue est choisi de façon uniforme sur une sphère centrée sur le maillage, en considérant une illumination indirecte dont la source est située en haut à gauche du maillage. Nous retenons la valeur Q_{ij} comme étant la moyenne des valeurs MS-SSIM de chaque prise de vue pour la combinaison $\mathcal{C}_{ij}$ ($\mathcal{M}^i$ associé à la texture $\mathcal{T}^j$).

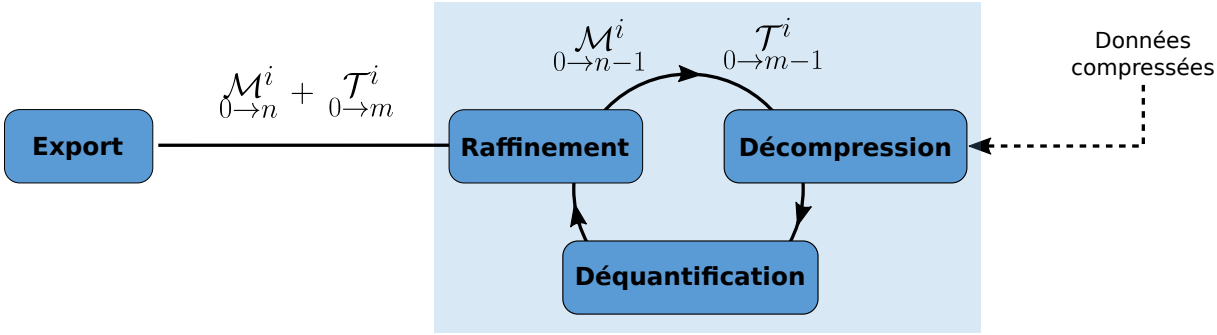
Ainsi, afin de choisir le multiplexage optimal pour aller de $\mathcal{M}^0$ associé à $\mathcal{T}^0$ (le plus faible LoD de texture) à $\mathcal{M}^n$ associé à $\mathcal{T}^m$ (la texture originale), nous comparons les valeurs Q de chaque combinaison (voir figure 3.17). Pour optimiser le processus, nous itérons sur les étapes suivantes : à partir de la combinaison courante $\mathcal{C}_{ij}$ ($i < n$ et $j < m$), nous calculons Q_{i+1j} et Q_{ij+1} ainsi que la taille T_{i+1j} et T_{ij+1} des paquets correspondants ; puis nous comparons les rapports $\frac{Q_{i+1j}}{T_{i+1j}}$ et $\frac{Q_{ij+1}}{T_{ij+1}}$; nous choisissons la combinaison avec le plus grand rapport ; elle devient la combinaison courante. Si, pour une combinaison $\mathcal{C}_{ij}$, $i = n$, alors il ne reste plus de LoD de maillage à multiplexer, nous ajoutons donc directement le reste des LoDs de texture. Si $j = m$, de la même façon, nous ajoutons directement le reste des LoDs de maillage. C'est avec cette démarche que le chemin optimal (en rouge dans la figure 3.17) est construit, bifurquant lorsqu'il est préférable de raffiner la géométrie plutôt que la texture, et *vice versa*. Nous pouvons constater l'apport de ce multiplexage grâce à la figure 3.18, qui nous montre trois combinaisons différentes : un maillage très raffiné associé à une texture grossière, un maillage grossier associé à une texture très raffinée et enfin une combinaison choisie par notre méthode (faisant partie du chemin optimal). Pour une même taille de données décompressées, les deux premières approches souffrent visuellement du manque d'information, soit au niveau de la géométrie, soit au niveau de la texture. En revanche, le choix fait par notre métrique offre un compromis satisfaisant en équilibrant l'apport visuel des deux types d'informations.

FIGURE 3.18 – Vue de trois combinaisons maillage-texture possibles, proches en terme de taille, avec le modèle *Tiger Fighter*. À gauche, le maillage est très raffiné tandis que la texture est grossière. Au centre, le maillage est grossier et la texture très raffinée. À droite, notre méthode choisit un compromis géométrie-texture pour optimiser la qualité visuelle.

Ce parcours permet d'optimiser la qualité visuelle de chaque niveau de détails issu de la décompression, en fonction de la métrique MS-SSIM. En pratique, cela détermine quel type de paquet doit être ajouté au flux. Le décodeur ne peut pas déterminer implicitement ce choix.

Il est donc nécessaire d'ajouter un bit supplémentaire au début de chaque paquet (0 pour un paquet de maillage, 1 pour un paquet de texture). Comme les deux premiers paquets transmis sont forcément un paquet de maillage suivi d'un paquet de texture (voir figure 3.16), les deux premiers bits sont omis. Cela occasionne donc un surcoût maximum de $m + n - 1$ bits. Il serait théoriquement possible de regrouper ces bits dans l'entête pour les compresser et ainsi réduire significativement ce surcoût. Néanmoins, nous avons choisi de ne pas surcharger l'entête pour obtenir un début de transmission le plus léger possible.

3.9 Décompression



Durant la décompression, le décodeur lit progressivement le flux transmis pour reconstruire étape par étape le maillage. Cette lecture passe par différentes phases successives. Premièrement, le décodeur reçoit l'entête. Cela lui permet de savoir si le maillage sera accompagné d'une texture. Dans le cas contraire, le traitement des informations relatives à l'image et aux coordonnées de texture sera omis. L'entête décrit également les paramètres de quantification. Ceux-ci permettront au décodeur de retranscrire les nouvelles coordonnées déterminées directement dans l'espace cartésien d'origine. Celui-ci pourra également construire $\mathcal{M}^0$ avec, si nécessaire, la texture $\mathcal{T}^0$ associée.

Pour la suite du processus, le décodeur devra lire le prochain bit pour déterminer si le paquet courant décrit un raffinement du maillage (0) ou de la texture (1). Dans le cas d'un raffinement de la texture, le paquet sera simplement concaténé à l'image de texture actuelle pour obtenir un LoD plus détaillé (sous le format JPG-progressif). Pour un raffinement de maillage, le décodeur devra suivre l'enchainement énoncé plus haut. Tout d'abord, nous utilisons le codeur arithmétique, avec le contexte approprié, pour décompresser les informations de construction de(des) arbre(s) couvrant(s). Étant donné que nous construisons le ou les arbres pendant cette décompression, nous connaissons naturellement la fin de cette section car elle coïncide avec le moment où la totalité des sommets du maillage a été visitée. Si un sommet a pour code 1, nous l'ajoutons à la file des sommets à expandre pendant la vague courante. Cette stratégie indique alors l'ordre dans lequel les sommets seront traités. Si, pendant la construction, le sommet courant est adjacent à un sommet déjà ajouté à la file (nous marquons spécifiquement ces derniers à des fins d'optimisation) alors il sera implicitement associé à un code 0. Si le sommet courant a pour code 0, nous passons directement au suivant dans l'ordre de construction.

Une fois la construction de l'arbre (ou des arbres) achevée et la file de sommets à traiter remplie, nous procédons aux expansions. Les paragraphes suivants décrivent notre opérateur générique d'expansion de sommets. Celui-ci est une généralisation de l'opérateur de Popović et Hoppe [PH97]. Il permet d'expandre des sommets situés sur un patch 2-variété ou non-variété, triangulaire ou polygonal, de façon générique.

Pour chaque sommet s de la file, un vecteur de déplacement est décomprimé par le codeur arithmétique à l'aide du contexte approprié. Un repère de Frenet est construit sur le patch de s de la même manière que lors de la compression. Le vecteur de déplacement $\overrightarrow{u}$ exprimé dans le repère de Frenet est alors déterminé dans le repère global. Nous pouvons ainsi positionner les nouveaux sommets s_0 et s_1 tels que $s_0 = \frac{\overrightarrow{u}}{2}s$ et $s_1 = -\frac{\overrightarrow{u}}{2}s$. À ce moment là, il faut compenser l'arrondi réalisé lors de la simplification pour retrouver les coordonnées exactes.

Tout comme l'information géométrique, les données permettant de retrouver la connectivité originelle du patch sont décomprimées à l'aide du contexte approprié. Néanmoins, pour cette partie, le flux doit être décomprimé code par code car la séquence en question dépend de chaque patch. Nous trions les arêtes incidentes de s dans le même ordre déterministe que lors de la simplification (ordre naturel dans l'espace 3D), pour que chacune corresponde à son code. Le but ici est de redistribuer les arêtes incidentes à s sur les nouveaux sommets s_0 et s_1, mais également, par extension, les faces incidentes.

La gestion des coordonnées de texture est réalisée en parallèle de celle de la connectivité pour une raison simple. D'une part, il est important pour le décodeur de connaître la configuration de la paramétrisation avant l'expansion pour qu'elle corresponde à celle estimée par le codeur, durant la simplification. D'autre part, la nouvelle paramétrisation ne peut être déterminée tant que la nouvelle géométrie n'a pas été résolue. Ainsi, comme lors de la simplification, les différentes régions de texture du patch sont détectées pour connaître les *corners* appariables ou non-appariables. Ensuite, pour chaque face, un indice de région est décompressé grâce au contexte approprié et, pour chaque région détectée (plus celles décrites comme nouvelles par les indices de région), un autre contexte permet de décompresser un vecteur erreur. Ces deux informations permettent de retrouver les coordonnées de textures des *corners* associés à s_0 et à s_1. Néanmoins, pour toute nouvelle face $s_0s_1s_2$, il est également nécessaire de résoudre le *corner* (s_2, $s_0s_1s_2$). Comme expliqué dans la Section 3.7.3, s'il est possible de trouver une autre face incidente à s_2 étant associée à la même région (ici, par forcément à l'intérieur du patch), trouver les coordonnées UV de ce *corner* est alors direct. Sinon, il faut décompresser à nouveau un vecteur erreur. Tous les vecteurs erreurs sont utilisés avec la prédiction basée géométrie décrite dans la Section 3.7.3 pour créer ou mettre à jour les *corners* du patch avec les coordonnées UV originelles.

Les informations d'expansion étant lues code par code en fonction du patch, une fois toutes les étapes citées plus haut réalisées, le décodeur passe au prochain sommet à expandre dans la file. Cela continue jusqu'à se que la file soit vide. Le paquet est ainsi entièrement lu et la vague de raffinement est terminée. Le décodeur peut s'arrêter définitivement à la fin de chaque paquet, qu'il soit relatif à la texture ou au maillage. Cela permet donc théoriquement d'arrêter la décompression en fonction de diverses caractéristiques du périphérique de réception. Si toutefois la décompression était menée à son terme, un code de fin de flux est présent après le dernier paquet pour notifier la fin de transmission au décodeur.

Chapitre 4

Expérimentations et Résultats

Afin de valider nos travaux, nous proposons dans ce chapitre un certain nombre de résultats. Ces derniers permettent de prouver de façon quantitative et qualitative l'efficacité de notre approche. Nous avons également comparé les performances de notre approche à d'autres méthodes existantes pour mieux nous situer dans l'état de l'art. Nous avons choisi de présenter ces résultats dans un ordre induit par la méthode elle-même.

Notre méthode de compression progressive appliquée aux maillages surfaciques polygonaux non-variétés se décompose en plusieurs étapes. Le maillage $\mathcal{M}^n$ est simplifié itérativement durant la compression. Afin de minimiser la distorsion engendrée, à cause de la décimation, sur les LoDs successifs, nous avons mis en place une file qui détermine quelles arêtes doivent être contractées en priorité. Au sein de la file, les arêtes sont triées par leurs poids. Nous avons proposé plusieurs métriques permettant de calculer ces poids. Nous illustrons, dans la Section 4.1, les performances de ces différentes stratégies de sélection d'arêtes servant à contrôler la distorsion.

Les arêtes sont alors contractées, dans l'ordre choisi, par vague. Le regroupement de ces contractions au sein d'un parcours déterministe permet de connaître l'emplacement des sommets à expandre, lors de la décompression, pour un coût minime. Cependant, la taille de ces vagues de décimation a une influence sur le taux de compression et sur la qualité visuelle des LoDs produits. Nous justifions donc, dans la Section 4.2, notre choix d'un seuil dynamique appliqué à la file de priorité des arêtes en confrontant plusieurs approches possibles.

Avant chaque contraction d'arête, il est essentiel d'encoder les informations qui permettront au décodeur d'inverser la simplification. Le codage d'un ou de plusieurs arbres couvrants par vague est également nécessaire pour savoir à quel sommet appliquer ces informations de raffinement. Grâce à différentes stratégies de prédiction et d'optimisation entropique, nous proposons un encodage efficace de la géométrie, de la connectivité et des coordonnées de texture. Nous comparons nos travaux à l'état de l'art en terme de performance de compression pour la connectivité et la géométrie (voir Section 4.3), et la paramétrisation de texture (voir Section 4.4). La plupart des modèles utilisés lors de nos expérimentations sont illustrés dans les figures 4.3 et 4.1.

Afin de présenter une méthode de compression totalement progressive, nous avons mis en place une stratégie de multiplexage des informations de maillage et de texture. Pour valider cette dernière approche, des résultats quantitatifs et qualitatifs sont finalement présentés dans la Section 4.5. Ils sont accompagnés de comparaisons avec certaines méthodes de l'état de l'art.

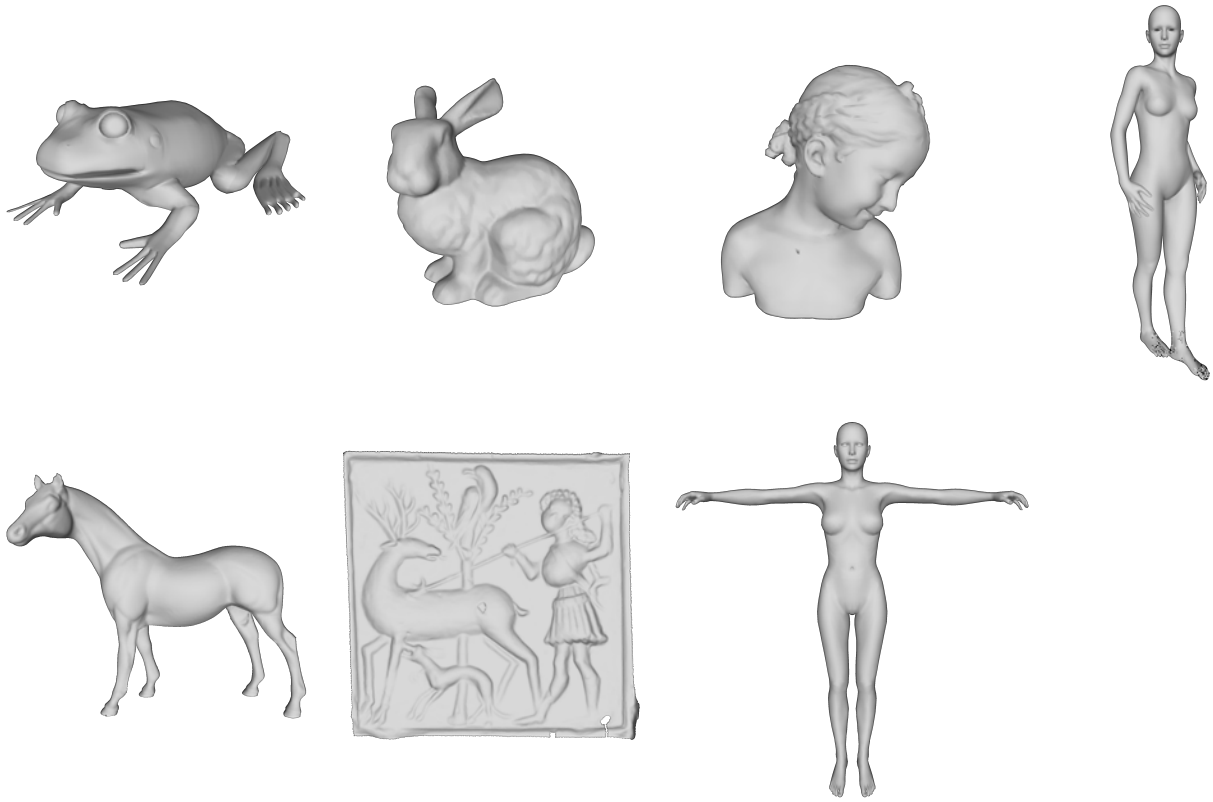


FIGURE 4.1 – Modèles 3D avec paramétrisation présents dans le tableau 4.3. De gauche à droite et de haut en bas : *Frog*, *Bunny*, *Bimba*, *Fiery*, *Horse*, *Kachel*, *Victoria*.

4.1 Métrique de sélection d'arêtes

Dans le but de valider notre stratégie de sélection de sites de contraction, présentée en Section 3.5, nous présentons les courbes de débit-distorsion obtenues avec trois métriques différentes (voir figure 4.2). Étant donné que le critère de longueur d'arête ne suffit pas à limiter convenablement la distorsion de maillages texturés, nous avons choisi de nous attarder sur les autres métriques détaillées dans la Section 3.5. Tout d'abord, la courbe "Hausdorff" correspond au critère p_{haus}. Celui-ci correspond au calcul de la distance de Hausdorff accompagné d'ajustements en fonction de la connectivité autour de l'arête concernée. Ensuite vient la courbe "Texture", obtenue grâce au critère p_{text} permettant de contrôler la distorsion engendrée au niveau de la paramétrisation de la texture. Et enfin, la courbe "Hausdorff + Texture" correspond à la combinaison linéaire p_{comb} des deux métriques précédentes. Pour tester les performances de ces différents critères, nous mesurons la qualité visuelle du maillage durant sa décompression. Plus vite le maillage courant devient proche du maillage original durant la décompression, meilleur est le critère de sélection, car l'avantage souhaité de notre méthode est d'obtenir une version fidèle du maillage original en utilisant le moins d'information possible. Ici, nous quantifions cette qualité visuelle à l'aide de la métrique perceptuelle MS-SSIM. Pour cette expérimentation, la texture associée à chaque LoD durant la décompression se trouve en pleine résolution.

Ainsi, la figure 4.2 montre clairement que le critère p_{text} est insuffisant pour atteindre notre objectif. En effet, le caractère géométrique étant totalement ignoré, cela pénalise grandement la qualité globale du maillage. Cependant, en associant cette mesure à p_{haus}, nous montrons que l'on obtient de meilleurs résultats que lors de l'utilisation de p_{haus} seul. Nous concluons donc assez directement que si le contrôle de géométrie du maillage est essentiel pour obtenir une bonne qualité visuelle, la conservation de l'information de texture permet également de l'améliorer de

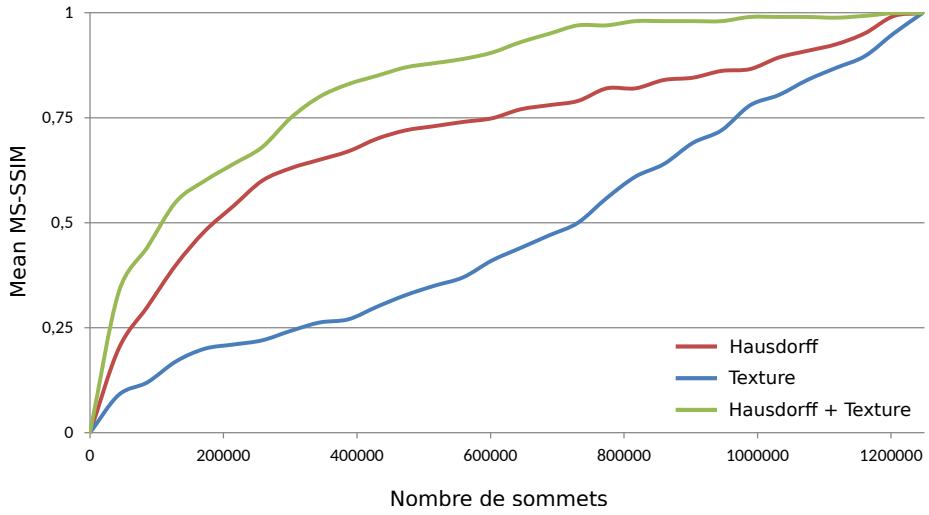


FIGURE 4.2 – Courbes de débit-distorsion obtenues avec le modèle *Dwarf* pour différentes métriques de sélection d'arêtes : p_{haus} (Hausdorff), p_{text} (Texture) et p_{comb} (Hausdorff + Texture). La qualité visuelle du maillage est mesurée à l'aide de la métrique perceptuelle MS-SSIM (plus l'indice se rapproche de 1 et meilleure est la qualité visuelle).

façon significative. Nous avons ainsi estimé qu'un compromis entre ces deux types de données était probablement le meilleur choix à faire, dans notre cas.

4.2 Taille des vagues de décimation

Comme nous l'avons vu dans la Section 3.6, le nombre d'arêtes contractées par vague durant la simplification joue un rôle important, à la fois au niveau de la qualité visuelle des LoDs créés mais également vis à vis du taux de compression du maillage. Afin de déterminer l'influence de la taille de chaque vague sur notre méthode, nous étudions les distances MRMS de chaque LoDs lors de la décompression. Ici, seule la distorsion géométrique est prise en compte pour mieux cerner cet impact. La figure 4.4 illustre les courbes de débit-distorsion pendant la décompression du modèle *Tiger Fighter*. Trois stratégies sont illustrées :

— Une approche prenant en compte une taille minimale pour chaque vague. Cela correspond, dans notre cas, à une seule et unique contraction d'arête par itération. La mise à jour de la file de priorité à la fin de chaque vague garantit une qualité optimale par rapport à la métrique de sélection employée. Il faut noter que c'est également l'approche que choisi Hoppe et Popović dans [Hop96] et [PH97]. Néanmoins, dans le cadre de notre méthode, cela augmente de manière drastique le nombre de vagues et donc le nombre de paquets. Cette augmentation pénalise logiquement le taux de compression même si l'entropie de l'information relative à l'arbre couvrant devient mécaniquement plus faible.
— Le choix de contracter le maximum d'arêtes, à chaque vague, i.e. de décimer toutes les arêtes contractables de la file de priorité. Cette stratégie correspond d'avantage à celle employé par Alliez et Desbrun [AD01a]. Ceci permet de diminuer au maximum la taille des informations liées aux arbres couvrants. Toutefois, dans ce cas, la gestion d'une file de priorité n'a plus aucun sens et le contrôle de la distorsion devient nul. Il en résulte une moins bonne qualité visuelle des LoDs durant la décompression.
— Notre stratégie qui consiste à calculer une taille adaptée pour chaque vague. Le nombre

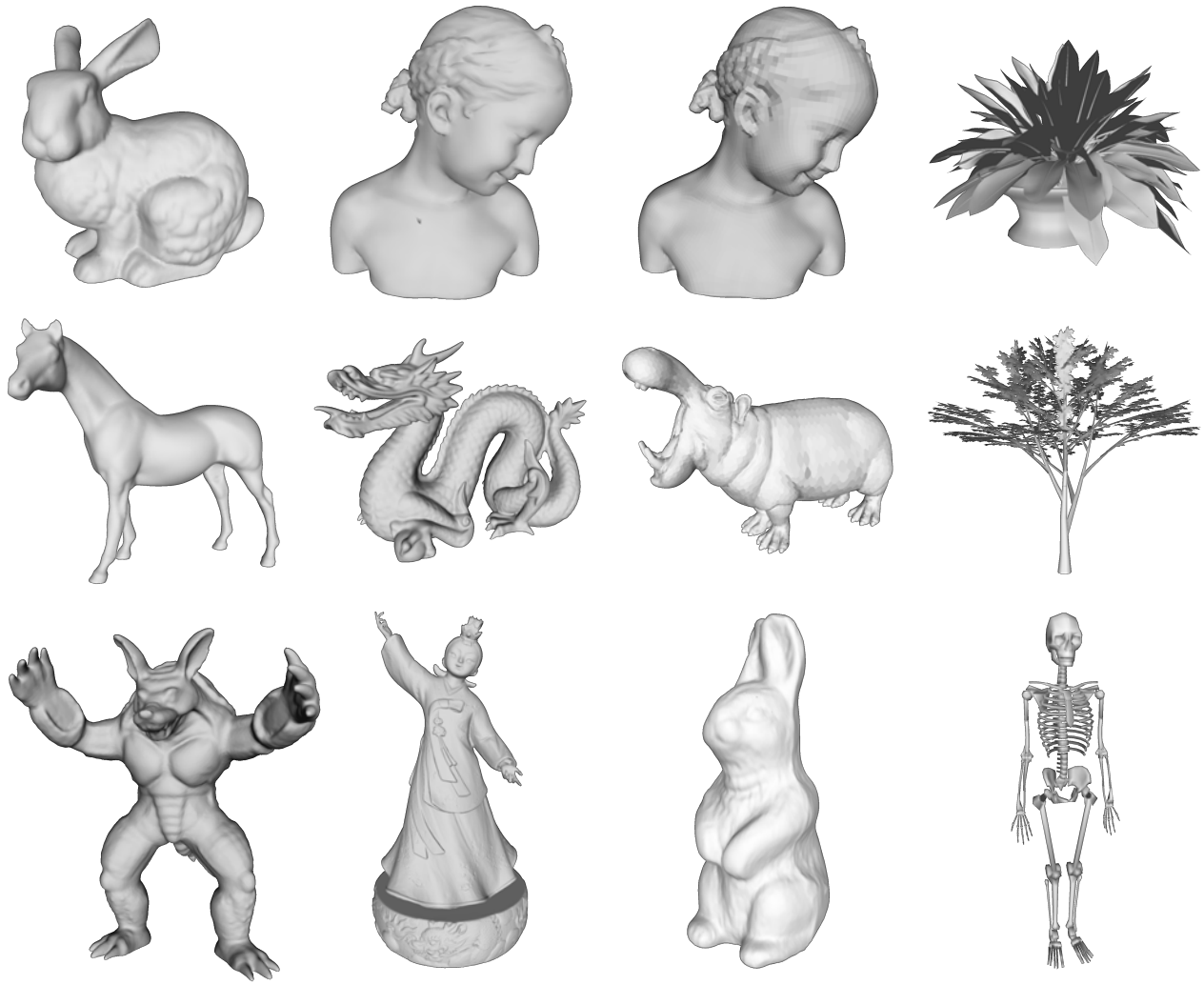


FIGURE 4.3 – Modèles 3D présents dans les tableaux 4.1 et 4.2. De gauche à droite et de haut en bas : *Bunny*, *Bimba*, *Bimba-q*, *House Plant*, *Horse*, *Dragon*, *Hippo*, *Maple*, *Armadillo*, *Dancing Ari*, *Rabbit*, *Skeleton*.

d'arêtes décimées durant une itération est limité par un seuil calculé sur les poids des arêtes de la file de priorité. Cette flexibilité permet d'obtenir des vagues de taille raisonnable tout en limitant la distorsion.

Dans la figure 4.4, nous constatons que le seuil dynamique, que nous proposons pour déterminer la taille de chaque vague, permet, dans une certaine mesure, de limiter les différents aspects négatifs évoqués. En effet, la courbe débit-distorsion associée à cette stratégie se rapproche, à bas débit, de celle liée à la création de vagues de taille minimale. Cela signifie que l'on retrouve une très bonne qualité visuelle dans les premiers niveaux de détails. De plus, nous obtenons, un taux de compression total du maillage proche de la stratégie de taille maximale de vague. Ainsi, l'optimisation de la taille de chaque vague permet de réaliser un maximum de contractions d'arêtes tant que celles-ci n'engendre qu'une faible distorsion. Cette stratégie représente donc un compromis avantageux entre les deux autres approches et offre une réponse adaptée à tout maillage surfacique, mais également à chacun de ces LoDs durant la simplification.

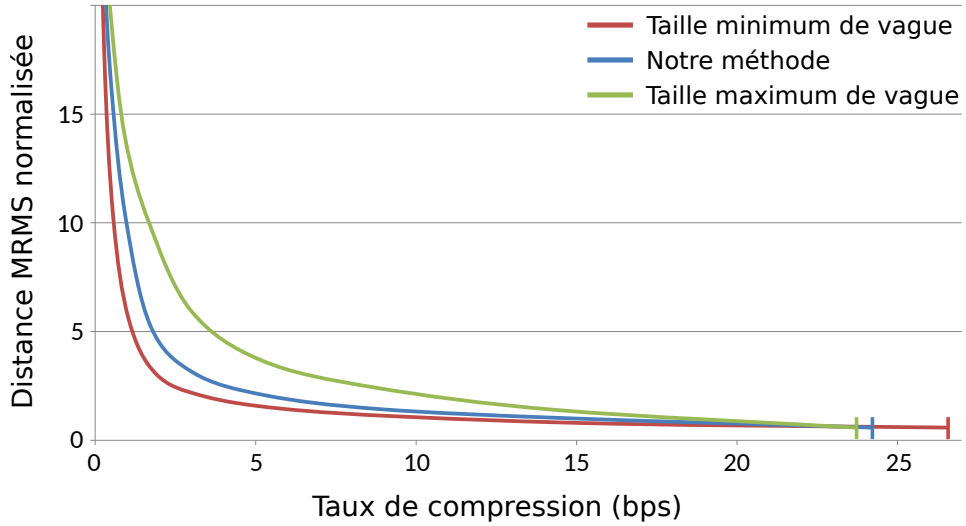


FIGURE 4.4 – Courbes de débit-distorsion obtenues avec le modèle *Tiger Fighter* pour différentes stratégies de détermination de la taille des vagues de décimation : La taille minimum (une contraction par vague), la taille maximum (toutes les contractions possibles par vague) et la taille définie par notre seuil dynamique. La qualité visuelle du maillage est mesurée à l'aide de la métrique MRMS, pour ne capturer que la distorsion géométrique.

4.3 Compression de la géométrie et de la connectivité

Dans cette section, nous exposons les résultats de notre méthode appliquée à plusieurs maillages, puis nous comparons notre algorithme avec deux méthodes de l'état de l'art [PK05, PHK+10]. Ces travaux ont été choisis car ils permettent de compresser progressivement des maillages non-variétés triangulaires. Nous avons mesuré les temps d'exécution grâce à un ordinateur CMSTORM SCOUT2 muni d'un processeur Intel Core i5 3.4GHz.

Avec le tableau 4.1, nous présentons les taux de compression sans perte et les temps d'exécution (à la fois pour la compression et pour la décompression) sur plusieurs modèles 3D. Ces différents maillages ont été choisis pour offrir une diversité de type de connectivité. À notre connaissance, la compression progressive présentée dans ce manuscrit est la première à autoriser une gestion générique des maillages polygonaux non-variétés. Cette classe de maillage représente l'ensemble des maillages surfaciques.

Les taux de compression présentés dans le tableau 4.1 comprennent les informations de géométrie et de connectivité. Tout comme les résultats des Sections 4.1 et 4.2, les compressions ont été réalisées avec une quantification des coordonnées 3D à 12 bits. Comme nous le montrons, notre approche décomprime les maillages présentés à la vitesse de 0.022 à 0.061 secondes par millier de sommets. Ainsi, environ 23000 à 8000 sommets sont décompressés et affichés en l'espace de 500ms. Ces performances montrent que notre méthode rentre dans le cadre d'une décompression en temps réel.

Nous avons également souhaité comparer notre méthode à des approches capables de compresser progressivement des maillages triangulaires non-variétés comme [PK05] et [PHK+10]. Comme mentionné dans la Section 2.5, les travaux de Peng et Kuo [PK05] utilisent un octree afin de coder l'ensemble des informations relatives au maillage. Peng *et al.* [PHK+10], quant à eux, ont une approche de regroupement des sommets guidée par les caractéristiques géométriques du maillage.

Modèle	# Sommets	Type	Taux comp.	Temps comp.	Temps décomp.
Bimba	8,857	Triangle 2-variété	35.31	0.264	0.056
Dragon	437,645	Triangle 2-variété	23.65	4.261	0.022
Dancing Ari	1,175,653	Triangle 2-variété	22.94	6.169	0.026
Bunny	35,947	Triangle 2-variété	27.62	0.287	0.057
Armadillo	125,340	Quadrilatère 2-variété	25.37	0.438	0.035
Bimba-q	15,653	Triangle/Quadrilatère 2-variété	29.05	0.298	0.061
Hippo	65,456	Polygonal 2-variété	28.55	0.589	0.048
Tractor	27,251	Triangle/Quadrilatère non-variété	26.60	0.330	0.055
House Plant	35,372	Triangle non-variété	31.29	0.339	0.056

Tableau 4.1 – Taux de compression sans perte et temps d'exécution pour plusieurs modèles 3D de types de connectivité divers : triangles, quadrilatères, polygones de degré supérieur, 2-variété et non-variété. Les taux de compression sont exprimés en bits par sommet (bps). Les temps de compression et de décompression sont affichés en secondes par millier de sommets. Les coordonnées 3D ont été quantifiées à 12 bits.

Modèle (# Sommets)	Méthode	Taux comp. (bps) 1.0	2.0	4.0	8.0	12.0	16.0
Horse (19,851)	Notre méthode	20.2	**9.3**	**4.1**	**1.6**	1.0	0.5
	[PK05]	31.9	19.4	10.2	3.3	**0.9**	**N/A**
	[PHK+10]	**19.0**	12.8	5.1	3.3	1.6	1.1
Rabbit (67,039)	Notre méthode	**5.3**	**2.4**	**1.1**	**0.6**	**0.3**	0.2
	[PK05]	18.9	10.2	8.3	2.2	0.6	**N/A**
	[PHK+10]	5.5	4.2	2.1	1.7	1.0	0.5
Maple* (45,499)	Notre méthode	28.3	17.7	9.2	**4.9**	**3.2**	**2.5**
	[PK05]	29.8	18.6	13.8	11.3	6.1	4.0
	[PHK+10]	**21.5**	**13.6**	**8.4**	5.4	3.7	2.8
Tractor* (27,251)	Notre méthode	31.2	26.3	14.5	6.2	3.8	2.0
Skeleton* (6,308)	Notre méthode	19.0	11.9	10.6	7.3	5.5	3.8

Tableau 4.2 – Tableau des erreurs géométriques par rapport à l'original (divisées par 10^4), mesurées à l'aide de MRMS, pour notre méthode, [PK05] et [PHK+10], à différents taux de compression. Les modèles marqués (*) sont non-variétés. Les meilleurs résultats sont notés en **gras**.

Le tableau 4.2 montre les distances MRMS obtenues par rapport au maillage original pour différents taux de compression, i.e. à différents moments de la décompression. Les valeurs affichées pour les méthodes [PK05] et [PHK+10] ont été fournies par les auteurs. La valeur "N/A" signifie que la décompression s'est terminée avant d'atteindre le taux de compression correspondant.

La figure 4.5 permet également de visualiser la progression de la qualité visuelle pendant la décompression. Ces courbes montrent que notre méthode offre un excellent compromis à bas et

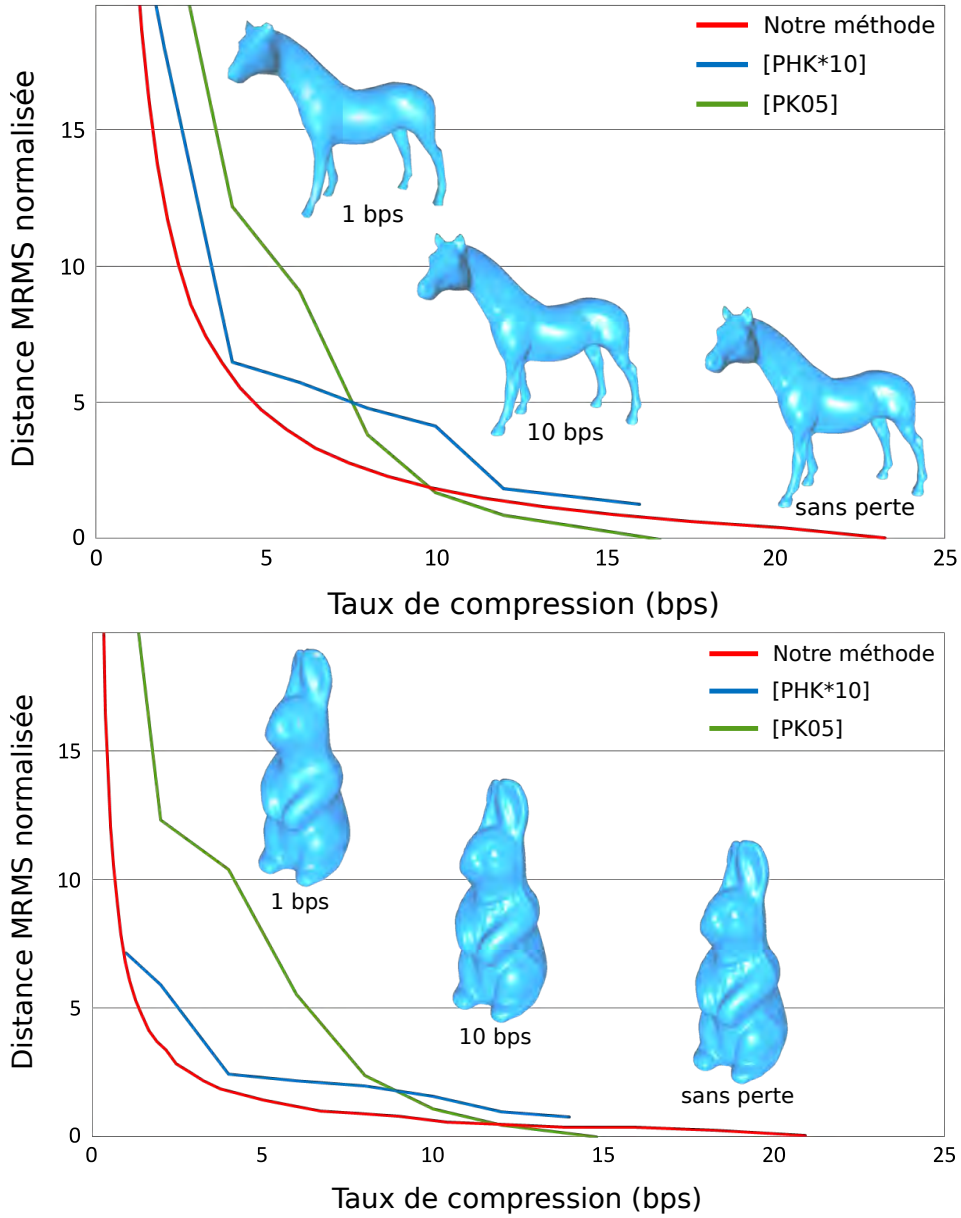


FIGURE 4.5 – Courbes de débit-distorsion obtenues avec les modèles *Horse* (à gauche) et *Rabbit* (à droite) à l'aide de notre méthode, [PK05] et [PHK+10]. La qualité visuelle des maillages est mesurée à l'aide de la métrique MRMS. Les valeurs MRMS sont divisées par 10^4.

moyen débit grâce, notamment, à son fort contrôle local de la distorsion. De plus, contrairement à [PHK+10], nous garantissons une restitution sans perte du maillage traité, même si le taux de compression final est plus coûteux. Il est également intéressant de voir que nos performances de compression ne chutent pas, par rapport aux autres méthodes, pour des maillages non-variétés. Par exemple, pour le modèle *Maple* (voir tableau 4.2), les résultats de notre méthode sont meilleurs que ceux de [PK05] dans toutes les colonnes et meilleurs que ceux de [PHK+10] sur les haut débits affichés.

4.4 Compression de la paramétrisation de la texture

La comparaison de nos travaux, concernant la compression de l'attribut de texture, est effectuée en deux temps. Premièrement, nous confrontons nos résultats avec ceux des méthodes proposées par Váša et Brunnett [VB14]. Ensuite, nous comparons notre compression du maillage et de ses coordonnées de texture avec celles de Hoppe [Hop96] et de Lavoué *et al.* [LCCD16]. Dans les deux cas, une comparaison visuelle est fournie.

Modèle	# Sommets	RMSE	[VB14] wp	[VB14] cotan	[VB14] mv	Notre méthode
Horse	52,345	0.0015	2.59	2.00	2.12	**0.71**
		0.0001	2.95	4.87	6.30	**2.80**
Fiery	66,216	0.0030	2.62	1.82	1.79	**0.43**
		0.0001	3.07	7.67	6.44	**2.69**
Victoria	17,259	0.0008	4.83	9.63	9.58	**4.62**
		0.0001	9.49	16.28	15.73	**6.81**
Bimba	9,285	0.0015	3.07	1.56	**1.24**	1.82
		0.0001	**5.64**	7.94	7.19	6.17
Frog	20,834	0.0015	4.03	7.64	6.86	**2.54**
		0.0001	8.96	15.55	14.64	**7.55**
Bunny	16,331	0.0030	3.00	3.22	3.21	**0.89**
		0.0001	6.58	12.77	12.17	**4.74**
Kachel	229,330	0.0004	2.82	**0.27**	0.29	0.86
		0.0001	2.84	**0.36**	0.39	1.73

Tableau 4.3 – Tableau des taux de compression obtenus avec notre méthode et [VB14] pour différentes erreurs MRMS sur les coordonnées de texture. [VB14]-wp, [VB14]-cotan et [VB14]-mv représentent respectivement les méthodes *weighted parallelogram prediction*, *cotan Laplacian* et *mean value Laplacian* proposées dans [VB14]. Les taux de compression sont exprimés en bits par sommet et correspondent seulement à la compression des coordonnées UV. Les meilleurs résultats sont notés en **gras**.

Le tableau 4.3 liste différents taux de compression, pour des valeurs de distorsion fixées. Les résultats relatives à [VB14] viennent de l'article original. Nous pouvons ainsi trouver trois méthodes différentes de [VB14] à côté de la notre : *weighted parallelogram prediction* (wp), *cotan Laplacian* et *mean value Laplacian*. Ces algorithmes représentent l'état de l'art de la compression dite *single rate* pour les coordonnées de texture. Les modèles présentés ainsi que leurs paramétrisations sont ceux utilisés dans [VB14]. Afin de correspondre aux mêmes erreurs globales sur les coordonnées UV (3ème colonne) pour des taux de compression sans perte, nous avons ajusté la valeur de quantification des coordonnées UV dans notre méthode.

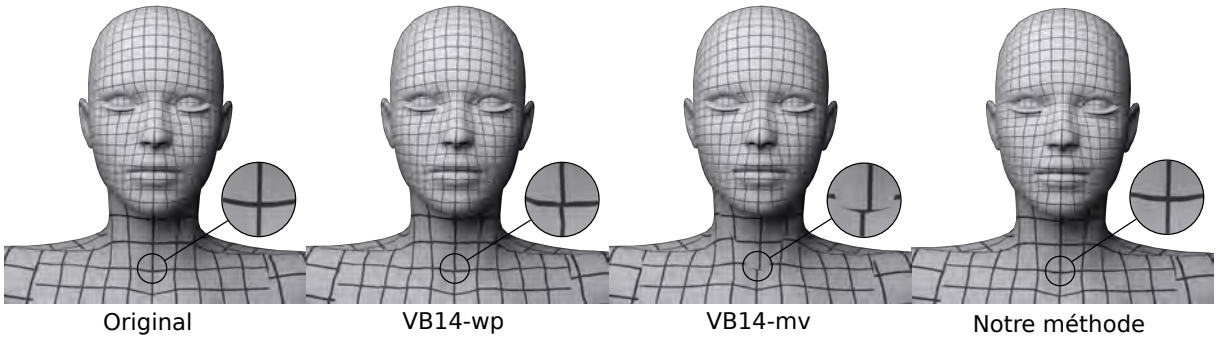


FIGURE 4.6 – Visualisation de la qualité de la paramétrisation après la compression du modèle *Victoria* par [VB14]-wp, [VB14]-mv et notre méthode. Les versions de [VB14] sont reconstruites en utilisant 5.25 bps pour les coordonnées UV alors que notre algorithme utilise 5.41 bps.

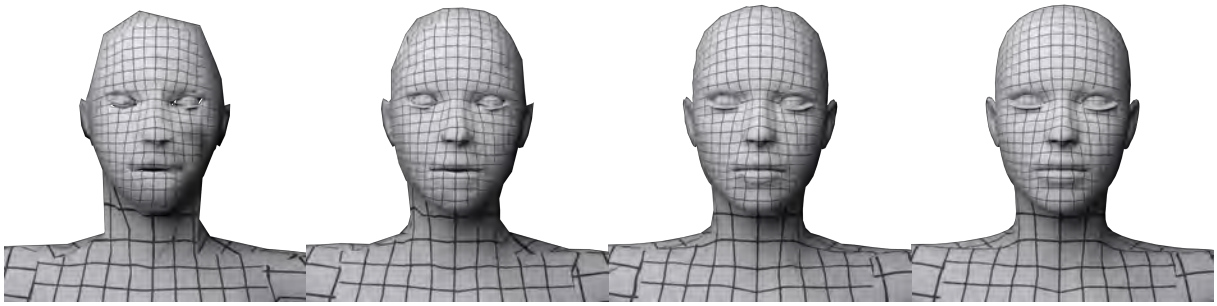


FIGURE 4.7 – Décompression progressive du modèle *Victoria* réalisée par notre méthode. Pour chaque LoD, nous affichons également la totalité du taux de compression nécessaire (géométrie, connectivité et paramétrisation de la texture), puis le nombre de sommets impliqués.

Nous voyons que notre méthode donne de meilleurs résultats sur la grande majorité des modèles proposés. Cela s'explique par le fait que, lors de la décompression, nous déterminons plusieurs *corners* grâce à la même information. Pour visualiser cette différence de résultat, nous présentons la figure 4.6. Cette figure représente le modèle *Victoria* après décompression. Nous voyons que, pour le même taux de compression, il nous est possible d'effectuer une quantification plus fine et donc de restituer une paramétrisation plus fidèle à l'original. Malheureusement, il ne nous a pas été possible de récupérer le résultat de [VB14]-cotan. C'est la raison pour laquelle il ne figure pas parmi les autres prises de vues.

Il est également important de préciser que la plupart des modèles du tableau 4.3 possèdent une paramétrisation obtenue au moyen d'un algorithme automatique. Ces modèles se prêtent donc particulièrement bien à notre encodage par prédiction géométrique. Cependant, même les paramétrisations manuelles comme celle de *Victoria* ou de *Frog* fournissent de bons résultats.

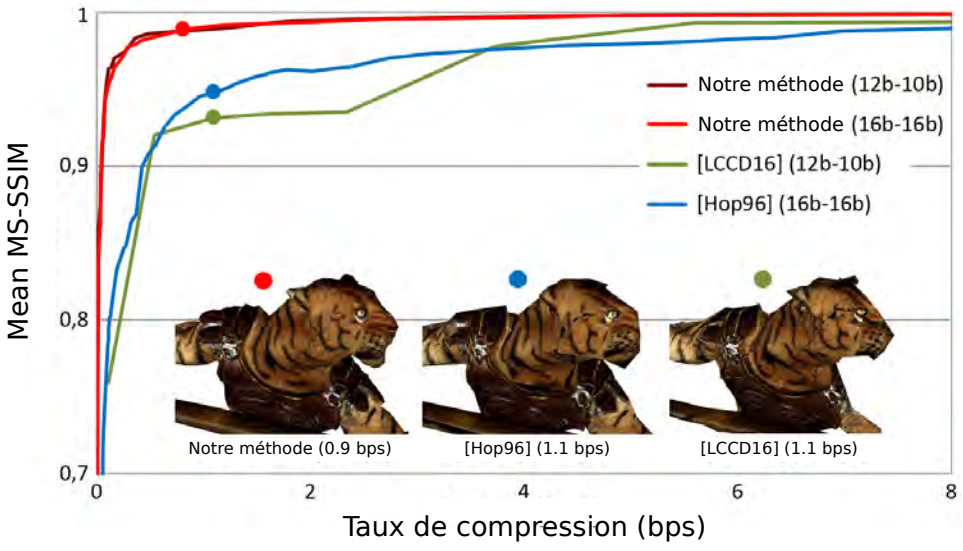


FIGURE 4.8 – Courbes de débit-distorsion obtenues avec le modèle *Tiger Fighter* pour notre méthode, [Hop96] et [LCCD16]. La qualité visuelle du maillage est mesurée à l'aide de la métrique perceptuelle MS-SSIM. Les LoDs affichés correspondent aux points marqués sur les différentes courbes. Ces LoDs sont associés à l'image de texture non-compressée. Pour chaque méthode, nous affichons à droite de son intitulé, entre parenthèses, le nombre de bits utilisés pour la quantification des coordonnées des sommets du maillages et des coordonnées UV.

De plus, nous pouvons noter, grâce à la figure 4.7, l'efficacité de notre métrique de sélection d'arête. En effet, la qualité de la conservation de la paramétrisation et des coutures de la texture dépendent entièrement de la partie p_{text} du calcul de poids effectué sur chaque arête.

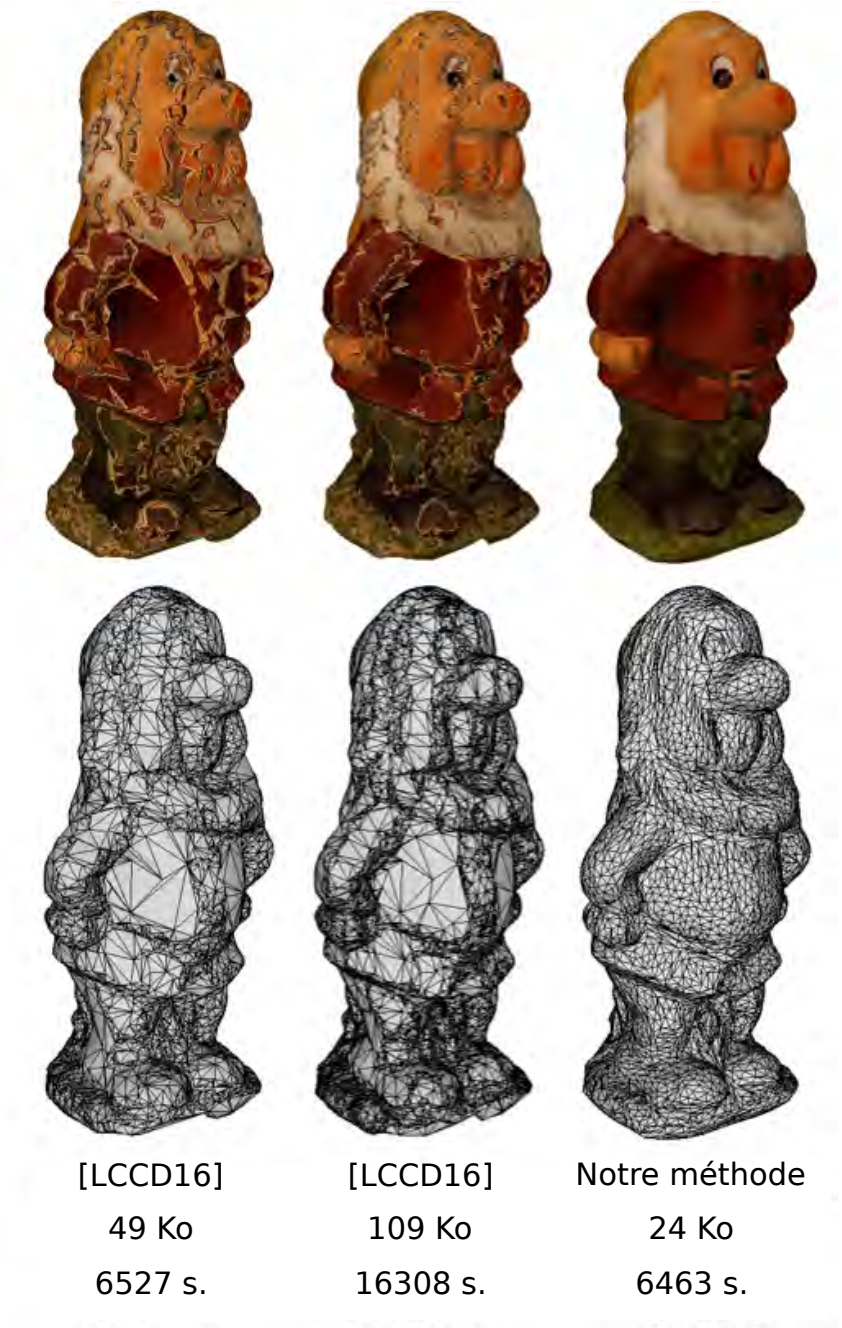


FIGURE 4.9 – Comparaison des LoDs obtenus avec notre méthode et [LCCD16]. Chaque modèle est illustré avec (en haut) et sans (en bas) texture. Nous affichons également la taille de données nécessaires pour la reconstruction de ces LoDs (géométrie, connectivité et paramétrisation de la texture), puis le nombre de sommets impliqués.

Dans un second temps, nous comparons nos travaux avec des méthodes de compression progressives capables de gérer les coutures de texture : *Progressive meshes* de Hoppe [Hop96] et l'algorithme récent de Lavoué *et al.* [LCCD16]. Ce dernier crée des bandeaux de triangles invisibles au niveau des coutures afin d'évoluer avec des coordonnées UV par sommet plutôt que par *corner*. Les deux méthodes sont néanmoins limitées aux maillages 2-variétés.

Nous avons utilisé l'implémentation originale des auteurs pour fournir les résultats présents dans les figures 4.8 et 4.9. La figure 4.8 représente les courbes débit-distorsion pendant la décompression du modèle *Tiger Fighter*, obtenues à l'aide de la métrique MS-SSIM. Les taux de compression correspondent, dans ce graphique, à l'encodage de la géométrie, de la connectivité et des coordonnées UV. En fonction des implémentations, il a fallu adapter la quantification de la géométrie et de la paramétrisation. Cela explique la présence de deux courbes relatives à notre méthode de compression.

Notre méthode donne de meilleurs résultats, comme nous le montrent les courbes. Les prises de vue présentes dans la figure 4.8 nous confirment aisément la différence de qualité visuelle. [Hop96] est très coûteuse (plus de 400 bps pour une décompression sans perte). [LCCD16], par contre, est plus efficace (26 bps contre 23 bps pour notre méthode pour une décompression sans perte), mais elle produit des LoDs de moins bonne qualité, à cause de l'apparition d'artefacts le long des coutures. La figure 4.9 permet de visualiser ces artefacts pour deux LoDs différents, ainsi que la connectivité du maillage (sans texture).

4.5 Multiplexage maillage-texture

Le but de nos travaux est véritablement de proposer une méthode de compression de maillages surfaciques entièrement progressive. Pour ce faire, nous utilisons le multiplexage des données liées au maillage et de celles liées à l'image de texture. La figure 4.10 illustre les résultats obtenus pour différents objets texturés (illustrés dans la figure 4.11). Cette fois, les taux présentés incluent l'ensemble des données nécessaires (i.e. la géométrie, la connectivité, la paramétrisation et la texture). Il est intéressant d'observer que pour des maillages assez similaires en terme de complexité de géométrie et de connectivité (par exemple, *Dwarf*, *Creature* et *Tiger Fighter*), les courbes restent équivalentes. La baisse de performance que l'on détecte pour la décompression du modèle *Tractor* est en grande partie structurelle. En effet, c'est un maillage polygonal non-variété comportant des zones non-convexes, des trous ou encore des parties fines qui rendent moins homogène l'encodage du maillage et pénalise donc le taux de compression et la qualité visuelle.

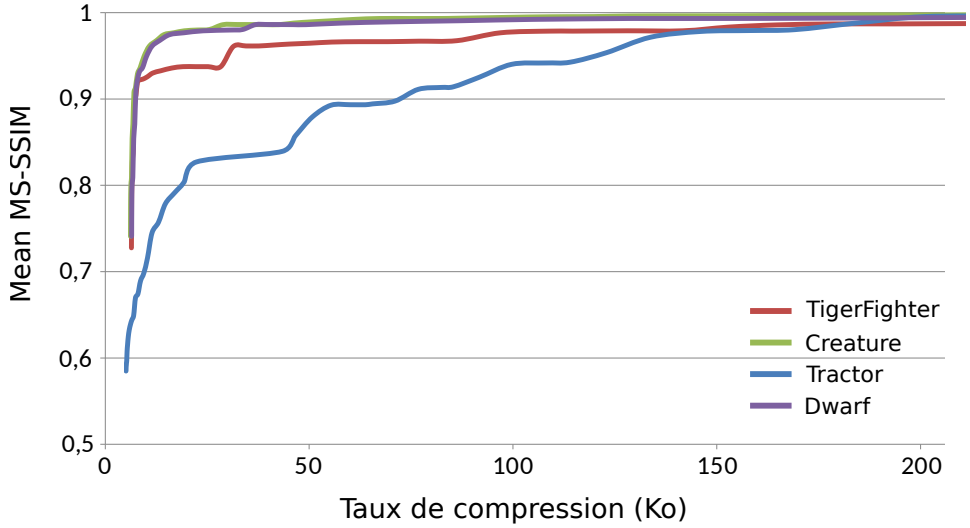


FIGURE 4.10 – Courbes de débit-distorsion obtenues avec différents modèles (*Tractor*, *Creature*, *Dwarf*, et *Tiger Fighter* à l'aide de notre méthode. La qualité visuelle du maillage est mesurée à l'aide de la métrique perceptuelle MS-SSIM. La décompression est réalisée grâce à notre stratégie de multiplexage maillage-texture.

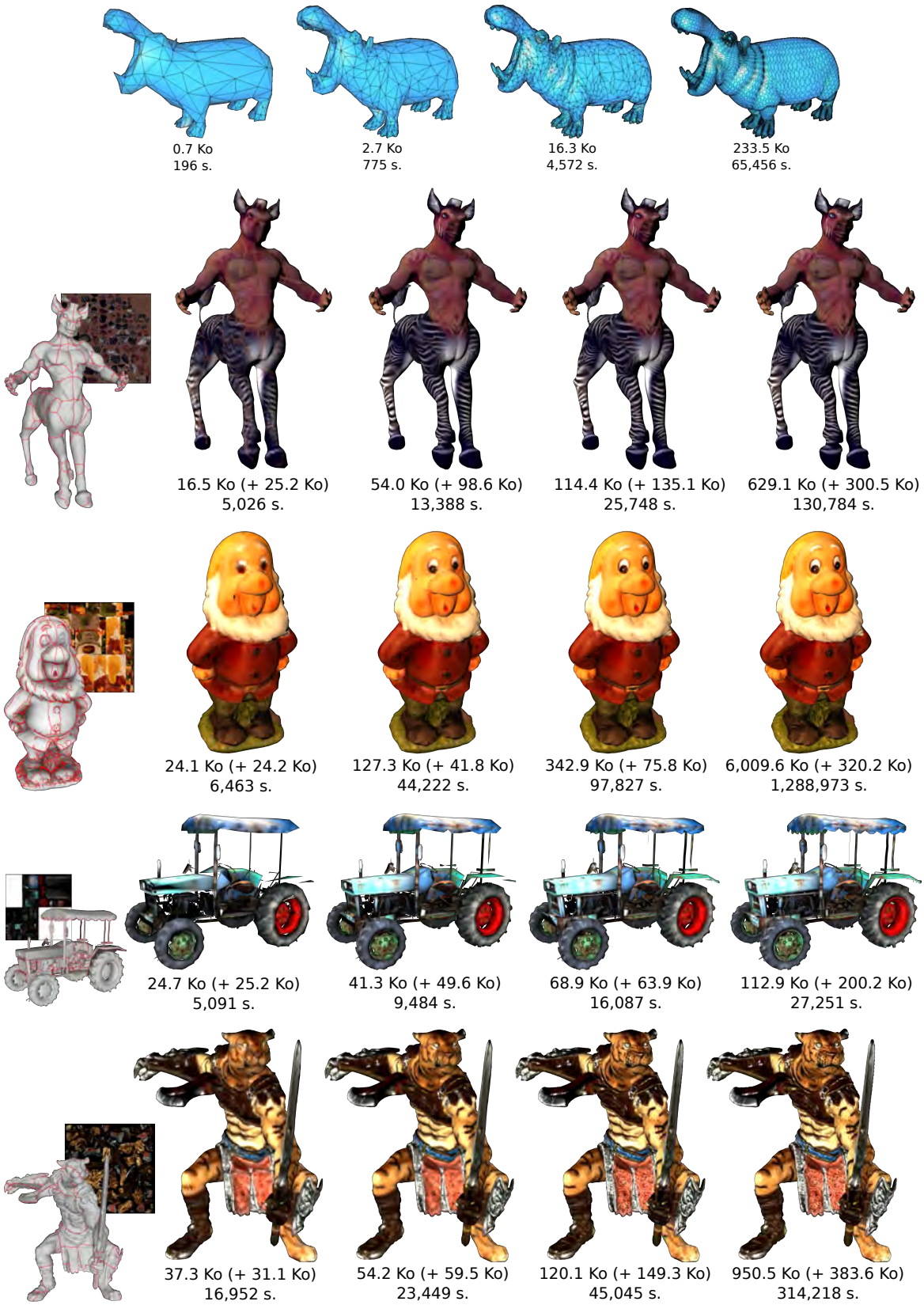


FIGURE 4.11 – Décompression progressive de différents modèles 3D (*Hippo*, *Creature*, *Dwarf*, *Tractor* et *Tiger Fighter*). La décompression est réalisée grâce à notre stratégie de multiplexage maillage-texture. Nous affichons également la taille des données relatives aux éléments du maillage (géométrie, connectivité et paramétrisation de la texture) et relatives à l'image de texture (entre parenthèse), puis le nombre de sommets impliqués.

Pour avoir un aperçu des résultats visuels de notre méthode, des niveaux de détail de ces maillages sont illustrés sur la figure 4.11. Nous avons volontairement choisi les différents LoDs présentés dans le but d'exposer au mieux les caractéristiques de notre méthode. Nous ajoutons également, à gauche des maillages texturés, les coutures présentes sur le modèle ainsi que l'image de texture, pour que le lecteur puisse se faire une idée de la complexité de la compression. Les LoDs de chaque maillage, même en dessous de 10% de la taille compressée sans perte, conservent une qualité visuelle satisfaisante. De plus, au-delà de 100 Ko (maillage et texture confondus), nous obtenons une version quasiment identique au maillage final. Ceci est d'autant plus intéressant lorsque l'on traite des maillages suréchantillonnés comme le modèle *Dwarf* : le deuxième LoD en partant de la droite ne demande que 2.7% de la taille finale, pour une qualité visuelle qui serait suffisante dans bien des applications.

Chapitre 5

Conclusion

La compression progressive d'objet 3D, en particulier de maillages surfaciques, est véritablement un problème d'actualité. Si les enjeux étaient déjà bien présents à la fin du 20ème siècle, l'expansion d'Internet et de tous ses services en a fait une préoccupation majeure dans le domaine de l'informatique graphique. En parcourant l'état de l'art traitant de ce sujet, il devient clair que c'est également un problème difficile sous bien des aspects. Dans le contexte actuel, il est nécessaire de prendre en compte un certain nombre de paramètres comme la structure générale ou la connectivité du maillage, mais également plusieurs attributs liés à ce dernier. Ces éléments doivent tous être traités avec le plus grand soin afin de parvenir à un compromis satisfaisant entre rapidité de compression et surtout de décompression, taux de compression et qualité visuelle. La solution, ou du moins une solution, à ce problème doit donc nécessairement tenir compte de ces différents facteurs.

5.1 Apport de la méthode

Dans le cadre de cette thèse, nous tentons de proposer une solution à ce problème. La méthode présentée dans ce manuscrit permet, de façon générique, de compresser progressivement l'ensemble des maillages surfaciques texturés ou non-texturés sans perte d'information. Ces travaux sont basés sur une étude de l'état de l'art qui nous a permis de repérer différents mécanismes efficaces pour la compression de maillages surfaciques. Notre méthode tire ainsi partie des avancées scientifiques dans ce domaine. Néanmoins, l'ensemble de nos recherche ont été guidées par la volonté d'offrir une méthode autant efficace que générique. Pour atteindre cet objectif, il nous a fallu mettre en place autant de solutions nouvelles pour construire notre méthode finale.

Durant la mise en œuvre de nos algorithmes, nous n'avons eu de cesse de privilégier et d'optimiser la qualité visuelle des LoDs générés durant la décompression. La première de nos contributions est donc, logiquement, notre stratégie de contrôle local de la distorsion. La priorisation des simplifications unitaires couplée à une métrique adaptée, facilement modifiable en fonction des applications, procure à notre méthode un avantage indéniable. Ce mécanisme nous a permis d'obtenir d'excellents résultats, en terme de qualité visuelle. La préservation des coutures le long du maillage offre, notamment, les meilleurs résultats de l'état de l'art, en terme de distorsion de texture.

Afin de conserver une méthode compétitive, en terme de taux de compression, nous avons également mis en place plusieurs stratégies de prédiction et de réduction d'entropie. La pleine

maîtrise des contraintes topologiques au sein d'un patch de simplification offre une réduction significative de l'encodage de la connectivité. De plus, les prédictions basées géométrie utilisées pour un codage plus efficace de la connectivité et des coordonnées UV ont largement contribué à diminuer la taille de l'information compressée. Cela, ajouté à l'emploi d'un repère de Frenet concernant la géométrie, nous donne une méthode de compression progressive efficace. Nous avons d'ailleurs montré qu'elle fournissait des meilleurs résultats que les méthodes de l'état de l'art à bas et moyen débits. Cette plage d'efficacité correspond, en outre, tout à fait au domaine d'applications ciblé.

Comme nous l'avons souhaité, notre méthode s'applique à tous les maillages surfaciques, qu'ils soient triangulaires ou polygonaux, 2-variétés ou non-variétés. Pour offrir cette généricité, nous avons en premier lieu étendu les opérateurs de simplification et de raffinement proposés par [PH97] aux configurations polygonales. Il a également été nécessaire de modifier l'encodage de certaines configurations. Ces aménagements non-triviaux ont été réalisés sans surcoût, mais également avec une optimisation du taux de compression et du temps d'exécution relatifs à ce mécanisme.

Pour finir, notre méthode procure des temps de compression et de décompression satisfaisants et tout à fait adaptés à la transmission et à la visualisation interactive d'objets 3D via le Web.

5.2 Limitations

À travers les différents résultats fournis dans la Section 4, nous pouvons voir que notre compression génère des taux de compression sans perte significativement plus élevés que d'autres méthodes existantes. Ce surcoût est en partie dû au caractère générique de la méthode, notamment à l'adaptation aux maillages non-variétés. En effet, là où certaines approches, comme [AD01a], s'appuient sur les contraintes topologiques qu'offrent les maillages 2-variétés, pour obtenir des taux de compression très faibles, nous sommes confronté à une connectivité complexe, sans *a priori* exploitable. Il est donc nécessaire de fournir plus d'information explicitement au décodeur. Mais une autre partie de ce surcoût vient de notre pilotage très fin de la distorsion qui permet de proposer d'excellents résultats à bas et moyen débit.

5.3 Perspectives

En ayant, à ce jour, un certain recul sur notre méthode, il est possible de proposer quelques améliorations, à divers niveaux. Tout d'abord, notre décompression est construite de telle sorte qu'une parallélisation des raffinements unitaires est réalisable. Ce procédé, en lien avec une optimisation de la structure de données utilisée, permettrait d'atteindre les performances de méthodes comme [PD15], spécifiquement adaptées à la visualisation interactive sur le Web.

Comme nous l'avons parfois mentionné dans ce manuscrit, certaines autres pistes de recherche, comme le calcul du seuil dynamique, le choix de la position d'un sommet résultant d'une contraction ou encore une meilleure analyse de l'image de texture, mèneraient sans doute à une amélioration de la méthode actuelle. Au fil de nos expérimentations, nous avons également observé que la place des arêtes pendantes dans la visualisation n'était probablement pas assez mise en valeur. En effet, une arête pendante peut tout à fait, en fin de simplification, représenter une partie entière du maillage original (par exemple, un membre de personnage). Or, lors d'une visualisation standard du maillage, ces arêtes sont totalement ignorées. La substitution conceptuelle de ces arêtes par des cylindres (de taille adaptée ou fixe) apporterait un plus pour

le contrôle de la distorsion, pendant la simplification. Mais cela donnerait surtout une meilleure qualité visuelle pour les LoDs les plus grossiers, durant la décompression, comme peuvent le laisser penser les travaux de Popović et Hoppe [PH97].

Cependant, une de nos plus vastes perspectives est sans nul doute l'extension de notre méthode à des maillages volumiques. Ce champ de recherche reste encore peu exploré. Pourtant, la compression progressive de maillages volumiques trouve assez naturellement une utilité dans le domaine de la visualisation scientifique ou encore de la simulation physique d'objets déformables.

Bibliographie

[AD01a] Pierre Alliez and Mathieu Desbrun. Progressive compression for lossless transmission of triangle meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 195–202. ACM, 2001.

[AD01b] Pierre Alliez and Mathieu Desbrun. Valence-driven connectivity encoding for 3d meshes. In *Computer graphics forum*, pages 480–489. Wiley Online Library, 2001.

[AS96] Maria-Elena Algorri and Francis Schmitt. Mesh simplification. In *Computer Graphics Forum*, pages 77–86. Wiley Online Library, 1996.

[Bau72] Bruce G Baumgart. Winged edge polyhedron representation. Technical report, DTIC Document, 1972.

[BBS+02] Botsch Steinberg Bischoff, M Botsch, S Steinberg, S Bischoff, L Kobbelt, and Rwth Aachen. Openmesh–a generic and efficient polygon mesh data structure. In *In OpenSG Symposium*, 2002.

[CDFW11] David Canino, Leila De Floriani, and Kenneth Weiss. Ia* : An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions. *Computers & Graphics*, 35(3) :747–753, 2011.

[COLR99] Daniel Cohen-Or, David Levin, and Offir Remez. Progressive compression of arbitrary triangular meshes. In *IEEE visualization*, volume 99, pages 67–72, 1999.

[CRS98] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro : measuring error on simplified surfaces. In *Computer Graphics Forum*, pages 167–174. Wiley Online Library, 1998.

[Dam16] Guillaume Damiand. Linear cell complex. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.8.1 edition, 2016.

[Dee95] Michael Deering. Geometry compression. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 13–20. ACM, 1995.

[DFH05] Leila De Floriani and Annie Hui. Representing non-manifold shapes in arbitrary dimensions. In *Proceedings of the 6th Israel-Korea Bi-National Conference on New Technologies and Visualization Methods for Product Development on Design and Reverse Engineering*, pages 11–15, 2005.

[DFKP05] Leila De Floriani, Leif Kobbelt, and Enrico Puppo. A survey on data structures for level-of-detail models. In *Advances in multiresolution for geometric modelling*, pages 49–74. Springer, 2005.

[DFMMP00] Leila De Floriani, Paola Magillo, Franco Morando, and Enrico Puppo. Dynamic view-dependent multiresolution on a client–server architecture. *Computer-Aided Design*, 32(13) :805–823, 2000.

[DFMPS04] Leila De Floriani, Paola Magillo, Enrico Puppo, and Davide Sobrero. A multiresolution topological representation for non-manifold meshes. *Computer-Aided Design*, 36(2) :141–159, 2004.

[DG00] Olivier Devillers and P-M Gandoin. Geometric compression for interactive transmission. In *Visualization 2000. Proceedings*, pages 319–326. IEEE, 2000.

[GBTS99] André Guéziec, Frank Bossen, Gabriel Taubin, and Claudio Silva. Efficient compression of non-manifold polygonal meshes. In *Visualization'99. Proceedings*, pages 73–512. IEEE, 1999.

[GD02] Pierre-Marie Gandoin and Olivier Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Transactions on Graphics (TOG)*, 21(3) :372–379, 2002.

[GH97] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[GLLR11a] Topraj Gurung, Daniel Laney, Peter Lindstrom, and Jarek Rossignac. Squad : Compact representation for triangle meshes. In *Computer Graphics Forum*, pages 355–364. Wiley Online Library, 2011.

[GLLR11b] Topraj Gurung, Mark Luffel, Peter Lindstrom, and Jarek Rossignac. *LR : compact connectivity representation for triangle meshes*. ACM, 2011.

[GTLH98] André Guéziec, Gabriel Taubin, Francis Lazarus, and William Horn. Simplicial maps for progressive transmission of polygonal surfaces. In *Proceedings of the third symposium on Virtual reality modeling language*, pages 25–31. ACM, 1998.

[Gue95] A GueÂziec. [a]surface simplification with variable tolerance. In [o] *Proc. Second Ann. Int'l Symp. Medical Robotics and Computer Assisted Surgery*, pages 132–139, 1995.

[H+52] David A Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9) :1098–1101, 1952.

[Ham94] Bernd Hamann. A data reduction scheme for triangulated surfaces. *Computer aided geometric design*, 11(2) :197–214, 1994.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM, 1993.

[HG97] Paul S Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, DTIC Document, 1997.

[Hop96] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996.

[Hop98] Hugues Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1) :27–36, 1998.

[IS01] Martin Isenburg and Jack Snoeyink. Compressing the property mapping of polygon meshes. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pages 4–11. IEEE, 2001.

[IS03] Martin Isenburg and Jack Snoeyink. Compressing texture coordinates with selective linear predictions. In *Computer Graphics International, 2003. Proceedings*, pages 126–131. IEEE, 2003.

[Ise02] Martin Isenburg. Compressing polygon mesh connectivity with degree duality prediction. In *Graphics Interface*, volume 2, pages 161–170, 2002.

[KADS02] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Graphical Models*, 64(3) :147–168, 2002.

[KBG02] Zachi Karni, Alexander Bogomjakov, and Craig Gotsman. Efficient compression and rendering of multi-resolution meshes. In *Visualization, 2002. VIS 2002. IEEE*, pages 347–354. IEEE, 2002.

[Ket99] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1) :65–90, 1999.

[KG00] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 279–286. ACM Press/Addison-Wesley Publishing Co., 2000.

[LCCD16] Guillaume Lavoué, Laurent Chevalier, Florian Caillaud, and Florent Dupont. Progressive streaming of textured 3d models in a web browser. In *Symposium on Interactive 3D Graphics and Games*, 2016.

[LL01] Sang Hun Lee and Kunwoo Lee. Partial entity structure : a compact non-manifold boundary representation based on partial topological entities. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 159–170. ACM, 2001.

[LLD09] Ho Lee, Guillaume Lavoué, and Florent Dupont. Adaptive coarse-to-fine quantization for optimizing rate-distortion of progressive mesh compression. In *VMV*, pages 73–82, 2009.

[LLV16] Guillaume Lavoué, Mohamed Larabi, and Libor Vasa. On the efficiency of image metrics for evaluating the visual quality of 3d models. *IEEE Transactions on Visualization and Computer Graphics*, 22(8) :1987–1999, 2016.

[MCAH12] Adrien Maglo, Clément Courbet, Pierre Alliez, and Céline Hudelot. Progressive compression of manifold polygon meshes. *Computers & Graphics*, 36(5) :349–359, 2012.

[MZP09] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. Tfan : A low complexity 3d mesh compression algorithm. *Computer Animation and Virtual Worlds*, 20(2-3) :343–354, 2009.

[P+96] Enrico Puppo et al. Variable resolution terrain surfaces. In *CCCG*, pages 202–210, 1996.

[PD15] Federico Ponchio and Matteo Dellepiane. Fast decompression for web-based view-dependent 3D rendering. In *ACM International Conference on 3D Web Technology*, pages 199–207, 2015.

[PH97] Jovan Popović and Hugues Hoppe. Progressive simplicial complexes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 217–224. ACM Press/Addison-Wesley Publishing Co., 1997.

[PHK+10] Jingliang Peng, Yan Huang, C-C Jay Kuo, Ilya Eckstein, and M Gopi. Feature oriented progressive lossless mesh coding. In *Computer Graphics Forum*, pages 2029–2038. Wiley Online Library, 2010.

[PK05] Jingliang Peng and C-C Jay Kuo. Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition. In *ACM Transactions on Graphics (TOG)*, pages 609–616. ACM, 2005.

[PR00] Renato Pajarola and Jarek Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :79–93, 2000.

[RB93] Jarek Rossignac and Paul Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in computer graphics*, pages 455–465. Springer, 1993.

[RL79] Jorma Rissanen and Glen G Langdon. Arithmetic coding. *IBM Journal of research and development*, 23(2) :149–162, 1979.

[Ros99] Jarek Rossignac. Edgebreaker : Connectivity compression for triangle meshes. *Visualization and Computer Graphics, IEEE Transactions on*, 5(1) :47–61, 1999.

[Ros01] Jarek Rossignac. 3d compression made simple : Edgebreaker with zipandwrap on a corner-table. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 278–283. IEEE, 2001.

[RR96] Rémi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. In *Computer Graphics Forum*, pages 67–76. Wiley Online Library, 1996.

[SB11] D. Sieger and M. Botsch. Design, implementation, and evaluation of the `Surface_mesh` data structure. In *Proceedings of the 20th International Meshing Roundtable*, pages 533–550, 2011.

[SG03] Frutuoso GM Silva and Abel JP Gomes. Aif-a data structure for polygonal meshes. In *Computational Science and Its Applications—ICCSA 2003*, pages 478–487. Springer, 2003.

[SL96] Marc Soucy and Denis Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Computer vision and image understanding*, 63(1) :1–14, 1996.

[SSGH01] Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM, 2001.

[SZL92] William J Schroeder, Jonathan A Zarge, and William E Lorensen. Decimation of triangle meshes. In *ACM Siggraph Computer Graphics*, pages 65–70. ACM, 1992.

[TA04] Dihong Tian and Ghassan AlRegib. Fqm : a fast quality measure for efficient transmission of textured 3d models. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 684–691. ACM, 2004.

[TG98] Costa Touma and Craig Gotsman. Triangle mesh compression. *Proc Graphics Interface. pp. 26-34. 1998*, 1998.

[TGHL98] Gabriel Taubin, André Guéziec, William Horn, and Francis Lazarus. Progressive forest split compression. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 123–132. ACM, 1998.

[TJL+12] Jiang Tian, Wenfei Jiang, Tao Luo, Kangying Cai, Jingliang Peng, and Wencheng Wang. Adaptive coding of generic 3d triangular meshes based on octree decomposition. *The Visual Computer*, 28(6-8) :819–827, 2012.

[Tut62] William Thomas Tutte. A census of planar triangulations. *Canad. J. Math*, 14(1) :21–38, 1962.

[VB13] Libor Vasa and Guido Brunnett. Exploiting connectivity to improve the tangential part of geometry prediction. *IEEE transactions on visualization and computer graphics*, 19(9) :1467–1475, 2013.

[VB14] Libor Váša and Guido Brunnett. Efficient encoding of texture coordinates guided by mesh geometry. In *Computer Graphics Forum*, pages 25–34. Wiley Online Library, 2014.

[VCP09] Sébastien Valette, Raphaëlle Chaine, and Rémy Prost. Progressive lossless mesh compression via incremental parametric refinement. In *Computer Graphics Forum*, pages 1301–1310. Wiley Online Library, 2009.

[WHTS01] Jian-Hua Wu, Shi-Min Hu, Chiew-Lan Tai, and Jia-Guang Sun. An effective feature-preserving mesh simplification scheme based on face constriction. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pages 12–21. IEEE, 2001.

[WNC87] IH Witten, RM Neal, and JG Cleary. Arithmetic coding for data compression. *Communicatiosn of the ACM*, 30(6) :520–540, 1987.

[WSB03a] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. Ieee, 2003.

[WSB03b] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. Ieee, 2003.

[YLK04] Sheng Yang, Chao-Hua Lee, and C-C Jay Kuo. Optimized mesh and texture multiplexing for progressive textured model transmission. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 676–683. ACM, 2004.

[YSK04] Sheng Yang, Meiyin Shen, and Chung-Chieh Jay Kuo. Joint mesh-texture optimization for progressive transmission. In *Electronic Imaging 2004*, pages 669–679. International Society for Optics and Photonics, 2004.

[ZLLY12] Long Zeng, Yong-Jin Liu, Sang Hun Lee, and Matthew Ming-Fai Yuen. Q-complex : Efficient non-manifold boundary representation with inclusion topology. *Computer-Aided Design*, 44(11) :1115–1126, 2012.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : **CAILLAUD** DATE de SOUTENANCE : **17 / 01 / 2017**
Prénoms : **Florian**

TITRE : **Compression progressive de maillages surfaciques texturés**

NATURE : **Doctorat** Numéro d'ordre :

Ecole doctorale : **Informatique et Mathématiques de Lyon**

Spécialité : **Informatique**

RESUME : Les modèles 3D deviennent de plus en plus détaillés. Cela augmente considérablement le volume de données les décrivant. Cependant un nombre croissant d'applications sont contraintes en mémoire et/ou en vitesse. Dans un contexte Web, ces difficultés sont encore plus présentes. Cette situation entraîne des incompatibilités, des latences de transmission ou d'affichage qui sont souvent problématiques.

La compression progressive de ces modèles est une des solutions envisageables. Le but étant de compresser les informations (géométrie, connectivité et attributs associés) de façon à pouvoir reconstruire progressivement le maillage. La compression progressive propose très rapidement un aperçu fidèle du modèle 3D pour ensuite le raffiner jusqu'à retrouver le maillage complet. Ceci permet un certain confort pour l'utilisateur et un contrôle sur la quantité d'éléments à visualiser ou à traiter.

Généralement, les approches existantes pour la compression progressive se focalisent sur le traitement de maillages 2-variétés triangulaires. Très peu de méthodes sont capables de compresser progressivement des maillages surfaciques non-variétés et, à notre connaissance, aucune ne permet de compresser un maillage surfacique quelque soit son type (i.e. non-variété et/ou polygonal). Pour supprimer ces limitations, nous présentons une méthode de compression progressive permettant de traiter l'ensemble des maillages surfaciques. De plus, notre approche tient compte de l'attribut de texture potentiellement associé à ces maillages.

Pour ce faire, nous décimons progressivement le maillage à l'aide de nouveaux opérateurs de simplification et de raffinement. Cette décimation est guidée par une métrique locale ayant pour but de préserver la géométrie et la paramétrisation de la texture. Durant cette simplification, nous encodons progressivement les informations nécessaires. Nous mettons également en œuvre différentes stratégies d'encodage de la connectivité, de la géométrie et des coordonnées de texture. La carte de texture est compressée progressivement puis multiplexée au flux de compression afin d'optimiser le ratio débit-distorsion lors de la décompression.

MOTS-CLÉS : **Compression progressive, maillages surfaciques, textures, multi-résolution**

Laboratoire (s) de recherche : **LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information)**

Directeur de thèse: **Guillaume Lavoué (Maître de conférence HDR, INSA Lyon)**

Président de jury :

Composition du jury :

ALLIEZ Pierre, Directeur de recherche, INRIA Sophia-Antipolis (Rapporteur)

BONNEAU George-Pierre, Professeur des universités, Université Grenoble Alpes (Rapporteur)

BECHMANN Dominique, Professeure des universités, Université de Strasbourg (Examinatrice)

HUDELOT Céline, Maître de conférence, École Centrale Paris (Examinatrice)

LAVOUÉ Guillaume, Maître de conférence HDR, INSA Lyon (Directeur de thèse)

VIDAL Vincent, Maître de conférence, Université Claude Bernard Lyon (Co-encadrant)

DUPONT Florent, Professeur des universités, Université Claude Bernard Lyon (Examinateur)