Université Claude Bernard Lyon 1

Université de Lyon

Numéro d'ordre : 197-2013

Année : 2013

THÈSE DE L'UNIVERSITÉ DE LYON

Délivrée par

L'UNIVERSITÉ CLAUDE BERNARD LYON 1

ÉCOLE DOCTORALE INFOMATHS

INFORMATIQUE ET MATHÉMATIQUES DE LYON

DIPLÔME DE DOCTORAT

(arrêté du 7 août 2006)

soutenue publiquement le 24 octobre 2013

par

Jérémy Espinas

Transformations compactes de triangulations surfaciques par bascule d'arête

Membres du jury :

Rapporteurs :	Dominique Bechmann	-	Université de Strasbourg
	Jean-Daniel Boissonnat	-	INRIA Sophia-Antipolis
Directeur :	Raphaëlle Chaine	-	Université Lyon 1
Co-directeur :	Pierre-Marie Gandoin	-	Université Lyon 2
Examinateurs :	Georges-Pierre Bonneau	-	Université de Grenoble
	Ramsay Dyer	-	Université de Groningen
	Céline Hudelot	-	Ecole Centrale Paris
	Sébastien Valette	-	INSA de Lyon

Recherches effectuées au Laboratoire

LIRIS

UMR 5205 du CNRS

Résumé :

Le développement de la numérisation systématique des formes 3D (conservation du patrimoine national, commerce électronique, *reverse engineering*, intégration d'objets réels dans des environnements de réalité virtuelle) et le besoin toujours croissant de ces objets géométriques dans de nombreuses applications (conception assistée par ordinateur, calcul de simulations par éléments finis, système d'informations géographiques, loisirs numériques) a entraîné une augmentation vertigineuse du volume de données à traiter, avec l'émergence de nombreuses méthodes de compression de modèles 3D. Ce volume de données devient encore plus difficile à maîtriser lorsque l'aspect temporel entre en jeu.

Les maillages correspondent au modèle classiquement utilisé pour modéliser les formes numérisées et certaines approches de compression exploitent la propriété qu'une bonne estimation de la connectivité peut être déduite de l'échantillonnage, lorsque ce dernier s'avère suffisamment dense. La compression de la connectivité d'un maillage revient alors au codage de l'écart entre deux connectivités proches. Dans ce mémoire, nous nous intéressons au codage compact de cette différence pour des maillages surfaciques.

Nos travaux sont fondés sur l'utilisation de la bascule d'arête (*edge flip*) et l'étude de ses propriétés. Nos contributions sont les suivantes :
- Etant donné deux triangulations connexes partageant le même nombre de sommets et un même genre topologique, nous proposons un algorithme direct et efficace pour générer une séquence de bascules d'arêtes permettant de passer d'un maillage à un autre. Nous nous appuyons sur une correspondance entre les sommets des deux maillages, qui, si elle est non fournie, peut être choisie de manière totalement aléatoire.
- Nous généralisons ensuite la bascule d'arête à des triangulations dans lesquelles nous donnons un indice différent à chaque arête. Nous mettons en évidence le fait qu'une séquence de bascules d'arêtes peut être utilisée pour transposer des indices, sous certaines conditions. Ce résultat ouvre la généralisation des bascules d'arêtes à des maillages dont les faces ne sont pas forcément triangulaires. Il nous a également permis de développer un algorithme de réduction de séquences de bascules d'arêtes.
- Nous présentons enfin une approche de codage compact d'une séquence de bascules d'arêtes entre deux triangulations surfaciques, et nous déterminons dans quelles conditions il est préférable d'utiliser cette transformation compacte entre deux connectivités au lieu de les coder indépendamment par un algorithme statique.

Mots clés : maillage, bascule d'arête, compression, permutation, géométrie algorithmique.

Abstract :

The development of scanning 3D shapes (national heritage conservation, e-commerce, reverse engineering, virtual reality environments) and the growing need for geometric objects in many applications (computer-aided design, simulations, geographic information systems, digital entertainment) have led to a dramatic increase in the volume of data to be processed, and the emergence of many methods of compression of 3D models. This volume of data becomes even more difficult to control when the temporal aspect comes in.

Meshes correspond to the pattern typically used to model the scanned forms and some approaches exploit a property of compression that a good estimation of connectivity can be derived from sampling, when it appears sufficiently dense. Compressing the connectivity of a mesh is equivalent to coding the difference between two close connectivities. In this thesis, we focus on the compact coding of this difference for 2-dimensional meshes.

Our work is based on the use and study of the properties of the edge flip. Our contributions are the following :
- Given two connected triangulations that share the same number of vertices and the same topological genus, we propose a direct and efficient algorithm to generate a sequence of edge flips to change one mesh into the other. We rely on a correspondence between the vertices of the two meshes, which, if not provided, may be chosen randomly. The validity of the algorithm is based on the fact that we intend to work in a triangulation of a different class from those generally used.
- We then generalize the edge flips to triangulations in which we identify each edge with a label. We show that a sequence of edge flips can be used to transpose two labels, under certain conditions. From this result, the edge flip can be generalized to meshes whose faces are not necessarily triangular, which allowed us to develop an algorithm for reducing sequences of edge flips.
- Finally, we present a compact coding approach for a sequence of edge flips, and determine under what conditions it is better to use this compact transformation between two connectivities instead of coding them independently by a static algorithm.

Keywords : mesh, edge flip, compression, permutation, computational geometry.

Remerciements

Je tiens tout d'abord à remercier Raphëlle et Pierre-Marie pour m'avoir proposé cette thèse et pour la confiance qu'ils m'ont accordé durant ces quatre années de recherche. Le temps passé avec Raphaëlle m'a beaucoup apportée scientifiquement et humainement. Pierre-Marie, quant à lui, a su me soutenir tout au long de mes recherches et me remotiver suite aux aléas tels que des refus d'articles.

Je remercie également Dominique Bechmann et Jean-Daniel Boissonnat pour l'intérêt qu'ils ont bien voulu me porter en acceptant d'être rapporteurs. Leurs remarques constructives ont permis d'améliorer grandement la qualité de ce manuscrit.

Je tiens aussi à exprimer mes remerciements à Georges-Pierre Bonneau, Ramsay Dyer, Céline Hudelot et Sébastien Valette pour leur participation à mon jury de soutenance.

Durant mon doctorat, j'ai fait de nombreuses rencontres très enrichissantes. Je tiens notamment à remercier une nouvelle fois Sébastien pour l'ensemble des réunions qui a toujours été une source d'idées nouvelles.

Je remercie également les nombreux permanents du laboratoire LIRIS qui m'ont accompagné : Samir, Eric et Eric, Julie, Elianne, Thierry, Gilles, Nicolas, Alexandre, Said, Carole, Martine, Martial et Romuald. Merci aussi à Catherine, Faty, Hélène, Isabelle, Josiane, Renée, Saïda et Sylvie pour leur aide administrative.

Je tiens aussi à remercier les très nombreux doctorants et post-doctorants que j'ai eu la chance de côtoyer : Maxime, Adrien, Clément, Lucian, Baptiste, Miguel, Aurélien, Gilles, Jean-David, Houssam, Fabien, Mahmoud, Camille, Elsa, François, Xavier, Petru, Ricardo, Jacques, Lucile, Samba, Vincent, Ho, Hichem, Nadim, Mickaël, ainsi que les footeux de l'Institut Lumière et Matière.

Ces remerciements auraient un goût d'inachevé si je ne remerciais pas l'ensemble de ma famille qui compte beaucoup pour moi. En particulier, un grand merci à ma jolie Bénédicte pour son soutien quotidien, ainsi qu'à mes parents et ma soeur qui sont des gens formidables.

Table des matières

Chapitre 1

Introduction

1.1 Cadre et motivations

Quel sentiment prédominait à la vue du colosse de Rhodes et comment ont été construites les pyramides d'Egypte ? Ces questions avaient une réponse dans le passé, mais elles sont aujourd'hui oubliées. Il reste peu d'informations de l'antiquité car les hommes n'avaient pas la technologie nécessaire pour protéger, stocker et diffuser son histoire et ses connaissances aux générations actuelles et futures. Dans ce mémoire, nous travaillons à notre échelle, à l'élaboration des outils contemporains permettant de représenter un monument, un objet, une personne ou même une scène complexe : l'infographie, la modélisation géométrique et la visualisation 3D. Ces branches de l'informatique utilisent une description de type *géométrique* basée sur des primitives telles que des points, des polygones ou des sphères, positionnés dans un espace de dimension 3. Mais ces outils ne se limitent pas à la visualisation. Ils permettent aussi la création et la manipulation de formes, et leurs applications sont nombreuses : conception assistée par ordinateur, la modélisation de terrains, la simulation, le calcul par éléments finis, la réalité augmentée, l'assistance à l'homme pour des manipulations complexes ou encore les loisirs numériques et le divertissement (cinéma, jeux vidéo, *etc.*). L'informatique graphique en général et la modélisation géométrique en particulier occupent ainsi une place centrale dans notre société.

La compression de données, une nécessité !

Dans ce mémoire, nous nous intéressons à la *compression* des formes préalablement modélisées, c'est-à-dire à l'opération consistant à réduire la taille d'une description, de sorte que l'opération inverse, la *décompression*, permette de restituer la description originale. Même si l'évolution des technologies permet de stocker des volumes d'information toujours plus grands et de les diffuser de plus en plus vite, le besoin d'algorithmes efficaces pour la compression reste entier. En effet, la quantité d'information à stocker croît généralement plus vite que la taille des mémoires. Les raisons sont nombreuses :

– La taille des structures géométriques est en constante augmentation car il existe une volonté de décupler la précision des objets pour améliorer la qualité du rendu ou la fiabilité des simulations ; par ailleurs, de plus en plus d'informations annexes sont associées aux objets représentés.

– L'informatique graphique en général et la modélisation géométrique en particulier sont utilisées dans des domaines applicatifs toujours plus nombreux, en science comme dans l'industrie, et l'on observe une augmentation permanente de la quantité d'objets manipulés.

De plus, au sein d'un ordinateur, toutes les mémoires ne sont pas équivalentes en terme de capacité et d'efficacité. Certaines représentations sont trop volumineuses pour être exploitées en mémoire vive. Par ailleurs, la modélisation géométrique s'applique aussi sur des appareils possédant une capacité de calcul et d'affichage performante mais un espace mémoire beaucoup plus faible qu'un ordinateur, tels qu'une tablette numérique ou un téléphone portable par exemple. Enfin, plus la taille d'une information est compacte et plus sa transmission est rapide à travers les réseaux de télécommunication. La compression apparaît donc comme un outil permettant de réorganiser les données de manière optimale afin de proposer une solution algorithmique efficace aux problèmes de stockage et de transmission.

De nombreuses approches différentes de la compression.

Il existe plusieurs types de compression ; on distingue tout d'abord la compression *sans perte* de la compression *avec perte*. Dans le premier cas, l'algorithme restitue, après les opérations successives de compression et de décompression, une information strictement identique à l'originale. Ce type de compression est adapté en particulier aux fichiers exécutables et aux données sensibles comme les images médicales. En ce qui concerne la compression avec perte, l'information restituée n'est pas strictement identique à l'information originale, mais elle en est voisine, et les méthodes permettent généralement de quantifier et de borner l'erreur commise. Ces algorithmes sont souvent utilisés pour compresser des sons, des images et des vidéos.

Dans le cas de la compression d'objets 3D, on distingue également les algorithmes dits *mono-résolution* des algorithmes *multi-résolution*. Dans le premier cas, la forme globale du modèle original est accessible uniquement à la fin de la décompression, alors qu'en *multi-résolution*, l'objet apparaîtra avec un faible niveau de détail dès le début de la décompression, puis sera progressivement raffiné jusqu'à épouser la forme originale en fin de décompression. Les contraintes imposées dans la seconde famille de méthodes rendent généralement l'algorithme moins efficace en terme de taux de compression.

Une bibliographie des principales approches de compression sera rappelée en début du chapitre 5 plus spécifiquement consacré à ces aspects.

Dans ce mémoire, nous travaillons sur des représentations de formes 3D correspondant à des *triangulations surfaciques* (*cf* figure 1.1). Ce sont des structures de données particulières et la conception d'algorithmes de compression adaptés doit tenir compte de leurs propriétés géométriques.

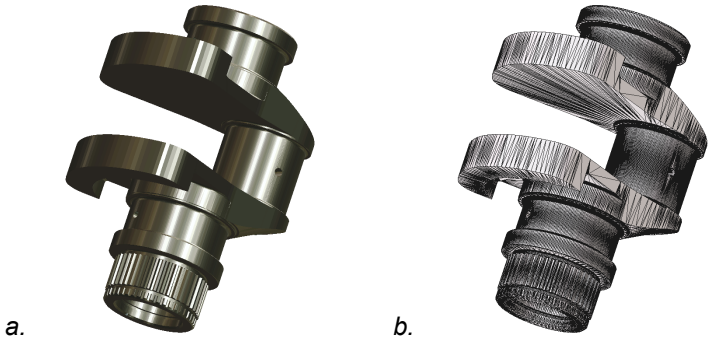


FIGURE 1.1 – En a) une forme représentée en 3D et en b), le maillage correspondant. L'objet a été développé par CAO.

Étude de la relation entre des connectivités différentes.

Dans ce qui suit, nous n'allons pas développer de nouvelles approches complètes pour compresser des surfaces triangulées, mais apporter des contributions en terme de transformation compacte de connectivité entre deux maillages à un même jeu de sommets. Nous n'aborderons pas le déplacement des sommets dans l'espace ambiant (ce problème a déjà été abordé par de nombreux auteurs) mais nous nous focaliserons sur les modifications concernant le graphe sous-jacent des triangulations, appelé le plus souvent *connectivité*, parfois *topologie*. Nous voyons plusieurs applications à une telle transformation :

– À partir de procédés utilisés dans les algorithmes de compression avec perte, il est possible de générer un maillage très proche du maillage original avec un coût bien inférieur aux algorithmes sans perte. Nous pouvons donc générer de nouveaux algorithmes mono-résolution et multi-résolution, en déterminant une transformation compacte entre le maillage obtenu et celui souhaité, et combiner le résultat au fichier compressé. Cette méthode est intéressante si le maillage souhaité a une connectivité très "proches" du maillage obtenu. Elle peut être interprétée comme une manière de corriger un maillage à faible coût.

– L'utilisation de transformations compactes est aussi intéressante pour compresser un *ensemble de maillages*. En effet, plutôt que de coder tous les maillages de l'ensemble indépendamment, il peut être intéressant de coder certains maillages "clés" par un algorithme de compression géométrique mono-résolution classique et les autres par des transformations à partir des maillages clés. Cela nécessite de mettre en relation des maillages en fonction de leur distance relative (par rapport à la transformation compacte envisagée), puisque il ne sera intéressant d'utiliser une transformation compacte que lorsque deux maillages sont assez "proche". Cette approche peut alors

améliorer le coût de codage d'un ensemble de maillages (*cf* figure 1.2).

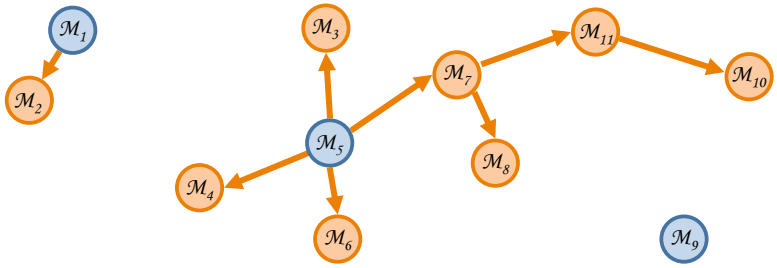


FIGURE 1.2 – Graphe orienté montrant les relations qui pourraient être impliquées dans le codage d'un ensemble de maillages M_i. La distance entre deux sommets sur la figure est proportionnelle au coût du codage de la transformation entre les maillages correspondants. On en déduit donc un ensemble de graphes orientés où les sommets en bleu représentent des maillages codés par un algorithme classique de compression géométrique et les sommets en orange représentent des maillages obtenus par transformation du maillage.

– Une transformation compacte prenant en compte des modifications de connectivité peut ouvrir des perspectives dans la compression de *séquences de maillages*. C'est un cas particulier d'ensemble de maillages composé d'une liste ordonnée de maillages, principalement utilisé pour modéliser un objet animé. Cette séquence peut être de deux types : *contrainte* (et appelée aussi maillage *dynamique*), ou *non contrainte*. Dans le premier cas, la connectivité est la même tout au long de la séquence et ne nécessite pas de codage de son évolution, alors que les séquences non contraintes autorisent des changements de connectivité entre deux maillages successifs (*cf* figure 1.3).

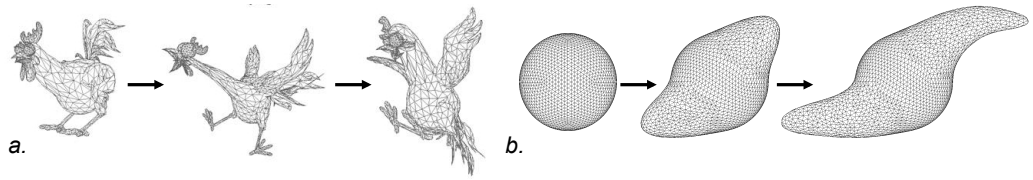


FIGURE 1.3 – Deux exemples de séquences de maillages : une séquence contrainte en a) et une séquence non contrainte en b).

De nombreux travaux ont été réalisés sur la compression de séquences, car l'utilisation de la cohérence temporelle entre maillages permet de prédire la position des sommets. Cependant, aucune méthode codant efficacement une modification de connectivité n'a été portée à notre connaissance et les auteurs se focalisent sur la compression d'une séquence contrainte [Yang 2002, Ibarria 2003, Sattler 2005] [Muller 2005, Payan 2005, Váša 2007, Váša 2009]. Un outil de modification

compacte de connectivité peut donc permettre de généraliser les résultats aux séquences dans lesquelles deux maillages consécutifs peuvent être moins "proches" et posséder plusieurs perturbations au niveau de la connectivité (*cf* figure 1.4).

FIGURE 1.4 – Représentation sous forme de graphes orientés d'une séquence de maillages. La distance entre les sommets correspond au coût du codage d'une transformation.

1.2 Transformation compacte par séquence de bascules d'arêtes entre deux triangulations

Actuellement, nous n'avons pas connaissance de travaux abordant le concept de transformation compacte de la connectivité entre deux triangulations. En toute généralité, la principale difficulté est de réussir à proposer une méthode de modification de connectivité, avec un coût mémoire pouvant concurrencer un algorithme de compression mono-résolution. Autrement dit, étant données deux triangulations T_1 et T_2 possédant un même jeu de sommets, une transformation de T_1 vers T_2 est compacte, si le coût du codage de la connectivité de T_2 à partir de la connectivité de T_1, est inférieur au codage de T_2 par un algorithme mono-résolution.

Pour répondre à ce problème, nous avons choisi de privilégier une opération locale de modification de connectivité : la bascule d'arête (appelée aussi *edge flip* ou *edge swap*). Ce choix est motivé par le postulat que notre approche ne sera utilisée que pour coder des évolutions très locales de la connectivité entre deux maillages.

Étant donnée (bc) une arête incidente à deux triangles orientés (abc) et (cbd), la *bascule* de l'arête (bc) supprime l'arête (bc) et la remplace par l'arête (ad). Les triangles orientés (abc) et (cbd) sont remplacées par (abd) et (adc) (*cf* figure 1.5).

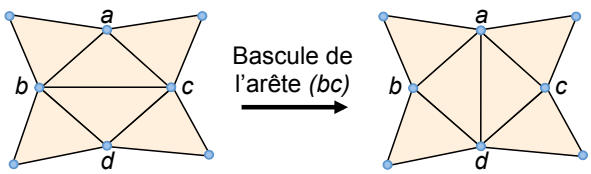


FIGURE 1.5 – L'opération de bascule de l'arête (bc).

La bascule d'arête est également un outil adapté pour générer une transformation compacte à d'autres titres. C'est une opération qui peut se coder simplement en désignant une arête, et lorsque deux triangulations ont le même nombre de sommets

et le même genre topologique, nous verrons qu'il n'est pas toujours nécessaire de l'accompagner d'autres opérations de modification de connectivité. De plus, nous montrerons qu'une séquence composée uniquement de bascules d'arêtes, possède des propriétés combinatoires permettant de la réduire jusqu'à une taille raisonnable.

À titre de comparaison, nous aurions pu utiliser la méthode d'insertion d'arête au sein d'une triangulation présentée par Bern *et al.* [Bern 1993]. Dans ce cas, la construction d'une arête (ab), nécessite le codage des sommets a et b et d'une zone de la triangulation dans laquelle sera construite l'arête, ainsi qu'une heuristique de re-triangulation de la région perturbée (*cf* figure 1.6). Cependant, le codage d'une insertion semble couteux, et la détermination d'une séquence d'insertions d'arête de petite taille, paraît difficile à concevoir.

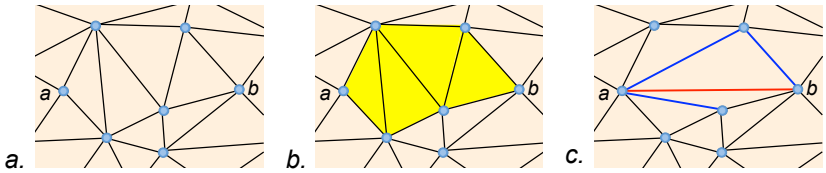


FIGURE 1.6 – Construction d'une arête (ab). En b), détermination d'une zone contenant l'arête (ab) (en jaune). En c), insertion de l'arête (ab) (en rouge) et re-triangulation à partir des arêtes restantes de la zone sélectionnée (arêtes en bleues).

Nous aurions aussi pu utiliser une méthode générant des séquences composées à la fois d'ajouts de sommets, de retraits de sommets et de bascules d'arêtes. Cependant, désigner l'opération à choisir au sein d'une séquence aura un coût non négligeable. Si l'on souhaite construire une transformation entre deux triangulations, la meilleure solution semble être de séparer la séquence contenant les ajouts ou retraits de sommets et celle contenant les bascules d'arêtes. Dans ce cas, la désignation d'une opération de modification de connectivité devient beaucoup moins coûteuse, et le problème se ramène là encore à la détermination d'une séquence de bascules d'arêtes possédant la plus petite taille possible.

1.3 Contributions à l'élaboration d'une transformation compacte

La construction d'une transformation compacte à partir d'une séquence de bascules d'arêtes, nécessite la résolution de trois problèmes correspondant aux trois principales contributions de la thèse.

• La première contribution concerne la détermination d'une séquence de bascules d'arêtes entre deux triangulations connexes, possédant un même nombre de sommets et un même genre topologique. L'originalisé de notre approche est de

travailler avec des triangulations appartenant à une classe assez large dans laquelle les possibilités de passage d'une triangulation à une autre seront plus grandes que dans une classe plus restreinte. Il s'agit de la *la classe des triangulations orientées combinatoirement manifold*, introduit dans le chapitre 2. Nous énonçons alors les conditions nécessaires et suffisantes assurant l'existence d'une séquence de bascules d'arêtes entre deux triangulations données. Nous présentons aussi un algorithme polynomial permettant de déterminer une séquence de bascules d'arêtes sans avoir à connaître le genre topologique commun.

• L'utilisation des bascules d'arêtes dans des algorithmes de compression, nécessite de travailler avec des séquences dont la longueur est la plus petite possible. L'approche que nous proposons dans le chapitre 4 est fondée sur une affectation d'indice à chaque arête, avec transfert de l'indice d'une arête détruite sur l'arête créée lors d'une bascule d'arête. Ceci donne un moyen d'identifier et de suivre une arête pendant une séquence de bascules et permet le développement d'outils simples capables d'augmenter ou de diminuer la longueur d'une séquence. Nous en déduisons un algorithme polynomial de réduction de séquences de bascules d'arête, qui, s'il ne garantit pas la minimisation optimale, garantit néanmoins une optimisation locale de sa longueur.

• Notre dernière contribution concerne enfin l'élaboration de méthodes de codage d'une séquence de bascules d'arêtes avec pour objectif de générer la plus petite séquence de bits possible. Le principe des différentes méthodes de codage introduites est basée sur une réorganisation des bascules d'une séquence, en n sous-ensembles tels que :

- chaque sous-ensemble est composé de bascules qui commutent deux à deux,
- chaque arête basculée dans un sous-ensemble est voisine dans la triangulation d'au moins une arête basculéee dans le sous-ensemble précédent,

Des résultats expérimentaux montrent que dans certains cas, notre transformation peut concurrencer les algorithmes de compression mono-résolution, et nous l'utiliserons pour élaborer une nouvelle version de l'algorithme de compression multi-résolution *IPR* [Valette 2009]. Enfin, nous associons la transformation compacte avec de nouveaux outils de modification de connectivité, pour coder la différence de connectivité entre deux triangulations ne possédant pas forcément le même nombre de sommets et le même genre topologique.

Les aspects bibliographiques à mettre en relation avec chacunes des contributions seront intégrés dans chacun des chapitres correspondants.

Chapitre 2

Classes de triangulations et bascule d'arête associée

Dans l'introduction, nous nous sommes intéressés aux maillages correspondant à des surfaces triangulées, mais nous allons ensuite faire évoluer ces triangulations au sein de classes assez larges qui méritent d'être spécifiées de manière plus exacte. Le lecteur pourra se référer à l'annexe B, si certaines notions de géométrie ou concernant les graphes ne sont pas connues.

L'ensemble des triangulations avec lesquelles nous travaillons sont des triangulations de dimension 2. Nous dirons qu'une classe de triangulations est de type *combinatoire* si la bascule d'arête est conditionnée exclusivement par des critères liés à la connectivité. Si la bascule d'arête est conditionnée par des critères géométriques, nous dirons qu'elle est de type *géométrique*.

2.1 Classes de triangulations de type combinatoire

2.1.1 Classe générale

Nous présentons une première classe de triangulations qui a la particularité d'inclure toutes les classes suivantes. Les résultats présentés dans le chapitre 4 sont suffisamment généraux pour pouvoir s'appliquer à des triangulations de cette classe.

2.1.1.1 Définition

Les *triangulations combinatoires* au sens le plus général considéré dans cette thèse, sont constituées d'un ensemble de sommets, d'arêtes et de faces satisfaisant les conditions suivantes :

1. chaque face est bordée par un cycle d'arêtes de longueur 3 ;
2. chaque arête est incidente à deux sommets possiblement identiques. Une arête incidente à deux sommets identiques est une *boucle*.
3. chaque arête est incidente à au plus deux faces pouvant être identiques. Une arête incidente à une seule face est appelée arête de *bord* ;

Nous notons $\mathcal{T}$ la classe de ces triangulations combinatoires (*cf* figure 2.1). Elle est souvent utilisée en mathématiques avec une approche beaucoup plus formelle sous le nom de Δ-complexe [Hatcher 2002]. En particulier, les triangulations introduites par Bobenko sont des Δ-complexe avec un plongement particulier [Bobenko 2007].

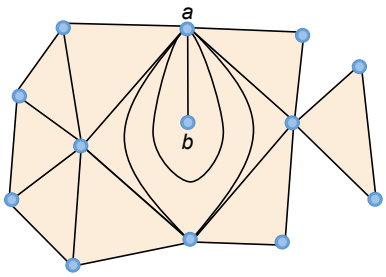


FIGURE 2.1 – Exemple d'une triangulation combinatoire. Elle possède une boucle et des arêtes multiples. L'arête (ab) ainsi que les arêtes de bord ne sont pas basculables.

2.1.1.2 Bascule d'arête associée

On munit la classe $\mathcal{T}$ de l'opération de bascule d'arête définie en 1.2. Cette opération peut-être appliquée à toute arête incidente à des faces distinctes. En particulier, nous interdisons donc les bascules d'arêtes de bord qui n'ont pas de sens. Lorsque la bascule d'une arête (ab) n'est pas possible, nous dirons que l'arête (ab) *n'est pas basculable*.

2.1.2 Triangulations orientées combinatoirement *manifold*

Nous introduisons à présent une classe de triangulations plus restreinte qui aura un rôle central dans le manuscrit. Tous nos résultats du chapitre 3 sur la détermination de séquences de bascules d'arêtes, nécessite l'exploitation des configurations particulières offertes par cette classe. Il s'agit d'une sous-classe de la classe de triangulations précédente.

2.1.2.1 Définition

Les *triangulations orientées combinatoirement manifold*, notées $\mathbb{T}$, sont constituées d'un ensemble de sommets, d'arêtes et de faces satisfaisant les conditions suivantes :

1. chaque face est bordée par un cycle orienté d'arêtes de longueur 3 et incidente à trois sommets différents ;
2. chaque arête (ab) est incidente à deux sommets distincts a et b et à une ou deux faces orientées de manières cohérentes (abc) et (bad). Une arête incidente à une seule face est appelée arête de *bord* ;
3. en considérant l'adjacence des faces orientées incidentes à un même sommet non bord (c'est-à-dire non incident à une arête de bord), on obtient un cycle de faces (et d'arêtes) que l'on désignera par la suite comme *un disque topologique combinatoire*. Dans le cas d'un sommet de bord, on impose la présence d'un seul demi-disque topologique.

Dans la littérature, il existe une classe de triangulations autorisant les arêtes multiples mais pas les boucles, qui se nomme les *simplicial posets* [Björner 1984, Stanly 1991]. Les triangulations orientées combinatoirement *manifold* peuvent être caractérisées comme des *simplicial posets* dont la surface est orientable, pouvant contenir des bords et dont les sommets sont plongés dans $\mathbb{E}^2$ ou $\mathbb{E}^3$.

Remarques :

• Si l'on choisit de plonger les sommets, aucune contrainte de non intersection n'est imposée sur les faces et les arêtes qui demeurent des objets purement combinatoires (*cf* figure 2.2).

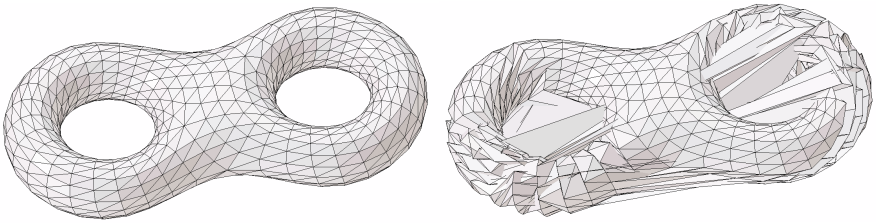


FIGURE 2.2 – Exemple de deux maillages identiques de $\mathbb{T}$ avec deux plongements différents de leurs sommets (elles ont donc même nombre de sommets, genre topologiques, et sont tous les deux orientés).

• La classe de triangulations étudiée permet à deux faces d'être incidentes à plus de deux sommets ou arêtes communes. De même, la condition (2) n'interdit pas à deux sommets a et b d'être incidents à plus de deux arêtes (ab) (il n'y a pas de limites). On peut donc rencontrer plusieurs faces incidentes à un même triplet de sommets (*cf* figure 2.3).

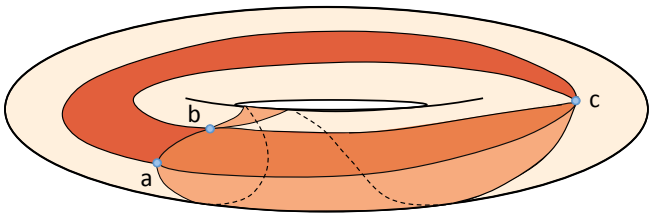


FIGURE 2.3 – Dans cette triangulation du tore, on peut trouver au moins trois faces orientées incidentes à un même triplet de sommets : (acb), (abc), et (cab).

2.1.2.2 Bascule d'arête associée

Dans la classe des triangulations orientées combinatoirement *manifold*, la bascule d'une arête appartenant à une triangulation $T \in \mathbb{T}$, est interdite lorsque son image n'appartient pas à $\mathbb{T}$. Nous interdisons donc la bascule de (ab) lorsqu'elle génère

une arête connectant un sommet c à lui-même, ce qui est équivalent à interdire de basculer l'arête (ab) lorsqu'elle est adjacente à deux faces incidentes au même triplet de sommets a, b et c (*cf* figure 2.4).

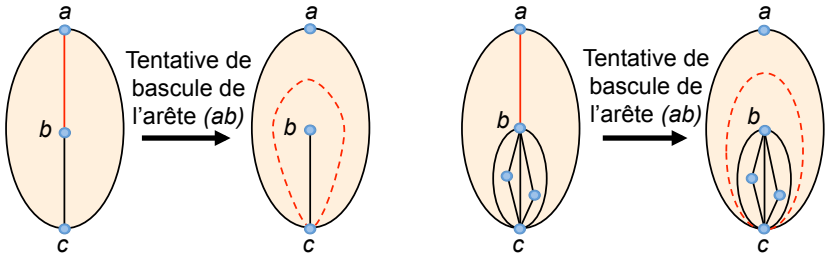


FIGURE 2.4 – Deux exemples d'arêtes non basculables dans $\mathbb{T}$.

En pratique, on préfère ne pas visualiser les arêtes comme des segments, mais plutôt des courbes qui respectent la position relative des arêtes dans les disques topologiques combinatoires associés à leurs extrémités (*cf* figure 2.5).

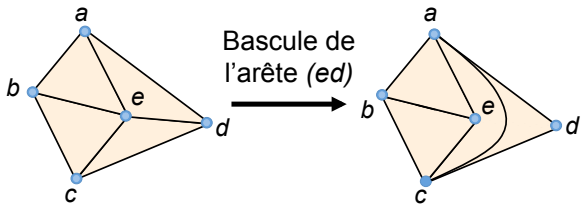


FIGURE 2.5 – L'arête (ac) résultante est comprise entre les (ae) et (ad) autour du sommet a ainsi que (ec) et (ed) autour de c.

2.1.3 Triangulations de Wagner

Une autre classe de triangulations a permis la détermination de résultats importants sur l'établissement de séquences de bascules d'arêtes. Il s'agit de la classe introduite par Wagner [Wagner 1936]. Cette classe est incluse dans la classe générale des triangulations combinatoires et se distingue des triangulations orientées combinatoirement *manifold* par le fait qu'elle n'autorise pas les faces à partager plus d'une arête commune, mais elle permet en revanche à un sommet d'être partagé par plusieurs cycles de faces incidentes à un même sommet. Les résultats établis dans le chapitre 3 ne peuvent être généralisés à cette classe.

2.1.3.1 Définition

Les *triangulations de Wagner* sont constituées d'un ensemble de sommets, d'arêtes et de faces satisfaisant les conditions suivantes :

1. chaque face est bordée par un cycle d'arêtes de longueur 3 et possède au maximum une arête incidente commune ;
2. chaque arête est incidente à deux sommets distincts et exactement deux faces ;
3. L'adjacence entre faces autour d'un sommet permet de définir un ou plusieurs cycles de faces et d'arêtes.

On remarque que les sommets ne sont pas nécessairement plongés et que les triangulations n'ont pas de bords.

2.1.3.2 Bascule d'arête associée

Les bascules d'arêtes sont conditionnées uniquement par la connectivité de la triangulation (comme dans notre classe de triangulations orientées combinatoirement *manifold*, le plongement éventuel des sommets n'est pas déterminé). La bascule d'une arête est interdite lorsque celle-ci engendre une triangulation avec deux faces incidentes partageant deux arêtes communes.

2.1.3.3 Relation avec la classe des triangulations orientées combinatoirement *manifold*

Etant donnée une triangulation T de Wagner, il n'est pas toujours possible de trouver une triangulation orientée combinatoirement *manifold* T' telle que T et T' ont une même connectivité. C'est le cas lorsque T est une surface non orientée ou qu'elle possède un sommet dont l'ensemble des faces adjacentes ne correspond pas à un disque topologique combinatoire (*cf* figure 2.6). Inversement, une triangulation de la classe des triangulations orientées combinatoirement *manifold* peut contenir des sommets de valence deux, contrairement à celles de Wagner.

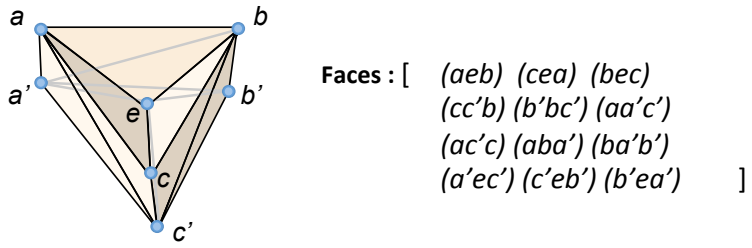


FIGURE 2.6 – Dans cette triangulation de Wagner, le sommet situé au centre ne possède pas de cône simple formé de ces faces adjacentes. Les faces (bea), (ceb) et (aec) incidentes à e sont incluses dans la triangulation mais aussi les faces $(a'eb')$, $(b'ec')$ et $(c'ea')$.

2.1.4 Triangulations combinatoires d'un polygone simple

Nous appelons triangulation d'un polygone simple, une triangulation de $\mathcal{T}$, de genre 0, contenant un seul bord et ne possédant aucun sommet interne. Toutes les

arêtes internes sont incidentes à des faces distinctes, donc elles sont toutes basculables. De plus, les triangulations d'un polygone simple forment une classe stable par bascule d'arête (*cf* figure 2.7).

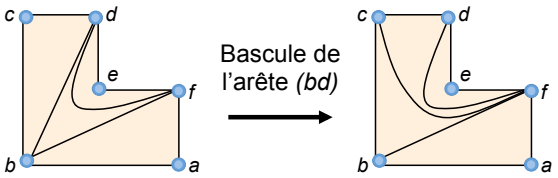


FIGURE 2.7 – Représentation d'un polygone simple triangulé.

2.2 Classes de triangulations dites géométriques

2.2.1 Triangulations géométriques orientées

Nous présentons maintenant une classe de triangulations dans laquelle le plongement géométrique joue un rôle important avec l'établissement d'une contrainte de non intersection qui conditionne également la possibilité d'une bascule d'arête. Cette classe peut être vue comme une spécialisation de la classe des triangulations orientées combinatoirement *manifold*.

2.2.1.1 Définition

Les *triangulations géométriques orientées*, sont constituées d'un ensemble de sommets, d'arêtes et de triangles satisfaisant les conditions suivantes :

1. les sommets sont plongés (ou plongeables) dans un espace euclidien $\mathbb{E}^2$ ou $\mathbb{E}^3$;
2. chaque arête est incidente à deux sommets distincts et à un ou deux triangles. Une arête incidente à un seul triangle est appelée arête de *bord* ;
3. tout couple de triangles partage au plus un sommet ou une arête (et ses deux extrémités) ;
4. pour chaque sommet, l'ensemble des arêtes opposées à ce sommet dans l'ensemble des triangles incidents forme une chaîne polygonale simple fermé et orienté, appelé *link*, sauf si le sommet est un bord auquel cas l'ensemble des arêtes constitue une unique chaîne polygonale simple ouverte. Les faces incidentes à un sommet non bord constituent un voisinage homéomorphe à un disque ou à un unique demi-disque topologique dans le cas du bord ;

Cette classe de triangulation peut également être définie par la classe des *surfaces simpliciales orientées* [Spanier 1994].

Dans le cas où les sommets sont plongés dans $\mathbb{E}^2$, on retrouve la classe des triangulations planaires. Dans le cas où les sommets sont plongeables dans $\mathbb{E}^3$ et où les triangulations sont connexes et fermées, on retrouve les triangulations de surfaces fermées définies par Giblin [Giblin 2010].

2.2.1.2 Bascule d'arête associée

Dans cette classe, c'est la géométrie du maillage qui détermine si une arête est basculable ou non. Une arête n'est pas basculable si la bascule génère une triangulation violant la propriété (3) des triangulations géométriques.

2.2.1.3 Relation avec la classe des triangulations orientées combinatoirement *manifold*

Le lemme suivant présente les relations entre les triangulations géométriques orientées et la classe des triangulations orientées combinatoirement *manifold* :

Lemme II.1 : Nous pouvons représenter n'importe quelle triangulation géométrique orientée comme une triangulation orientée combinatoirement *manifold*.

Démonstration. Soit T une triangulation géométrique orientée. Le *link* d'un sommet permet de définir un cycle orienté de faces deux à deux adjacentes autour de ces sommets. Nous obtenons alors un plongement de T dans la classe des triangulations orientées combinatoirement *manifold*. □

Nous pouvons donc transformer une triangulation géométrique orientée en une seconde de la même classe à partir des triangulations orientées combinatoirement *manifold* et de la bascule associée.

2.2.2 Triangulations d'un n-gone convexe

Les triangulations combinatoires d'un polygone simple possèdent leur analogue dans la classe des triangulations dite géométriques. Il s'agit des triangulations du n-gone convexe. Toutes les arêtes sont basculables à chaque instant car les triangles incidents à toute arête interne constituent un quadrilatère convexe.

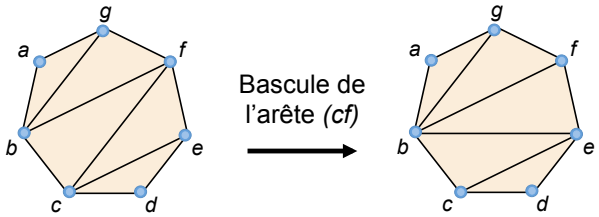


FIGURE 2.8 – Bascule d'arête appliquée à une triangulation d'un 7-gone convexe.

Les résultats présentés dans le chapitre 3 s'appliquent à des triangulations de la classe des triangulations orientées combinatoirement *manifold* et ne peuvent pas se généraliser en se restreignant au sein d'une classe plus restreinte. Les résultats du chapitre 4 s'appliquent dans la classe générale des triangulations combinatoires

mais peuvent aussi se restreindre au sein des autres classes de triangulations présentées. Enfin, dans le chapitre 5, tous les résultats ont été obtenus en travaillant dans la classe des triangulations orientées combinatoirement *manifold* munie de son opération de bascule d'arête.

Chapitre 3

Génération de séquences de bascules d'arêtes

Ce chapitre traite de la détermination d'une séquence de bascules d'arêtes entre deux triangulations partageant un même nombre de sommets et un même genre topologique. Le problème de leur existence a déjà été abordé dans la classe des triangulations géométriques et la classe des triangulations de Wagner. Cependant, il existe peu d'algorithmes de détermination de séquences. Ils ne permettent pas toujours de s'accorder avec une mise en correspondance préalable des sommets des deux triangulations, et se limitent généralement aux triangulations planes ou aux surfaces de genre 0.

Nous remédions à ces limitations en proposant une manière directe de déterminer une séquence de bascules d'arêtes entre deux triangulations orientées combinatoirement *manifold*, partageant un même nombre de sommets et un même genre topologique. Nous prenons également en compte une correspondance entre les sommets des deux maillages, qui, si elle est non fournie, peut être choisie de manière aléatoire, sauf pour les sommets adjacents à une arête de bord. Nous présentons les résultats suivants :

- Etant donnée une triangulation contenant deux sommets a et b, ainsi que des arêtes contraintes, que l'on interdit de basculer, nous présentons une approche directe pour tenter de construire une arête (ab). Dans un premier temps, nous cherchons à établir un chemin de faces reliant a à b et n'intersectant aucune arête contrainte. Lorsque cela est possible, nous construisons ensuite l'arête (ab) en combinant des bascules d'arêtes incidentes à deux faces consécutives du chemin, et des déplacements locaux de chemin.
- Etant données deux triangulations T_{init} et T_{cible} de $\mathbb{T}$, nous énonçons les conditions nécessaires et suffisantes assurant l'existence d'une séquence de bascules d'arêtes entre elles. T_{init} et T_{cible} doivent être connexes, avoir le même nombre de sommets et le même genre topologique. Si elles possèdent des arêtes contraintes (dont les arêtes de bord), ces dernières doivent vérifier les conditions suivantes de bonne correspondance entre les deux triangulations :

  - les arêtes contraintes sont en quantité égale dans T_{init} et T_{cible} ;

  - pour chaque arête contrainte incidente aux sommets a et b de T_{init}, il existe une arête contrainte correspondante dans T_{cible}, incidente aux sommets de T_{cible} mis en correspondance avec a et b ;

  - l'ordre des arêtes contraintes dans le disque topologique combinatoire autour des sommets mis en correspondance, est identique ;

- les cycles d'arêtes contraintes ainsi mises en correspondance doivent avoir la même orientation dans T_{init} et T_{cible} ;

- Si deux sommets a et b sont adjacents dans une triangulation, il existe un chemin de faces entre les sommets correspondant dans la seconde triangulation ne traversant pas d'arêtes contraintes .

De manière itérative, il est alors possible de construire une séquence de bascules entre les deux triangulations T_{init} et T_{cible} en utilisant une stratégie que nous décrivons dans un algorithme très général.

– Etant données deux triangulations T_1 et T_2 satisfaisant les conditions précédentes, nous présentons ensuite un algorithme direct permettant de changer la connectivité de T_1 en celle de T_2 en utilisant une séquence de bascules d'arêtes, sans avoir à connaître le genre topologique commun. L'algorithme procède par construction successive des faces selon un mécanisme de croissance de régions. Nous offrons une preuve de la validité de cet algorithme et l'efficacité de l'approche est illustrée par des résultats expérimentaux.

Les résultats présentés dans ce chapitre exploitent la diversité des configurations offertes par la classe des triangulations orientées combinatoirement *manifold* et leur validité ne s'étend pas forcément à une classe de triangulation plus restreinte.

3.1 État de l'art sur la détermination de séquences de bascules d'arêtes

Nous avons divisé l'état de l'art en trois ensembles de triangulations : les triangulations d'un n-gone convexe, les triangulations géométriques orientées et les triangulations de la classe de Wagner.

3.1.1 Résultats sur les triangulations d'un n-gone convexe

Etant données deux triangulation d'un n-gone convexe, il existe toujours une séquence de bascules entre elles. En effet, Lucas et Hurtado [Lucas 1987, Hurtado 1999] montrent que le *graphe des bascules d'arêtes* (c'est-à-dire le graphe dont les sommets sont des triangulations et tel que deux sommets sont connectés s'il existe une bascule d'arête qui transforme le premier en le second) est hamiltonien, c'est-à-dire qu'il possède au moins un cycle d'arêtes passant une unique fois par tous les sommets.

En ce qui concerne la détermination d'une séquence, une méthode naïve consiste à construire le graphe de bascules d'arêtes du n-gone convexe et à déterminer un chemin reliant le sommet contenant la triangulation de départ à celui contenant la triangulation d'arrivée. Le graphe possédera $C_{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}$ sommets car il existe C_{n-2} triangulations différentes (correspondant au nombre de Catalan). Nous pouvons aussi procéder par deux méthodes itératives d'insertion de la connectivité

de T_{cible} dans T_{init} :

• **1ère méthode -** On utilise la propriété suivante : toutes les triangulations d'un n-gone convexe possèdent au minimum deux faces adjacentes et incidentes à deux arêtes de bord de la triangulation. Nous proposons donc un algorithme construisant itérativement les faces de T_{cible} dans T_{init} (*cf* algorithme 3.1).

Algorithme 1 Détermination d'une séquence de bascules par construction itérative des faces de T_{cible} dans T_{init}

tant que T_{init} et T_{cible} possèdent plus de trois sommets **faire**
  Soit une face (abc) dans T_{cible} telle que (ab) et (ac) sont des arêtes de bord
  Basculer toutes les arêtes incidentes à a dans T_{init}
  $T_{init} \leftarrow T_{init}$ privée de la face (abc)
  $T_{cible} \leftarrow T_{cible}$ privée de la face (abc)
fin tant que
Retourner la séquence des bascules d'arêtes utilisées

La figure 3.1 illustre cette méthode.

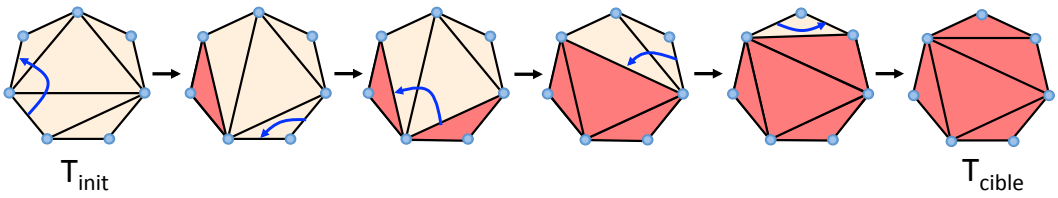


FIGURE 3.1 – Détermination d'une séquence de bascules en construisant successivement les faces souhaitées (en rouge, les faces construites). Les flèches en bleues montrent les arêtes basculées pour construire une nouvelle face.

• **2ème méthode -** On définit un *chemin de faces exploitables* entre deux sommets a et b comme une séquence de faces $F = (f_1, ..., f_q)$ vérifiant :

- le sommet a est incident à la face f_1, mais pas à f_2 ;
- le sommet b est incident à la face f_q, mais pas à f_{q-1} ;
- les faces f_i et f_{i+1} sont incidentes à une même arête ;
- $f_i \neq f_j$ si $i \neq j$.

On définit le *chemin d'arêtes* associé $E = (e_1, ..., e_{q-1})$, composé des arêtes adjacentes à f_i et f_{i+1}, c'est-à-dire $e_i = f_i \cap f_{i+1}$.

On utilise la propriété qu'il existe toujours un unique chemin de faces exploitables F reliant deux sommets a et b dans un n-gone convexe triangulé. Le basculement des arêtes du chemin d'arêtes E dans l'ordre (e_1, puis e_2, ..., e_{q-1}), permet d'insérer l'arête (ab). Nous en déduisons un algorithme construisant

itérativement les arêtes de T_{cible} dans T_{init} (*cf* algorithme 2) et la figure 3.2 illustre cette méthode. Les chemins de faces seront abordés plus en détail dans la partie 3.3.1.

Algorithme 2 Détermination d'une séquence de bascules par construction itérative des arêtes de T_{cible} dans T_{init}

tant que Il existe au moins une arête (ab) de T_{cible} non présente dans T_{init} **faire**
  Trouver un chemin de faces exploitable F dans T_{init} entre les sommets a et b
  Basculer dans l'ordre les arêtes du chemin associé E
fin tant que
Retourner la séquence des bascules d'arêtes utilisées

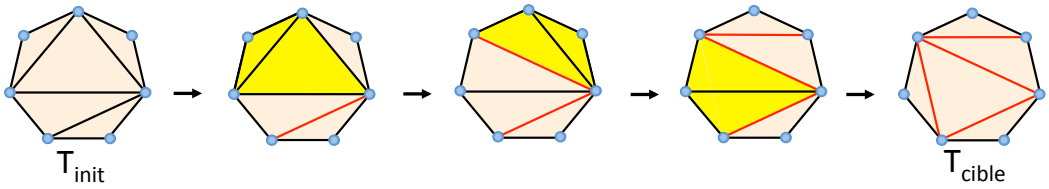


FIGURE 3.2 – Détermination d'une séquence de bascules à partir des faces incidentes à deux arêtes du bord de la triangulation. En jaune, les chemins de faces et en rouge, les arêtes construites.

Dans les cas suivants, la détermination de séquences sera beaucoup plus délicate car les triangulations possèderont des arêtes non basculables, et il n'y aura plus unicité des chemins de faces exploitables entre deux sommets.

3.1.2 Résultats sur les triangulations géométriques orientées

Dans ce cas, il n'y a aucune application de mise en correspondance à définir car chaque sommet de T_{init} est mis en correspondance avec le sommet de T_{cible} possédant la même géométrie. Nous supposerons donc qu'il n'y a aucun point confondu dans les deux triangulations.

Nous présentons dans un premier temps les méthodes qui font appel à une triangulation appelée *pivot* notée T_{pivot}. La séquence de bascules recherchée sera la concaténation de la séquence entre T_{init} et T_{pivot} et de la séquence entre T_{pivot} et T_{cible} (qui est équivalente à la séquence entre T_{cible} et T_{pivot} dont on inverse l'ordre des bascules). La triangulation pivot doit pouvoir être générée à partir de n'importe quelle triangulation partageant le même nuage de points.

Dans le cas des triangulations planaires (les sommets sont plongés dans $\mathbb{R}^2$), nous présentons deux algorithmes permettant de générer une triangulation pivot :

• **À partir d'une triangulation de référence** [Lawson 1972] **-** Pour $m > 0$, on définit un ensemble ordonné de points $V_m = \{v_1, v_2, ..., v_m\}$ tel que v_m est un point n'appartenant pas à l'enveloppe convexe de V_{m-1}, v_{m-1} n'appartient pas à l'enveloppe convexe de V_{m-2} et ainsi de suite. En supposant que le nuage de points possède s sommets, on définit une *triangulation de référence* de V_s de la manière suivante :

- la triangulation de référence de V_2 est simplement composée d'une arête (v_1v_2) ;

- étant donnée une triangulation de référence T_{i-1} correspondant à V_{i-1}, on construit la triangulation de référence T_i de $V_i = V_{i-1} \cup v_i$ en insérant v_i ainsi que toutes les arêtes pouvant relier v_i à un sommet du bord de T_{i-1} sans créer d'intersections avec les arêtes de T_{i-1}.

On remarque qu'il existe généralement plusieurs triangulations de référence d'un même nuage de points en fonction de l'ensemble ordonné de points.

Soient T_{pivot} une triangulation de référence déterminée à partir de l'ensemble ordonné de points V_s et T une triangulation quelconque de V_s. Nous allons chercher à transformer T en T_{pivot} en reliant le sommet v_s uniquement aux sommets de V_{s-1} voisins dans T_{pivot}, puis le sommet v_{s-1} aux sommets de V_{s-2} voisins dans T_{pivot} et ainsi de suite. Supposons que cela soit effectué pour les sommets v_s, v_{s-1}, ..., v_{i+1}, alors on relie le sommet v_i aux sommets de V_{i-1} voisins dans T_{pivot}, simplement en basculant les arêtes incidentes à v_i non présentes dans T_{pivot}. Lawson démontre que ces arêtes sont toujours basculables et qu'il restera les arêtes souhaitées. On en déduit l'algorithme 3 permettant de construire une triangulaion de référence à partir de bascules d'arêtes.

Algorithme 3 Construction d'une triangulation de référence

$i \leftarrow s$
tant que $i \neq 0$ **faire**
  Basculer toutes les arêtes incidentes à v_i et non présentes dans T_{pivot}
  $i \leftarrow i - 1$
fin tant que

• **À partir d'une triangulation de Delaunay** [Lawson 1977] **-** La seconde méthode consiste à construire une triangulation dite de *Delaunay* à partir de bascules d'arêtes. Étant donnée une triangulation T, on dira qu'une arête (ab) est *illégale* si le cercle circonscrit à la face adjacente (abc) contient le sommet d (en son intérieur), ou si celui de la face adjacente (abd) contient le sommet c. Dans le cas inverse, on dira que l'arête (ab) est *légale*. Une *triangulation de Delaunay* d'un ensemble de points S est une triangulation de S sans arête illégale.

Il est aussi possible de détecter qu'une arête (ab) est illégale en vérifiant que la somme des angles opposés à (ab) au sein des deux faces adjacentes est supérieure

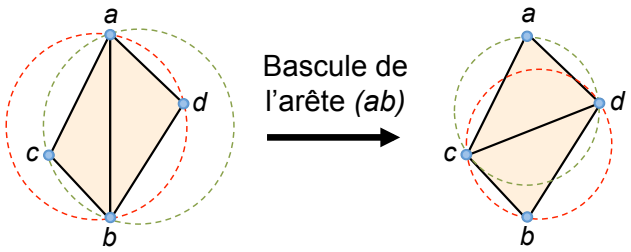


FIGURE 3.3 – Basculer une arête illégale génère une nouvelle arête légale. Les cercles en pointillés représentent les cercles circonscrits aux faces.

à π. La bascule d'arête remplace une arête illégale par une arête légale (*cf* figure 3.3), et Lawson montre que quel que soit l'ordre des arêtes illégales basculées, la triangulation converge vers une triangulation de Delaunay. Il propose un algorithme utilisant $O(a^2)$ bascules d'arêtes, où a est le nombre d'arêtes (*cf* algorithme 4).

Algorithme 4 Construction d'une triangulation de Delaunay

```
Insérer toutes les arêtes de la triangulation initiale dans une pile
tant que La pile n'est pas vide faire
  Dépiler la première arête (ab)
  si (ab) est illégale alors
    Empiler les quatres arêtes voisines de (ab)
    Basculer l'arête (ab)
  fin si
fin tant que
```

Cependant, la triangulation de Delaunay n'est pas toujours unique. Dans le cas où les sommets a, b, c et d sont *cocycliques* (c'est-à-dire qu'ils appartiennent au même cercle), les arêtes (ab) et (cd) sont légales ; ce qui entraine l'existence d'au moins deux triangulations de Delaunay différentes. L'utilisation de l'algorithme 4 engendrant une triangulation pivot entre T_{init} et T_{cible} nécessite que la triangulation de Delaunay résultante soit unique. On peut par exemple définir une relation d'ordre entre les points telle que pour deux sommets a et b dans $\mathbb{E}^2$, de coordonnées (x_a, y_a) et (x_b, y_b), on a $a < b$ si $x_a < x_b$ ou si $x_a = x_b$ et $y_a < y_b$. Après la génération d'une triangulation de Delaunay, on peut alors imposer à tout quadruplet de sommets cocycliques que l'arête interne relie toujours le sommet supérieur par rapport à la relation d'ordre précédente.

Bobenko et Springborn [Bobenko 2007] présentent un résultat analogue à celui de Lawson dans le cas des surfaces dites *plates par morceaux*. On définit une surface plate par morceaux (S, d) comme une variété différentielle de dimension 2 notée S pouvant posséder des bords, et accompagnée d'une métrique d qui est plate sauf au niveau de points isolés, appelés *points-cône*, où d possède des singularités en forme

de cône. De la même manière que Lawson [Lawson 1977], nous pouvons construire une triangulation de Delaunay d'une surface plate par morceaux, et l'utiliser comme pivot pour déterminer une séquence de bascules d'arêtes entre deux triangulations d'une même surface plate par morceaux.

Nous dirons qu'une arête (ab) est illégale si après avoir déplié isométriquement les deux faces adjacentes pour les rendre coplanaires, celles-ci possèdent des sommets à l'intérieur de leur cercle circonscrit. Ce critère est équivalent à dire que la somme des angles opposés à (ab) au sein des deux triangles adjacents est supérieur à π. Nous pouvons ensuite déterminer une séquence de bascules d'arêtes entre une triangulation donnée d'une surface plate par morceaux et la triangulation de Delaunay correspondante par l'algorithme de Lawson 4. Indermitte *et al.* [Indermitte 2001] montrent que l'algorithme se termine après un nombre fini d'étapes. Ce résultat se généralise aux triangulations combinatoires avec un plongement géométrique particulier.

Hanke *et al.* [Hanke 1996] proposent un algorithme de détermination de séquence sans utiliser de triangulation pivot. Étant données T_1 et T_2 deux triangulations d'un même ensemble de points, l'idée consiste à les superposer afin que les sommets possédant la même géométrie se confondent. On introduit les notations $\#(T_1, T_2)$ désignant le nombre d'intersections d'arêtes entre T_1 et T_2, et $\#(e, T_2)$ où e est une arête de T_1, désignant le nombre d'intersections entre e et T_2.

Dans le cas du n-gone convexe, Sleator *et al.* [Sleator 1987] ont démontré que si la bascule d'une arête e de T_1 génère une arête de la triangulation T_2, c'est-à-dire que $\#(e, T_2) = 1$, alors il existe une séquence de bascules d'arêtes de longueur minimale entre T_1 et T_2 dont e est la première arête basculée. À partir de ce résultat, les auteurs ont développé un algorithme consistant à faire converger T_{init} et T_{cible} vers une même triangulation, en basculant à chaque étape une arête e d'une triangulation T_i telle que $\#(e, T_j) = 1$ avec $i \neq j$. Lorsqu'il n'existe aucune arête vérifiant cette dernière propriété, on choisit de basculer l'arête qui réduira au mieux le nombre d'intersections des deux triangulations courantes. On en déduit l'algorithme 5 suivant :

Si T_{init} et T_{cible} sont des triangulations d'un n-gone convexe, et si à chaque étape il existe toujours une arête e' dans T_i telle que $\#(e', T_j) = 1$, $i, j \in \{1, 2\}$, alors l'algorithme terminera correctement et génèrera une séquence minimale. Dans le cas contraire, l'algorithme peut ne pas converger.

3.1.3 Résultats sur les triangulations de Wagner

Les travaux abordés concernent des triangulations connexes dont les sommets ne sont pas indexés, c'est-à-dire que l'on insère une connectivité sans prendre en considération la mise en correspondance obtenue.

Le premier résultat est énoncé par Wagner [Wagner 1936]. Il introduit la classe de triangulations et montre qu'il existe une séquence de bascules d'arêtes entre

Algorithme 5 Détermination d'une séquence de bascules d'arêtes entre T_{init} et T_{cible} sans triangulation pivot

Posons $T_1 \leftarrow T_{init}$ et $T_2 \leftarrow T_{cible}$
tant que $T_1 \neq T_2$ **faire**
  tant que Il existe une arête e' dans T_i avec $\#(e', T_j) = 1$, $i, j \in \{1, 2\}$ **faire**
    Soit e l'arête de T_j intersectant e'
    Basculer l'arête e dans T_j
  fin tant que
  si $T_1 \neq T_2$ **alors**
    Soit e l'arête de T_i, $i \in \{1, 2\}$, telle que la bascule de e réduit au maximum $\#(T_i, T_j)$
    Basculer l'arête e
  fin si
fin tant que
Retourner la séquence des bascules d'arêtes utilisées

deux triangulations d'une sphère. Il prouve qu'il peut insérer la connectivité d'une *triangulation canonique* (*cf* figure 3.4) par des bascules d'arête, dans n'importe quelle triangulation.

Pour construire cette connectivité, on commence par choisir une face (abc) dans T, et on appelle T' la triangulation T privée de (abc). Si T' n'est pas réduite à une face, Wagner montre que l'on peut réduire la valence de n'importe quel sommet à 3 car il existe toujours une arête basculable parmi les arêtes incidentes à un sommet de valence supérieure à 3. Après avoir réduit la valence du sommet a à 3, c'est-à-dire que le sommet a est voisin aux sommets b, c et a', on continue au sein de la sous-triangulation T'' égale à T' privée des faces (aba') et (aca'), des arêtes (ab), (ac) et (aa'), et du sommet a. On réduit la valence du sommet a' à 3 et l'on continue l'opération jusqu'à ce que la sous-triangulation devienne une face. Au final, nous obtenons la triangulation canonique de la figure 3.4 (*cf* algorithme 6).

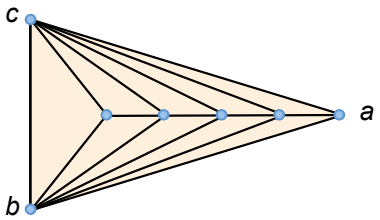


FIGURE 3.4 – Triangulation canonique introduite par Wagner privée de la face (abc).

L'existence de séquences de bascules d'arêtes entre les connectivités de T_{init} et T_{cible} a été généralisée dans le cas du tore par Dewdney [Dewdney 1973], et dans le cas du plan projectif et de la bouteille de Klein par Negami et Watanabe

Algorithme 6 Construction d'une triangulation canonique

Soit (abc) une face de la triangulation T
$T \leftarrow T$ privée de la face (abc)
tant que La valence du sommet $a > 2$ **faire**
  Réduire la valence du sommet a à 3
  On suppose a voisin de b, c et a'
  $T \leftarrow T$ privée des faces $(a'ba)$ et $(a'ac)$, des arêtes (ab), (ac) et (aa') et du sommet a
  $a \leftarrow a'$
fin tant que

[Negami 1990, Negami 1999]. Cependant, il est difficile d'en déduire un algorithme de détermination de séquences de bascules d'arêtes. Les preuves reposent sur la possibilité de déplacer chaque sommet de valence 3 dans n'importe quelle face d'une triangulation (*cf* figure 3.5). Si deux triangulations T_1 et T_2 possèdent le même nombre de sommets, le même genre topologique et qu'il existe une séquence de bascules d'arêtes transformant la connectivité de T_1 en T_2, alors il existe une séquence de bascules entre les triangulations T_1 et T_2 auxquelles on rajoute un sommet dans l'une des faces des deux triangulations. Nous appelons *triangulation minimale* une triangulation telle qu'on ne peut pas réduire la valence d'un de ses sommets à 3. Pour un genre topologique donné, s'il existe une unique triangulation minimale permettant de construire toutes les triangulations de genre g, alors il existe une séquence de bascules d'arêtes entre deux triangulations possédant le même nombre de sommets et le même genre g. Cependant, il est rarement possible de trouver une triangulation minimale permettant d'engendrer toutes les triangulations.

Negami généralise les résultats précédents [Negami 1993] en montrant que pour deux triangulations possédant le même genre topologique g et un même nombre de sommets N, il existe un entier N_g tel que si $N \geq N_g$, alors il existe une séquence de bascules d'arêtes. De plus, il démontre que $N_g = O(g^3)$ [Negami 1999].

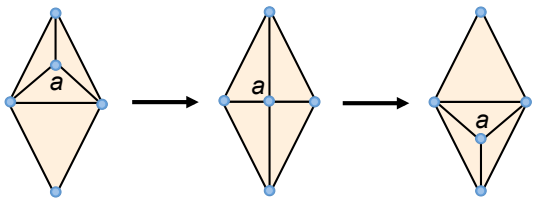


FIGURE 3.5 – Le sommet a est déplacé d'une face vers une face voisine par des bascules d'arêtes.

Nous présentons maintenant notre méthode de détermination d'une séquence de bascules d'arêtes entre deux triangulations orientées combinatoirement *manifold*

possédant le même nombre de sommets, de bords et le même genre topologique. Contrairement aux méthodes présentées, notre algorithme converge toujours, il génère des séquences de bascules d'arêtes plus courtes que les méthodes utilisant une triangulation pivot, et la mise en correspondante entre les sommets peut être choisie de manière quasiment aléatoire.

Soient T_{init} et T_{cible} deux triangulations de $\mathbb{T}$, nous présentons les conditions nécessaires pour assurer l'existence d'une séquence de bascules d'arêtes entre les deux connectivités. Afin de faciliter la lecture, les sommets en correspondance auront un même nom dans T_{init} et T_{cible}.

3.2 Conditions nécessaires pour permettre le passage d'une connectivité à une autre

On définit une arête *contrainte* comme une arête que l'on s'interdit de basculer et l'on caractérise les arêtes du bord comme des arêtes contraintes. Deux faces sont *connectées* si elles sont adjacentes à une même arête non contrainte, et l'on définit un *chemin de faces* comme une séquence ordonnée de faces deux à deux connectées. Une *composante connexe* est l'ensemble maximal de faces tel que pour chaque paire, il existe un chemin de faces.

Soit $T_{courant}$ une triangulation initialisée par T_{init}. Nous dirons qu'une composante connexe au sein de T_{cible} et une composante connexe au sein de $T_{courant}$ sont des *composantes correspondantes* si l'on peut insérer la connectivité de la première au sein de la seconde. Dans un premier temps, cela impose que les deux composantes possèdent le même nombre de sommets, le même genre topologique et des bords consistants. Cependant, ces conditions ne sont pas suffisantes et les deux composantes connexes doivent aussi satisfaire les critères suivants.

3.2.0.1 Hypothèses sur les faces appartenant aux composantes

Il est nécessaire que chaque composante connexe de $T_{courant}$ possède au moins une arête basculable pour faire converger sa connectivité (*cf* figure 3.6). Le lemme suivant donne les conditions nécessaires que doivent satisfaire deux composantes correspondantes.

Lemme III.1 : Dans le cas où une composante connexe de T_{cible} contient uniquement trois sommets a, b et c, il n'est pas toujours possible de faire converger la composante correspondante de $T_{courant}$ vers la connectivité souhaitée, en utilisant uniquement des bascules d'arêtes (et en restant dans la classe de triangulation $\mathbb{T}$). On ne rencontre pas cette situation de blocage lorsque cette composante connexe contient au plus deux faces.

Démonstration. Supposons que toutes les faces de la composante connexe sont adjacentes à un même triplet de sommets, alors toutes les arêtes internes ne sont pas basculables dans la classe de triangulation $\mathbb{T}$. Dans le cas où la composante connexe

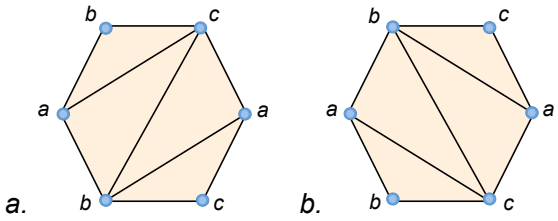


FIGURE 3.6 – Les deux triangulations sont différentes et partagent le même bord, mais il est impossible de transformer la première en la seconde par des bascules d'arêtes. En effet, aucune arête n'est basculable.

contient au plus deux faces, il existe seulement cinq triangulations possibles qui sont uniques et qui dépendent uniquement du nombre de faces et des arêtes du bord (*cf* figure 3.7).

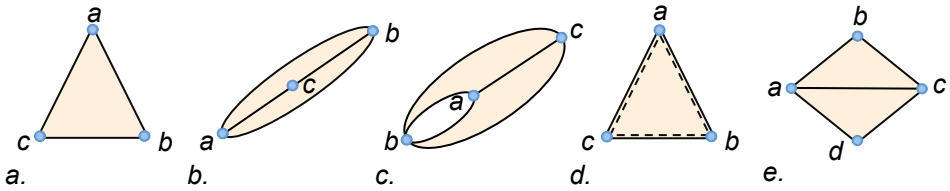


FIGURE 3.7 – Les différentes triangulations de $\mathbb{T}$ composées d'au plus deux faces : a) une face, trois arêtes de bord ; b) deux faces, deux arêtes de bord et un sommet intérieur ; c) deux faces et quatre arêtes de bord ; d) deux faces et aucune arête de bord ; e) deux faces et quatre sommets différents.

□

Afin de ne jamais se retrouver dans cette situation de blocage, nous supposerons que la triangulation T_{cible} ne contient pas plus de deux faces voisines partageant le même triplet de sommets. Cette remarque permet à n'importe quelle sous-triangulation de T_{cible} de ne pas être inconstructible à cause de cette situation de blocage. Cependant, nous insistons sur le fait qu'aucune contrainte n'est imposée à T_{init}, et si T_{cible} contient plus de deux faces adjacentes à un même triplet de sommets, il est possible d'utiliser une triangulation pivot ne contenant pas plus de deux faces adjacentes à un même triplet de sommets pour produire une séquence de bascules d'arêtes entre T_{init} et T_{cible}.

3.2.0.2 Hypothèses sur les arêtes contraintes

Si deux composantes correspondantes possèdent des arêtes contraintes, alors elles doivent être en quantité égale et elles doivent vérifier les critères suivants :

– La position relative des arêtes contraintes autour de chaque sommet dans le sens trigonométrique doit être identique dans les deux composantes connexes. Dans

le cas inverse, des faces de la composantes de T_{cible} ne peuvent pas être construites dans $T_{courant}$ (*cf* figure 3.8).

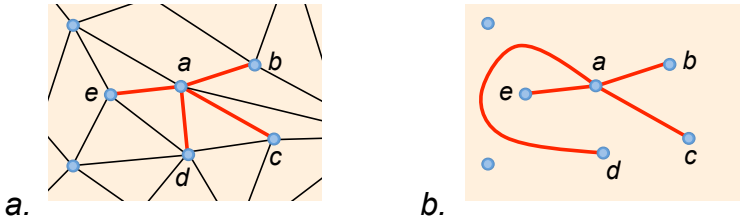


FIGURE 3.8 – La position relative des arêtes contraintes autour du sommet a est différente : en a) la face (abe) est constructible, mais pas en b).

– L'orientation des cycles d'arêtes contraintes doit être identique dans les deux composantes. Dans le cas contraire, des faces de la composantes de T_{cible} ne peuvent pas être construites dans la composante de $T_{courant}$ (*cf* figure 3.9). En particulier, les arêtes de bord d'une triangulation doivent satisfaire ce critère.

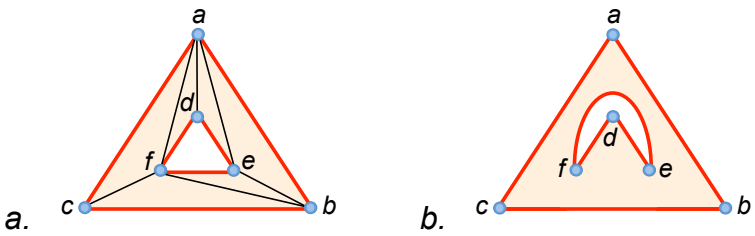


FIGURE 3.9 – L'orientation des cycles d'arêtes contraintes est différente. La totalité des arêtes présentes en a) ne peuve pas être construite en b).

Les bords étant considérés comme des arêtes contraintes, ils doivent satisfaire les conditions précédentes pour permettre la détermination d'une séquence de bascules d'arêtes. Dans ce cas, nous dirons que les bords sont *consistants*. Dans la suite, nous allons montrer le résultat suivant :

Théorème III.1 : si deux triangulations T_{init} et T_{cible} ont toutes leurs composantes correspondantes qui satisfont les hypothèses sur les arêtes contraintes, alors il existe une séquence de bascules d'arêtes permettant d'insérer la connectivité de T_{cible} dans T_{init}.

En particulier, si les deux triangulations possèdent une unique composante connexe, un même nombre de sommets, un même genre topologique et des bords consistants, alors on peut trouver une séquence de bascules d'arêtes entre elles. La mise en correspondance entre les sommets de composantes correspondantes peut être choisie de manière aléatoire, sauf pour les sommets adjacents aux arêtes contraintes (et donc en particulier aux arêtes de bord) qui doivent permettre

de satisfaire les hypothèses précédentes. Enfin, nous rappelons que la géométrie ne joue aucun rôle dans les résultats qui suivent. Cependant, pour faciliter la compréhension des schémas, nous avons choisi de représenter T_{init} et T_{cible} comme deux triangulations d'un même nuage de points, et nous mettons en correspondance les sommets qui possèdent la même géométrie. De plus, nous donnons un même nom aux sommets en correspondance dans les deux triangulations.

Nous allons montrer l'existence d'une bascule d'arête, en prouvant que l'on peut insérer la connectivité de T_{cible} dans $T_{courant}$. Dans un premier temps, nous présentons différentes approches pour construire une arête au sein d'une triangulation contenant des arêtes contraintes, puis nous montrerons montrerons une méthode générale pour construire l'ensemble des arêtes de la composante de T_{cible} dans $T_{courant}$. Nous en déduirons un algorithme pratique permettant de générer une séquence de bascules d'arêtes entre les deux triangulations.

3.3 Construire une arête au sein d'une triangulation

Nous présentons dans cette section une méthode pour construire une arête (ab) au sein d'une triangulation contenant des arêtes contraintes.

3.3.1 Détermination d'un chemin de faces exploitable entre deux sommets du maillage

Étant donnés a et b, deux sommets différents d'une composante connexe incluse dans $T_{courant}$ et contenant des arêtes contraintes. Nous présentons une méthode pour construire une arête (ab) dont on peut choisir la position relative par rapport à l'ordre des arêtes contraintes présentes autour de a et b. Le méthode consiste à déterminer un chemin de faces entre les sommets a et b tel que la construction de l'arête (ab) au sein du chemin respectera le positionnement souhaité autour des sommets a et de b (*cf* figure 3.10).

Soit $F = (f_1, ..., f_p)$ un chemin de faces entre les sommets a et b, et $E = (e_1, ..., e_{p-1})$ le *chemin d'arêtes* correspondant au chemin de faces F est composé d'arêtes non contraintes telles que $e_i = f_i \cap f_{i+1}$. On dira de F qu'il est *exploitable* s'il est simple, c'est-à-dire $f_i \neq f_j$ lorsque $i \neq j$, si le sommet a est incident à e_1 mais pas à e_2 et si le sommet b est incident à e_{p-1} mais pas à e_{p-2}. Nous faisons remarquer que pour un chemin de faces exploitable donné, le chemin d'arêtes correspondant n'est pas toujours unique car deux faces peuvent partager deux arêtes non contraintes différentes.

Supposons que l'on cherche à construire un chemin de faces exploitable entre les sommets a et b. On commence par choisir une face incidente à a et comprise entre les arêtes contraintes (ac_1) et (ac_2), et une face incidente à b et comprise entre les arêtes contraintes (bd_1) et (bd_2). On effectue ensuite un parcours en largeur de la triangulation sans franchir d'arêtes contraintes pour obtenir un

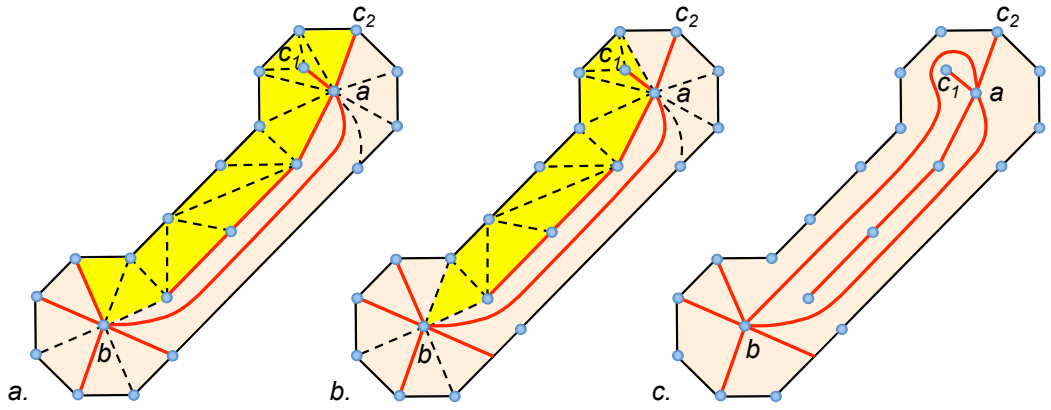


FIGURE 3.10 – À partir d'un chemin de faces simple entre a et b (en jaune) n'intersectant pas d'arêtes contraintes (en rouge), une séquence de bascules d'arêtes appartenant au chemin produira une arête (ab). Elle aura la même position que le chemin de faces initial en préservant l'ordre des arêtes contraintes autour de a (resp. b). Ici, le chemin et l'arête résultante (ab) commencent dans la partie de l'ombrelle comprise entres les arêtes contraintes (ac_1) et (ac_2) dans le sens trigonométrique. La même propriété est aussi vérifiée à l'autre extrémité du chemin.

chemin de faces simple $F_{init} = (f_1, ..f_q)$ et un chemin d'arêtes correspondant $E_{init} = (e_1, ..., e_{q-1})$. Puis on rend ces chemins exploitables en supprimant la première face de F_{init} et la première arête de E_{init} tant que cette dernière est incidente à a. On supprime également la dernière face de F_{init} et la dernière arête de E_{init} tant que celle-ci est incidente à b (*cf* figure 3.10)). Si F_{init} est vide, alors soit il existe déjà une arête (ab) dans T qui respecte l'ordre des arêtes contraintes autour de a et b, soit il n'existe pas de chemin de faces entre a et b. En ce qui concerne la complexité temporelle, la construction d'un chemin de faces exploitable entre a et b s'effectue en $O(e)$ si e est le nombre d'arête de $T_{courant}$.

Les conditions imposées sur les composantes correspondantes permettent de proposer le lemme suivant :
Lemme III.2 : Soient C_{cible} une composante connexe de T_{cible} et $C_{courant}$ sa composante correspondante dans $T_{courant}$. Si C_{cible} possède une arête (ab) non contrainte, alors soit il existe une arête (ab) dans $C_{courant}$ telle que la position relative par rapport aux arêtes contraintes autour de a et de b est identique, soit il existe un chemin de faces exploitable entre a et b tel que la position relative par rapport aux arêtes contraintes autour de a et de b est identique.

Démonstration. Supposons que $C_{courant}$ ne possède pas l'arête souhaitée, et montrons qu'il existe un chemin de faces entre a et b tel que la position relative par rapport aux arêtes contraintes autour de a et de b est identique à l'arête (ab) dans C_{cible}.

Par définition des composantes correspondantes, les sommets a et b sont adjacents à au moins une face de $C_{courant}$. De plus, nous savons que l'ordre des arêtes contraintes autour de a et b est identique dans T_{cible} et $T_{courant}$. Il existe donc au moins une face f_a dans $T_{courant}$ adjacente au sommet a et une face f_b adjacente au sommet b telles que leur position relative par rapport aux arêtes contraintes est identique avec l'arête (ab). Il nous reste à montrer que f_a et f_b appartiennent à $C_{courant}$:

Si le sommet a (resp. b) n'est pas adjacent à un cycle d'arêtes contraintes bordant $C_{courant}$, alors toutes les faces adjacentes à a (resp. b) appartiennent à $C_{courant}$. Donc f_a (resp. f_b) appartient à $C_{courant}$.

Si le sommet a (resp. b) est adjacent à un cycle d'arêtes contraintes, alors le respect de l'ordre des arêtes contraintes autour de a (resp. b) et de l'orientation des cycles bordant les composantes correspondantes, assurent que les positions relatives autour de a accessibles dans C_{cible}, sont aussi accessibles dans $C_{courant}$. Donc f_a (resp. f_b) appartient à $C_{courant}$. □

Dans la suite, nous représenterons les chemins de faces à partir d'un schéma dit "aplati", c'est-à-dire que nous dupliquons les sommets et nous le déformons pour obtenir une séquence de triangles incidents uniquement au niveau des arêtes (*cf* figure 3.11).

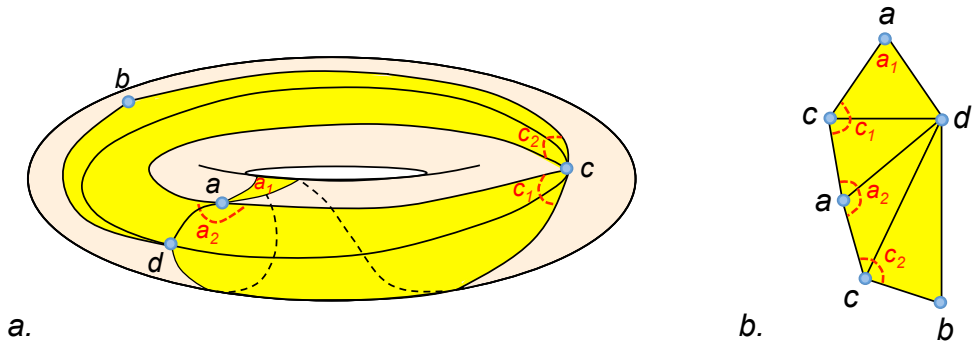


FIGURE 3.11 – a) un chemin de faces exploitable entre a et b sur une surface ; b) le chemin de faces "aplati" correspondant. En rouge, les portions d'ombrelles correspondantes dans les deux représentations.

3.3.2 Construction d'une arête à partir d'un chemin de faces exploitable

Nous allons à présent construire une arête (ab) au sein d'un chemin de faces exploitable en basculant les éléments du chemin d'arêtes associé E. Cette idée n'est pas nouvelle [Valette 2009], mais nous la généralisons pour des triangulations possédant des arêtes contraintes. Nous allons présenter trois méthodes pour réduire un chemin de faces en préservant sa position par rapport aux arêtes contraintes

présentes autour des sommets a et b. Lorsque le chemin contient uniquement deux faces, la bascule de l'unique arête de E est autorisée car les sommets a et b sont supposés différents, et produira bien l'arête (ab) souhaitée.

Dans la suite, nous désignons par *enveloppe d'un chemin de faces*, le cycle formé des arêtes incidentes à exactement une face du chemin.

3.3.2.1 Réduction d'un chemin de faces lorsque l'arête e_1 ou e_{p-1} de E est basculable

Si e_1 est basculable, alors après la bascule de e_1, le chemin résultant posséde une arête incidente au sommet a, et deux faces f_1' et f_2' qui sont nécessairement localisées dans l'enveloppe du chemin de faces original. Il n'est pas restrictif de considérer que f_2' est la face créée incidente à e_2. On peut extraire un nouveau chemin de faces $F' = (f_2', ..., f_i, ...f_p)$ qui relie les sommets a et b, est inclus dans F, et possède une face de moins que F (*cf* figure 3.12). De même, si e_{p-1} est basculable, nous pouvons construire un chemin de taille inférieure à celui de F.

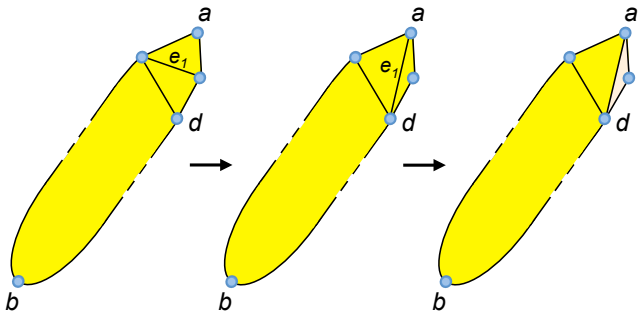


FIGURE 3.12 – Représentation abstraite du cas où e_1 est basculable : en jaune le chemin de faces et en beige la face retirée à F.

En particulier, si le sommet a n'est incident à aucune autre face dans F que la première, alors il suffit de basculer dans l'ordre les arêtes de E pour construire l'arête (ab) (*cf* figures 3.13). En effet, e_1 est basculable, sinon f_1 et f_2 seraient incidentes au même triplet de sommets contenant a, contrairement à l'hypothèse. La bascule de e_1 génère une arête incidente au sommet a, et nous pouvons extraire un nouveau chemin de faces $F' = (f_2', ..., f_i, ...f_p)$ entre les sommets a et b. Ce chemin est inclus dans F, possède une face de moins que F et le sommet a est incident à aucune face de F' sauf f_2'. Par un raisonnement équivalent, on en déduit que l'arête (ab) peut être construite en basculant itérativement toutes les arêtes $(e_1, ..., e_{p-1})$, et sa position sera incluse dans l'enveloppe du chemin de faces simplifé original F.
La complexité temporelle correspondant au retrait d'une face, s'effectue en $O(1)$ donc la construction d'une arête (ab) telle que a est incident à aucune face de F sauf la première, s'effectue en $O(e)$ avec e le nombre d'arêtes de E.

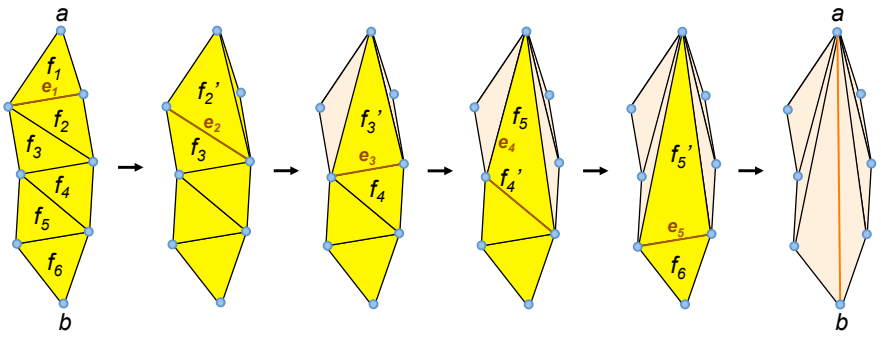


FIGURE 3.13 – Construction de l'arête (ab) à partir d'un chemin de faces dans lequel a n'est incident à aucune face de F à l'exception de f_1.

De même, si b n'est incident à aucune autre face dans F que (f_p), alors l'arête (ab) peut être construite simplement en basculant les arêtes de E dans l'ordre inverse, et la complexité est également $O(e)$.

3.3.2.2 Réduction d'un chemin de faces contenant au moins une arête basculable

Soient $F = (f_1, f_2, ..., f_p)$ le chemin de faces initial et $E = (e_1, ..., e_{p-1})$ le chemin d'arêtes correspondant tel que les arêtes e_1 et e_{p-1} ne sont pas basculables. Supposons que e_i est la première arête basculable, ce qui signifie que les i premières faces de F, $f_1, f_2, .., f_i$, partagent le même triplet de sommets que f_1, et que le sommet s_F opposé à e_i dans f_{i+1} n'est pas un sommet adjacent à $f_1, f_2, .., f_i$. Dans le chemin F' reliant les sommets a et s_F, s_F n'est incident à aucune autre face que la dernière. On peut donc construire l'arête (as_F) directement dans F' par la méthode présentée précédemment 3.3.2.1 (en basculant itérativement les arêtes $e_i, e_{i-1}, ..., e_1$). L'arête résultante (as_F) divise la région contenant le chemin initial F en deux parties non vides. L'une d'elles est un chemin de faces simple entre les sommets a et b (*cf* figure 3.14), et ce nouveau chemin respecte la position du chemin initial F et sa longueur est strictement inférieure à celle de F.

Dans la suite, l'opération consistant à appliquer l'une des deux méthodes de réduction de chemin est appelée une *réduction locale*.

3.3.2.3 Réduction d'un chemin de face ne contenant aucune arête basculable

Supposons qu'aucune arête de E n'est basculable (ce qui signifie que toutes les faces de F partagent un même triplet de sommets a, b et c). Dans ce cas, nous ne pouvons pas effectuer de réduction locale, mais le chemin F peut servir de point de départ à la construction d'un nouveau chemin simple et réductible. Après réduction locale, ce chemin aura une longueur inférieure ou égale à F, et il respectera les

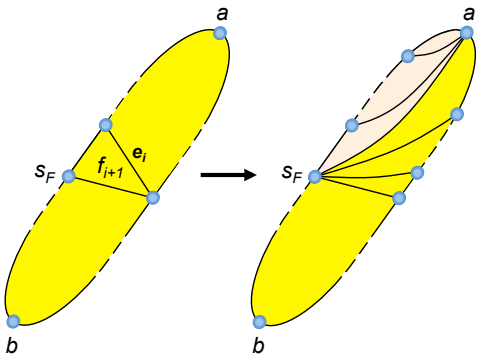


FIGURE 3.14 – Représentation abstraite du cas où a est incident à plusieurs faces de F et e_i est la première arête basculable dans le chemin d'arêtes entre a et b : en jaune, le chemin de faces et en beige, les faces retirées à F. Pour clarifier l'illustration, le chemin de faces est représenté en une bande, alors que le sommet a est incident à plusieurs faces.

mêmes positions relatives par rapport aux arêtes contraintes autour des sommets a et b.

En tournant dans le sens trigonométrique sur l'enveloppe de F, nous introduisons la *demi-enveloppe de droite*, composée de la liste de sommets rencontrés entre a et b, et la *demi-enveloppe de gauche*, composée de la liste de sommets rencontrés entre b et a.

Lemme III.3 : Soit F un chemin de faces exploitable entre a et b contenant exclusivement des faces incidentes aux trois mêmes sommets a, b et c. Si une demi-enveloppe de F contient au moins une occurrence de c et s'il est possible d'insérer un sommet d (différent de a, b et c) au sein de l'autre demi-enveloppe par des bascules d'arêtes, alors on peut construire un nouveau chemin F' de longueur inférieure au chemin F. L'enveloppe de F' ne sera pas incluse dans F, mais préservera les mêmes positions relatives par rapport aux arêtes contraintes autour des sommets a et b.

Démonstration. Si une demi-enveloppe notée DE_g, contient au moins une occurrence de c, on insère le sommet d dans la seconde demi-enveloppe DE_d. On construit un chemin de faces exploitable minimal entre d et une arête arbitraire e de DE_d, et on le complète avec la face de F incidente à e (*cf* figure 3.15). On obtient un chemin $F_d = (f'_1, ..f'_{p'})$ entre un sommet s_F appartenant à DE_g et d. Ensuite, on construit l'arête $(s_F d)$ au sein de F_d puisque d n'est incident à aucune face de F_d sauf $(f'_{p'})$. On insère ensuite la nouvelle arête (ds_F) dans le chemin d'arête E, et l'on obtient un nouveau chemin de faces de taille égale à celle de F plus un.

Remarquons que d ne peut pas être connecté à la fois au sommet de départ et d'arrivée de F, car cela signifierait que l'arête (ab) souhaitée était déjà présente avant la recherche d'un chemin entre d et s_F. Il n'est donc pas restrictif de considérer qu'il n'existait pas d'arête (ad) respectant la même position relative autour de a

que F. Cette arête est directement constructible car dans le sous-chemin de faces reliant l'origine a au sommet d, le sommet d n'est incident à aucune face sauf la dernière. Cette nouvelle arête sépare le chemin en deux parties non vides, dont l'une est un chemin de faces entre les sommets a et b respectant bien les mêmes positions relatives que F, et contenant au plus le même nombre de faces que F. La demi-enveloppe DE_g est inchangée, donc le chemin résultant est incident aux sommets a, b, c et d, ce qui signifie qu'il contient au moins une arête basculable, et nous pouvons le réduire par une réduction locale.

Dans le cas particulier où DE_g est composée uniquement des deux arêtes (bc) et (ca), le sommet s_F correspond au sommet c. □

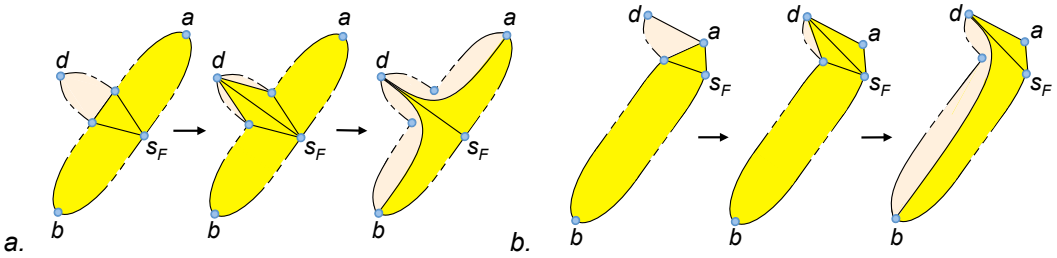


FIGURE 3.15 – Représentation abstraite du cas où le chemin F est composé de faces qui sont toutes incidentes au même triplet de sommets a, b et c. Si l'une des demi-enveloppes de F contient au moins une occurrence de c et l'autre demi-enveloppe peut être connectée à un quatrième sommet d, alors F peut être transformé en un nouveau chemin de faces par réduction locale : on détermine un chemin de faces entre d et une arête de la demi-enveloppe de F (en jaune) auquel on ajoute la face de F incidente à l'arête. À partir du chemin de faces, on construit l'arête (ds_F), on insère d dans une demi-enveloppe du nouveau chemin F et on laisse l'autre demi-enveloppe inchangée (le nouveau chemin contient une face de plus que F). La construction de (ad) et (db) (si elles ne sont pas déjà construites) réduit la longueur du chemin entre les sommets a et b.

Nous appellons cette opération un *déplacement local* du chemin, et nous pouvons construire une arête (ab) au sein d'un chemin de faces exploitables dans les cas suivants :

Lemme III.4 : Soit F un chemin de faces exploitable entre a et b. Si une demi-enveloppe de F contient au moins une occurrence de c et si l'autre demi-enveloppe contient un sommet d différent de a, b et c ou s'il existe un chemin de faces entre le sommet d et une arête non contrainte de la demi-enveloppe, alors l'arête (ab) est constructible et possèdera les mêmes positions relatives que F par rapport aux arêtes contraintes autour de a et b.

Démonstration. Dans les conditions du lemme, soit le chemin F est incident à au

moins quatre sommets différents et il est réductible par réduction locale, soit il est incident à seulement trois sommets différents et il est réductible par déplacement local. Il nous reste à démontrer qu'après une réduction locale et un déplacement local, les conditions du lemme sont encore vérifiées.

Si un chemin F vérifie les conditions du lemme, alors ce sera encore vrai après la bascule d'une de ses arêtes internes car les demi-enveloppes n'ont pas été modifiées. De plus, soient une face f retirée du chemin F, et une arête e adjacente à f n'appartenant plus à une demi-enveloppe du chemin. Pour tout sommet v tel qu'il existe un chemin de faces F' entre v et e, nous pouvons construire un chemin de faces entre v et une nouvelle arête de la demi-enveloppe en ajoutant la face f au chemin F'. Il existe donc un chemin de faces entre une nouvelle arête d'une demi-enveloppe et tous les sommets incidents et accessibles par les arêtes n'appartenant plus à cette demi-enveloppe. On en déduit que les conditions du lemme sont conservées après réduction locale. Après un déplacement local, l'une des demi-enveloppes est incidente à un sommet c différent de a et b, et la seconde est incidente à un sommet d différent de c, a et b. On se retrouve donc bien dans les conditions du lemme. □

En particulier, si la composante connexe contient au moins quatre sommets différents et si le chemin de faces exploitable F possède une demi-enveloppe incidente uniquement à deux arêtes contraintes, alors on peut construire l'arête désirée (*cf* figure 3.16.a). De même, si F ne divise pas la composante connexe en deux parties disjointes, alors on peut construire l'arête désirée (*cf* figure 3.16.b).

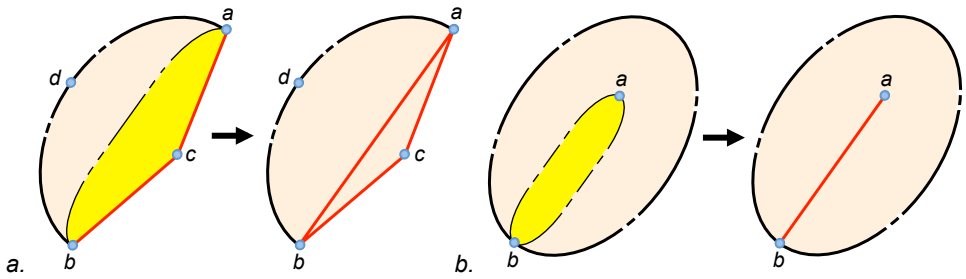


FIGURE 3.16 – Les configurations a) et b) sont des cas particuliers du lemme $III.4$, et l'on peut donc y construire les arêtes (ab) souhaitées (les arêtes contraintes sont en rouge).

En ce qui concerne la complexité temporelle, s'il existe une arête basculable dans le chemin F, le retrait d'une face nécessite la construction d'une arête (as_F) dans un sous-chemin de F, avec s_F incident à aucune face sauf la première. Le retrait s'effectue donc en $O(e)$, avec e le nombre d'arête de $T_{courant}$. Si aucune arête n'est basculable, l'opération nécessite la construction successive de trois arêtes dont l'une de leurs extrémités est incidente à aucune faces sauf la première dans les chemins considérés. La complexité temporelle est donc en $O(e)$. Au final, la construction d'une arête s'effectue en $O(e)$ lorsque l'une des extrémité de l'arête souhaitée est incidente à aucune faces du chemin sauf la première, et en $O(e^2)$ sinon.

3.3.3 Configurations ne permettant pas la construction d'une arête souhaitée

Les opérations présentées ne permettent pas toujours de construire une arête (ab) à partir d'un chemin de faces entre les sommets a et b :

Lemme III.5 : Dans certains cas, les chemins de faces exploitables présents au sein d'une composante connexe ne permettent pas de construire une arête souhaitée.

Démonstration. Nous avons identifié un *cas pathologique* où l'arête (ab) ne peut pas être construite en utilisant un chemin de faces exploitable (*cf* figure 3.17).

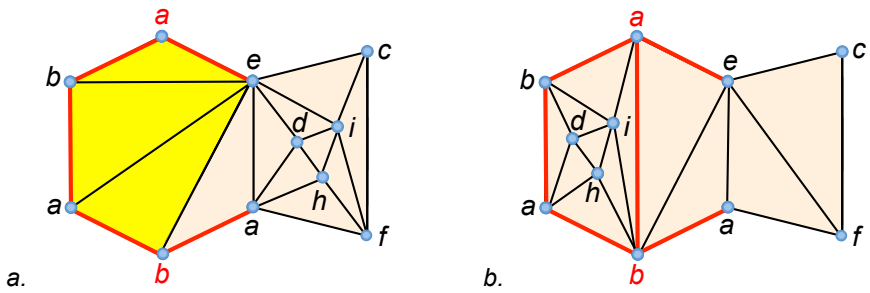


FIGURE 3.17 – a) Triangulation T dans laquelle on souhaite construire une nouvelle arête (ab) entre les arêtes contraintes (colorées en rouge). a) en jaune, l'unique chemin de faces exploitable disponible pour construire l'arête (ab) ; b) Il existe bien une triangulation de T contenant l'arête (ab) désirée, mais elle n'est pas constructible à partir des méthodes proposées.

□

3.4 Critères pour insérer la connectivité d'un maillage

Nous savons dans quels cas nous pouvons construire une arête et comment procéder. Nous montrons à présent une méthode générale pour insérer une connectivité complète dans une triangulation.

3.4.1 Stratégie pour insérer itérativement les arêtes

Nous avons montré que dans une triangulation $T_{courant}$ de $\mathbb{T}$, plusieurs arêtes pouvaient être construites à partir d'un chemin de faces. Cependant, les arêtes de T_{cible} ne peuvent pas être insérées dans un ordre arbitraire, car la construction d'une arête peut interagir avec une arête précédemment construite. Un algorithme naïf consistant à construire itérativement les arêtes de T_{cible} dans $T_{courant}$ peut ne pas converger car l'introduction d'une arête peut nécessiter une bascule — et donc la destruction — d'une arête précédemment construite. Afin d'éviter les risques de boucles infinies, nous choisissons de *bloquer* les arêtes construites afin de les préserver d'une destruction ultérieure potentielle. Lorsqu'une arête de T_{cible} est

construite dans $T_{courant}$, elle est bloquée, ce qui signifie qu'elle devient contrainte et n'est plus utilisée dans la construction de futurs chemins de faces. Cependant, il est nécessaire d'imposer des contraintes sur la stratégie d'insertion des arêtes au sein de $T_{courant}$ pour permettre la construction de toutes les arêtes de T_{cible}. Notre objectif est qu'après l'insertion et le blocage d'une arête, l'ensemble des régions correspondantes dans les deux triangulations satisfont les hypothèses sur les arêtes contraintes introduites dans la partie 3.2. Ainsi, d'après le lemme III.2, pour un arête (ab) dans T_{cible} non présente dans $T_{courant}$, on pourra toujours déterminer un chemin de faces exploitable dont les positions relatives par rapport aux arêtes contraintes autour des sommets a et b sont similaires. Nous devons donc vérifier les conditions suivantes :

- Le blocage d'une arête divisant une composante connexe en deux est interdit lorsque deux sommets adjacents dans T_{cible} se retrouvent dans des composantes différentes (*cf* figure 3.18).

- Le blocage d'une arête (ab) doit respecter les mêmes positions relatives par rapport aux arêtes contraintes autour des sommets a et b dans $T_{courant}$ et T_{cible} (ce qui est assuré si le chemin de faces utilisé respecte cet ordre).

- Lorsque la construction et le blocage d'une arête créent un cycle d'arêtes contraintes, il faut vérifier que l'orientation de ce cycle dans $T_{courant}$ est consistante avec son orientation dans T_{cible} (*cf* figure 3.19).

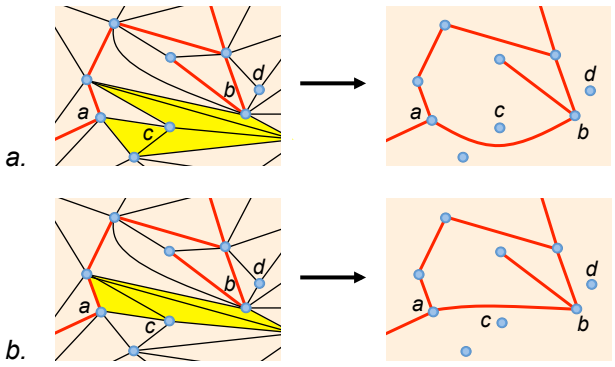


FIGURE 3.18 – En a) et b), il existe deux constructions possibles de l'arête (ab) respectant l'ordre des arêtes contraintes autour de a et b et fermant un cycle d'arêtes contraintes de $T_{courant}$. En a) le cycle ne permet pas de construire l'arête (cd), à la différence du cycle en b).

Dans le cas où l'une des contraintes n'est pas satisfaite, la construction proposée de (ab) est annulée. Nous démontrons maintenant le théorème III.1, en présentant un algorithme simple et prouvé d'insertion de connectivité en utilisant des bascules d'arêtes.

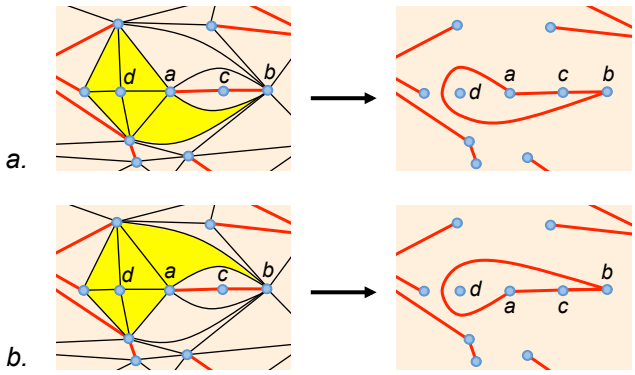


FIGURE 3.19 – En a) et b), il existe deux constructions possibles de l'arête (ab) générant deux cycles d'arêtes contraintes de $T_{courant}$ aux orientations opposées. La construction de l'arête (ad) en a) et en b) génère deux arêtes (ab) ne possédant pas le même ordre d'arêtes contraintes autour de a.

3.4.2 Algorithme générique d'insertion de connectivité

Soient deux triangulations T_{init} et T_{cible} appartenant à $\mathbb{T}$, dont toutes les composantes correspondantes ont un même nombre de sommets, un même genre topologique et satisfont les hypothèses sur les arêtes contraintes introduites dans la partie 3.2. Nous initialisons $T_{courant}$ avec T_{init}, et présentons un algorithme qui construit itérativement chaque arête de T_{cible} au sein de $T_{courant}$. Dans la suite, nous dirons qu'une arête (ab) de T_{cible} est *directement constructible* si :

- l'arête est constructible au sein du chemin par les techniques présentées dans la partie 3.3.2,
- l'arête vérifie les trois contraintes introduites dans la partie 3.4.1.

L'algorithme proposé est fondé sur la propriété qu'à chaque étape, il existe au moins une arête *directement constructible* :

Algorithme 7 Insertion de la connectivité de T_{cible} dans $T_{courant}$

tant que T_{cible} contient des arêtes non contraintes n'appartenant pas à $T_{courant}$ **faire**
  Trouver une arête e de T_{cible} directement constructible et n'appartenant pas à $T_{courant}$
  Construire e dans $T_{courant}$
  Bloquer e
fin tant que

3.4.3 Preuve de l'algorithme générique

Pour démontrer que l'algorithme fait bien converger la connectivité de T_{cible} au sein de $T_{courant}$, il suffit de prouver qu'à chaque étape il existe au moins une arête directement constructible :

• S'il existe un chemin de faces formant un cycle dans T_{cible} (c'est-à-dire que $f_0 = f_p$) ne traversant pas d'arêtes contraintes, alors toutes les arêtes (ab) de T_{cible} incidentes à deux faces du chemin sont directement constructibles (*cf* figure 3.20). La construction de l'arête (ab) est possible car le chemin de faces exploitable ne divise pas la composante connexe en deux parties disjointe et satisfait bien les conditions du lemme III.4. De plus, le blocage de (ab) ne ferme pas de cycle formé d'arêtes contraintes et n'implique pas la division d'une composante en deux parties. Les trois conditions de la partie 3.4.1 sont donc respectées.

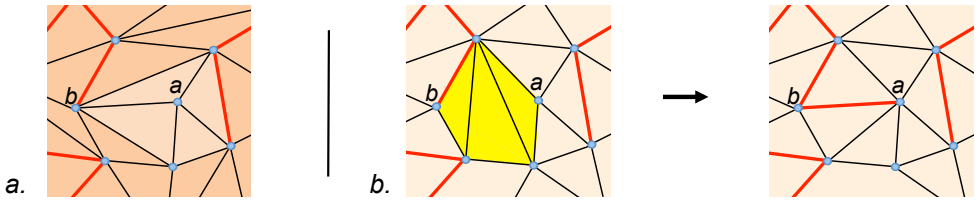


FIGURE 3.20 – a) Un cycle de faces (en beige) autour d'un sommet a dans T_{cible} (les arêtes contraintes sont en rouge) ; b) La construction de (ab) dans $T_{courant}$ est toujours possible (le chemin de faces utilisé est en jaune).

• Montrons maintenant qu'il existe toujours au moins une arête directement constructible lorsqu'il n'existe aucun cycle de faces dans T_{cible} (la construction d'une arête de T_{cible} dans $T_{courant}$ divisera donc une composante connexe en deux). Dans ce cas, chaque composante connexe est formée d'un arbre dont les noeuds sont des faces et les branches sont des arêtes non contraintes adjacentes aux faces. Si tous les arbres sont constitués d'un unique noeud, alors chaque composante connexe est constituée d'une face et l'algorithme est terminé. Sinon, on considère une feuille d'un arbre qui n'est pas réduit à un unique noeud. Cette feuille correspond à une face orientée (abc) de T_{cible} incidente à deux arêtes contraintes (ca) et (cb).

L'arête (ab) qui ferme la face (abc) est directement constructible dans $T_{courant}$ (*cf* figure 3.21) : en effet, un parcours en largeur compatible avec les arêtes contraintes autour des sommets a et b donne un chemin composé des faces incidentes au sommet c entre (ca) et (cb) dans $T_{courant}$. D'après le lemme III.4, l'arête (ab) est constructible, et respectera les mêmes positions relatives par rapport aux arêtes contraintes autour des sommets a et b. De plus, puisqu'il n'existe pas de cycles de faces dans $T_{courant}$, il y a unicité de l'orientation de la face (abc) réalisable. Pour satisfaire les contraintes de la partie 3.4.1, il reste à montrer que le triangle orienté (abc) obtenu ne contient aucun sommet. Or, après chaque bascule

d'arête ou déplacement local, l'une des deux demi-enveloppes du chemin de faces est toujours composée uniquement des arêtes (ca) et (cb), ce qui implique que l'arête (ab) est voisine des arêtes (ca) et (cb) et donc que le triangle orienté (abc) est vide.

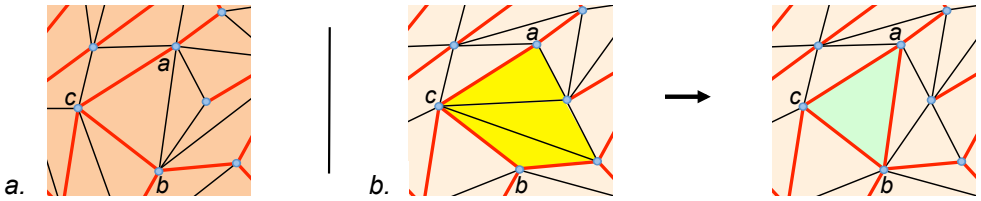


FIGURE 3.21 – a) L'arête (ab) borde la face orientée (abc) avec les arêtes contraintes (ca) et (cb) dans T_{cible} (les arêtes contraintes sont en rouge) ; b) La construction de (ab) dans $T_{courant}$ (le chemin de faces utilisé est en jaune).

En démontrant que l'algorithme générique fait bien converger la connectivité de T_{cible} au sein de $T_{courant}$, nous prouvons aussi le théorème III.1. En effet, il est possible de déterminer une séquence de bascules d'arêtes entre les triangulations T_{init} et T_{cible}.

La complexité de l'algorithme générique, tel qu'il est décrit dans l'algorithme 7, dépend dans une large mesure de celle de l'opération déterminant si une arête est directement constructible ou non. En particulier, si la construction d'une arête divise une composante connexe dans $T_{courant}$, la vérification des nouvelles composantes connexes peut avoir un coût important ($O(e^3)$ si e est le nombre d'arête de T_{cible}). Néanmoins, on peut déduire de la preuve de l'algorithme générique, une version beaucoup plus efficace car les tests déterminant si une arête est directement constructible ou non, se limite à vérifier s'il existe un chemin de faces formant un cycle dans T_{cible} ou si sa construction ferme une face dans T_{cible}. Dans la section suivante, nous proposons une stratégie par croissance qui se révèle encore plus efficace car nous n'effectuerons pas de recherche d'une arête directement constructible. À chaque instant de l'algorithme, nous aurons toujours connaissance d'au moins une arête directement constructible à partir des arêtes précédemment construites.

3.5 Algorithme pratique d'insertion de connectivité

Dans cette partie, nous travaillons avec deux triangulations T_{init} et T_{cible} appartenant à $\mathbb{T}$, qui possèdent une unique composante connexe, le même nombre de sommets, le même genre topologique et des bords consistants. Nous initialisons $T_{courant}$ par T_{init} et présentons un cas particulier de l'algorithme décrit précédemment, dans lequel nous construisons itérativement des faces orientées en procédant par croissance de régions. Cette approche assure qu'à chaque étape, toutes les arêtes

non bloquées appartiennent à une même composante connexe.

Durant le déroulement de l'algorithme, la triangulation $T_{courrant}$ est divisée en deux régions : la région *conforme*, R_c, contenant les faces de T_{cible} déjà construites dans $T_{courant}$ (et bloquées), et la région *non explorée*, R_n, contenant les autres faces.

3.5.1 Stratégie par croissance de régions

À partir d'une face construite dans $T_{courant}$, nous souhaitons que la stratégie par croissance de régions garantisse que la région non explorée R_n soit composée d'une unique composante connexe. Cela permettra de n'avoir aucune vérification à effectuer sur la validité des arêtes construites. Nous autorisons donc la construction d'une face (abc) uniquement dans l'un des trois cas suivants.

3.5.1.1 Fermeture de face (cas FF)

Si deux arêtes (ac) et (ab) sont contraintes, (abc) est construite en deux étapes (*cf* figure 3.22). On détermine dans un premier temps dans $T_{courant}$ l'unique chemin de faces $F = (f_1, f_2, ..., f_i, ..., f_p)$ qui relie b et c dans l'ombrelle composée des faces incidentes à a, puis on utilise ce chemin pour construire l'arête (bc) par réduction locale et déplacement local (la construction étant assurée par le lemme III.4). Après chaque bascule d'arête ou déplacement local, l'une des deux demi-enveloppes du chemin de faces est toujours composée uniquement des arêtes (ca) et (ab). L'arête (bc) obtenue est donc voisine des arêtes (ca) et (ab) et le triangle orienté (abc) est vide.

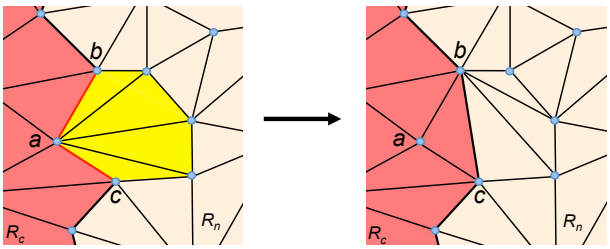


FIGURE 3.22 – Construction de la face (abc) dans le cas (FF).

3.5.1.2 Création d'une face vers un nouveau sommet (cas NS)

Si une arête (ab) est contrainte et le sommet c n'est incident à aucune arête contrainte, alors la face (abc) peut être construite (*cf* figure 3.23). Dans un premier temps, l'arête (ac) est construite : on détermine un chemin de faces minimal exploitable (c'est-à-dire que le sommet c n'est incident à aucune face du chemin sauf la dernière) joignant le sommet a avec le sommet c, puis l'arête (ac) est construite en réduisant le chemin de faces par réduction locale. Dans un second temps, l'arête (bc) est construite comme dans le cas (FF) précédent.

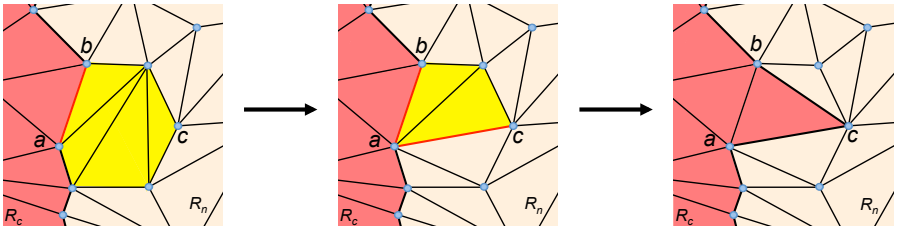


FIGURE 3.23 – Construction de la face (abc) dans le cas (NS).

3.5.1.3 Création d'une face vers un sommet déjà découvert (cas SD)

Si une arête (ab) est contrainte et le sommet c est déjà incident à des arêtes contraintes, la construction de la face orientée (abc) est autorisée uniquement dans le cas où la face résultante ne divise pas R_n en deux composantes connexes (*cf* figure 3.24). Dans un premier temps, l'arête (ac) est construite : on détermine un chemin de faces minimal exploitable (c'est-à-dire que le sommet c n'est incident à aucune face du chemin hormis la dernière) joignant le sommet a avec le sommet c, puis l'arête (ac) est construite par réduction locale et déplacement local du chemin (construction assurée par le lemme III.4). Enfin, l'arête (bc) est construite comme dans le cas (FF).

En pratique, le cas (SD) est utilisé lorsque les cas (NS) et (FF) ne peuvent plus être appliqués dans R_n (et R_n contient plus d'une face). R_n correspond donc à un cycle de faces ou à l'intersection de plusieurs cycles, et la construction d'une des faces (abc) à partir du cas (SD) ne divisera pas R_n en deux composantes connexes.

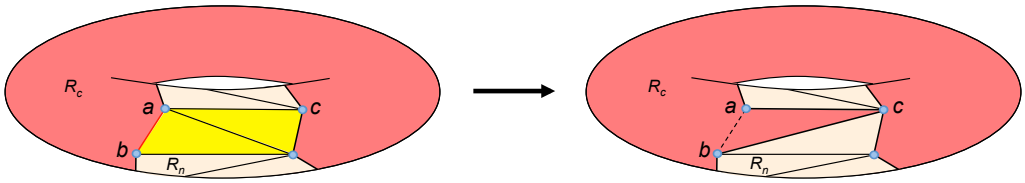


FIGURE 3.24 – Construction de la face (abc) dans le cas (SD). La région R_n possède encore une seule composante connexe.

La complexité temporelle correspondant à la construction d'une face, dépend de la complexité associée à la construction d'une arête au sein d'un chemin. Dans chacun des cas, la construction est en $O(e^2)$ lorsqu'il faut construire une arête dont ses extrémités sont incidentes à plus d'une face du chemin associé, et $O(e)$ sinon.

On remarque qu'aucun choix d'orientation n'est nécessaire lors de la construction d'une face dans les trois cas décrits. En effet, l'orientation de chaque face dépend de l'orientation de sa voisine déjà construite. Cela signifie que l'orientation des faces d'une région qui croît dépend uniquement de l'orientation de la face qui a initialisé la région.

3.5.2 Construire une face avec la bonne orientation

Nous souhaitons que les faces orientées de $T_{courant}$ soient identiques aux faces orientées de T_{cible}. Il suffit pour cela de garantir la cohérence de l'orientation de la première face (abc) dans $T_{courant}$ par rapport à son orientation dans T_{cible}.

Pour construire la première face (abc), on commence par rechercher un chemin de faces minimal entre les sommets a et b (le sommet a n'est incident à aucune face du chemin hormis la première), et on construit l'arête (ab) par réduction locale du chemin. Puis on fait de même pour l'arête (ac). Enfin, on construit l'arête (bc) comme décrit dans le cas (FF). Cependant, si le sommet a n'est pas adjacent à une arête contrainte différente de (ab) et (ac), il existe deux chemins de faces possibles dans l'ombrelle de a, correspondant à deux orientations différentes de la face (abc) obtenue (*cf* figure 3.25). Parmi ces deux configurations, l'algorithme choisira celui qui possède la bonne face orientée.

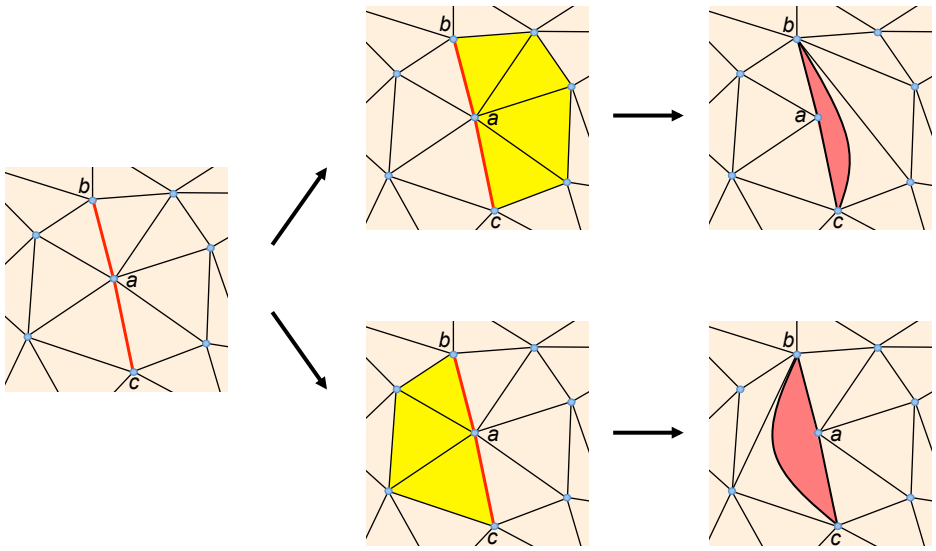


FIGURE 3.25 – Construction de la première face (abc) avec les deux orientations possibles.

3.5.3 Algorithme pratique

Nous pouvons maintenant écrire l'algorithme construisant la connectivité de T_{cible} dans $T_{courant}$:

Algorithme 8 Algorithme pratique insérant la connectivité de T_{cible} dans $T_{courant}$

```
Construire la première face (avec la bonne orientation)
tant que Il reste au moins deux faces à construire faire
  tant que Il existe une face correspondant aux cas (NS) ou (FF) faire
    Construire la face
    Bloquer ses arêtes
  fin tant que
  si Il reste au moins deux faces à construire alors
    Construire une face vérifiant (SD)
    Bloquer ses arêtes
  fin si
fin tant que
```

À la fin de l'algorithme, il reste une face incidente à trois arêtes contraintes. Puisque dans $T_{courant}$ et T_{cible}, cette dernière face est incidente aux mêmes sommets et possède la même orientation (par la consistance d'orientation discutée ci-dessus), elle est identique dans les deux triangulations. Lorsque toutes les faces de T_{cible} sont construites, l'algorithme est terminé et les connectivités de $T_{courante}$ et T_{cible} sont identiques.

Nous avons signalé que la construction d'une face a une complexité temporelle $O(e^2)$ dans le pire cas (avec e le nombre d'arêtes de T_{cible}). On en déduit que l'aglorithme pratique possède une complexité $O(e^3)$ dans le pire des scénarios. Si T_{init} et T_{cible} sont identiques, il n'y a aucune arête à construire et la complexité est en $O(e)$.

Dans cet algorithme, on fait croître une seule région initialisée par une face ayant la bonne orientation. Cependant, on peut aussi utiliser le même algorithme en initialisant k faces disjointes générant k régions croissantes. En effet, la région R_n doit être impérativement limitée à une seule composante connexe, mais la méthode n'impose aucune contrainte sur R_c.

3.5.4 Preuve de l'algorithme

L'algorithme par croissance de régions peut être prouvé comme un cas particulier de l'algorithme générique, mais nous préférons présenter ici une démonstration alternative.

Avant de démontrer que la connectivité de $T_{courant}$ converge bien vers la connectivité de T_{cible}, nous rappelons qu'à chaque étape de l'algorithme, la triangulation $T_{courant}$ est toujours composée de deux régions : R_c (la région conforme) et R_n (la région non explorée). Après la construction d'une face, R_n est toujours contenue dans une unique composante connexe. Montrons que la croissance de régions utilisant les cas (NS), (FF) et (SD) peut construire et bloquer toutes les

faces de $T_{courant}$. Nous rappelons que si R_c n'est pas vide alors R_n est une surface contenant un bord.

Lemme III.6 : Si une expansion (c'est-à-dire une croissance de régions utilisant uniquement les cas (NS) et (FF)) ne réduit pas R_n à une unique face orientée, alors R_n n'est pas contractile.

Démonstration. Supposons que R_n ne soit pas réduite à une face après expansion et qu'elle soit contractile. Soit B l'ensemble des faces non explorées de T_{cible} incidentes à au moins une arête du bord de R_n. Les faces de B ont leurs trois sommets appartenant à sa frontière, car sinon elles pourraient être construites par expansion à partir de (NS) et insérées dans R_c. Comme la région non explorée R_n est contractile, l'ensemble des triangles de B est donc inclus dans un pavage triangulaire d'un n-gone dont les sommets sont incidents au bord de R_c. Nous faisons remarquer que deux triangles de B peuvent ête voisins dans le pavage (car elles sont incidentes à un même couple de sommet) sans être nécessairement deux faces adjacentes dans la triangulation T_{cible} (*cf* figure 3.26). Cependant, comme toutes les triangulations d'un n-gone ont au moins une face incidente à deux arêtes du bord, l'ensemble B possède au moins un triangle t incident à deux arêtes du bord. La face de B correspondante à t est alors constructible par le cas (FF) et peut être insérée dans R_c, ce qui est contraire à l'hypothèse. □

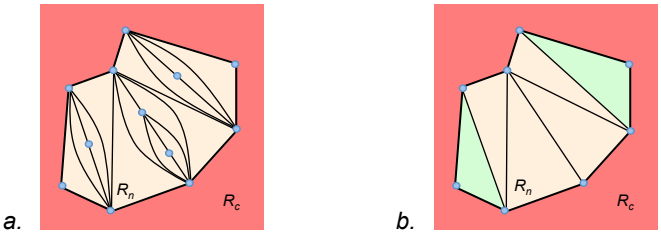


FIGURE 3.26 – Pavage d'un n-gone correspondant à la frontière entre la région non explorée R_n et la région conforme R_c : a) La région non explorée est contractile ; b) Le pavage triangulaire contient les faces incidentes aux arêtes du bord.

Theorème III.2 : À la fin de l'algorithme, la connectivité de $T_{courant}$ et T_{cible} sont identiques.

Démonstration. Montrons que tant qu'il reste des faces orientées de T_{cible} n'appartenant pas à $T_{courant}$, il est toujours possible de construire une nouvelle face orientée en utilisant l'un des trois cas de l'algorithme. Supposons que l'expansion (cas (NS) et (FF) uniquement) ne réduise pas R_n à une unique face orientée. D'après le lemme III.6, R_n est une surface qui n'est pas contractile, et il existe des cycles qui ne sont pas homotopes à un point. Les arêtes du bord entre R_c et R_n forment un chemin fermé, et l'on construit un chemin de triangles avec les faces de R_n incidentes à ces arêtes. Le chemin de faces est fermé (mais pas nécessairement simple), et il est

possible d'en extraire un ou plusieurs cycles de faces. Par conséquent, il existe au moins un cycle de faces simple, contenant au moins une face avec une arête sur le bord de R_c et R_n (et ses trois sommets sont incidents à une arête contrainte, sinon le cas (NS) aurait pu être utilisé). Cette face peut être insérée dans $T_{courant}$ par le cas (SD), car elle ne divisera pas la composante connexe de R_n en deux parties.

Par ces observations, nous déduisons qu'il est toujours possible de construire une nouvelle face de T_{cible} dans $T_{courant}$, car après chaque expansion, soit la détermination est complète, soit il est possible de construire une face en utilisant le cas (SD). □

En fait, les étapes d'expansion (cas (NS) et (FF) uniquement) peuvent être vues comme un moyen de faire croître R_c sans changer la caractéristique d'Euler. En revanche, cette caractéristique diminue de un à chaque fois que l'on construit une face en utilisant (SD). On en déduit le corollaire suivant :

Corollaire III.1 : Soient g le genre de la surface orientée et b son nombre de bords fermés disjoints. L'algorithme utilisera $2g + b$ fois le cas (SD).

Démonstration. Après la construction de la première face, la caractéristique d'Euler de R_c est égale à 1. L'expansion ne change pas la caractéristique d'Euler de R_c, alors que le cas (SD) la décrémente de 1. Comme la caractéristique d'Euler de T_{cible} privée d'une face est égale à $2 - 2g - (b + 1)$, le cas (SD) sera appliqué $2g + b$ fois. □

Notons que si l'on souhaite utiliser l'algorithme avec f faces de départs disjointes et non incidentes à un bord, l'algorithme utilisera $2g + b + f$ fois le cas (SD). En effet, il faudra utiliser f fois le cas (SD) afin de relier les différentes régions R_c.

On déduit du corollaire que les cas (NS) et (FF) sont suffisants lorsque T_{init} et T_{cible} sont de genre 0.

Notons enfin que les cycles rencontrés après chaque étape d'expansion sont similaires aux extractions de boucle effectuées sur les surfaces triangulées par Lazarus *et al* [Lazarus 2001] dans le but de découper une surface pour générer un schéma polygonal. Leur algorithme évolue par croissance de régions à partir des mêmes opérations (NS), (FF) et (SD).

3.5.5 Résultats expérimentaux

Les résultats expérimentaux présentés dans le tableau 3.1 ont été obtenus sur un PC équipé d'un processeur Intel Core 2 Duo cadencé à 2 GHz et de 4 Go de RAM. Les triangulations dites *perturbées* sont générées en modifiant de manière aléatoire les triangulations initiales. Les triangulations étant récupérées à partir de fichiers *.off* ou *.obj*, les sommets sont numérotés de 0 à S, et nous appelons les triangulations dites *avec échange de connectivité*, les triangulations obtenues en échangeant la connectivité du sommet k avec le sommet $S - k$, pour $0 < k < \frac{S}{2}$. On peut remarquer que le temps d'exécution de l'algorithme

de détermination de séquences dépend principalement de la taille des triangulations considérées et de la différence de connectivité les deux maillages. En effet, lorsque la différence entre les deux triangulations est faible, la complexité semble linéaire, et lorsque la différence devient importante, elle semble devenir quadratique.

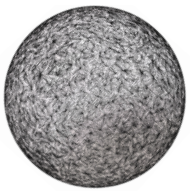
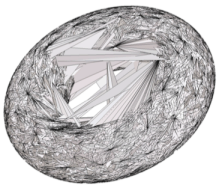
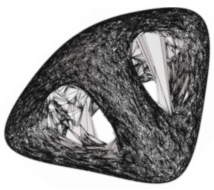
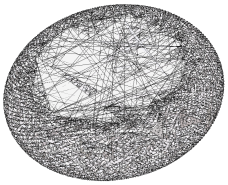
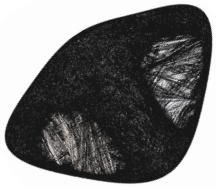
	 Sphère simple	 Bimba simple	 Tore simple	 Genre 3 simple
	 Sphère perturbée	 Bimba perturbée	 Tore perturbée	 Genre 3 perturbée
	 Sphère avec échanges de connectivité	 Bimba avec échanges de connectivité	 Tore avec échanges de connectivité	 Genre 3 avec échanges de connectivité
Genre	0	0	1	3
Sommets	21 872	192 135	12 150	179 708
Arêtes	65 610	576 399	36 450	539 136
T_{init}	Sphère simple	Bimba simple	Tore simple	Genre 3 simple
T_{cible}	Sphère simple	Bimba simple	Tore	Genre 3 simple
Bascules	0	0	0	0
Durée	220ms	3s	138ms	2s 400ms
T_{init}	Sphère simple	Bimba simple	Tore simple	Genre 3 simple
T_{cible}	Sphère perturbée	Bimba perturbée	Tore perturbée	Genre 3 perturbée
Bascules	114 417	645 970	125 148	492 386
Durée	916ms	9s 110ms	995ms	9s 539ms
T_{init}	Sphère simple	Bimba simple	Tore simple	Genre 3 simple
T_{cible}	Sphère avec échanges de connectivité	Bimba avec échanges de connectivité	Tore avec échanges de connectivité	Genre 3 avec échanges de connectivité
Bascules	11 950 677	79 856 110	2 144 832	95 018 006
Durée	8min 14s 210ms	16h 23min 58s	1min	19h 14min 06s

TABLE 3.1 – Résultats expérimentaux de l'algorithme pratique de détermination de séquences de bascules d'arêtes.

Les figures 3.28 et 3.27 illustrent le déroulement de l'algorithme pratique pour une surface de genre 0 et 2. La figure 3.29 illustre le déroulement de l'algorithme

pratique lorsque l'on démarre avec plusieurs faces orientées. Pour faciliter leur compréhension, nous avons donné la même géométrie aux triangulations T_{init} et T_{cible} et nous avons mis en correspondance les sommets possédant la même position.

FIGURE 3.27 – L'algorithme appliqué à une surface de genre 2. Le cas (SD) est utilisé au moment où la région non conforme correspond à l'union de plusieurs cycles d'arêtes.

Dans le cas particulier des triangulations planaires géométriques, nous avons comparé notre algorithme avec la méthode de détermination de séquences de bascules d'arêtes utilisant la triangulation de Delaunay comme pivot [Lawson 1977]. La comparaison a été effectuée sur un exemple ne contenant pas de sommets cocycliques (*cf* figure 3.30).

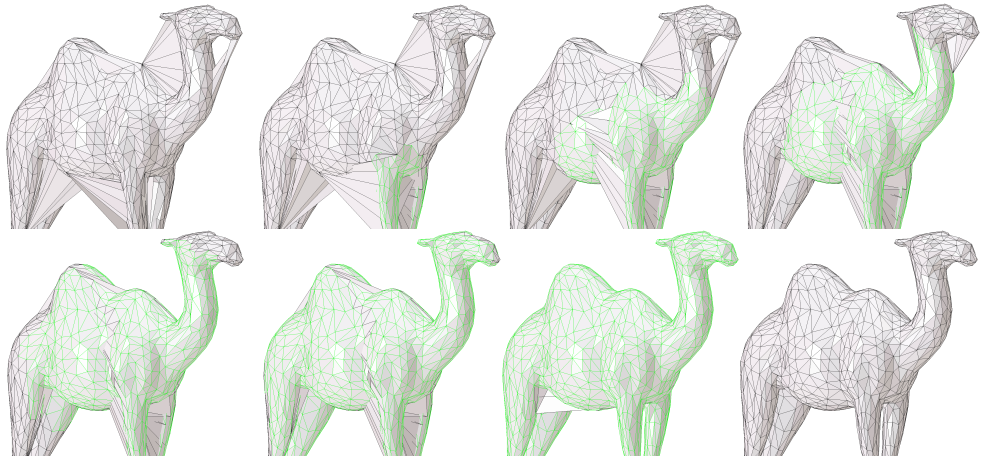


FIGURE 3.28 – L'algorithme appliqué à une surface de genre 0.

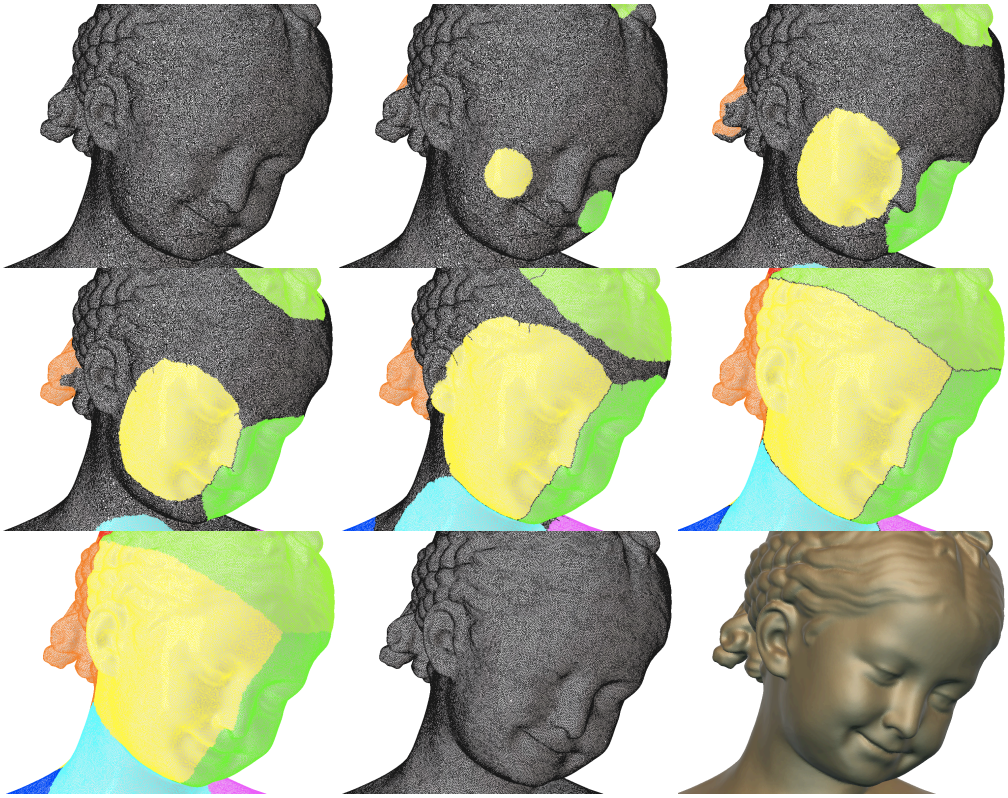


FIGURE 3.29 – Il est possible d'initialiser l'algorithme avec plusieurs faces de départ, afin de faire croître plusieurs régions conformes. Les faces de départ ont été sélectionnées par l'utilisateur mais elles peuvent être choisies de manière aléatoire en vérifiant que deux faces ne possèdent aucun sommet en commun.

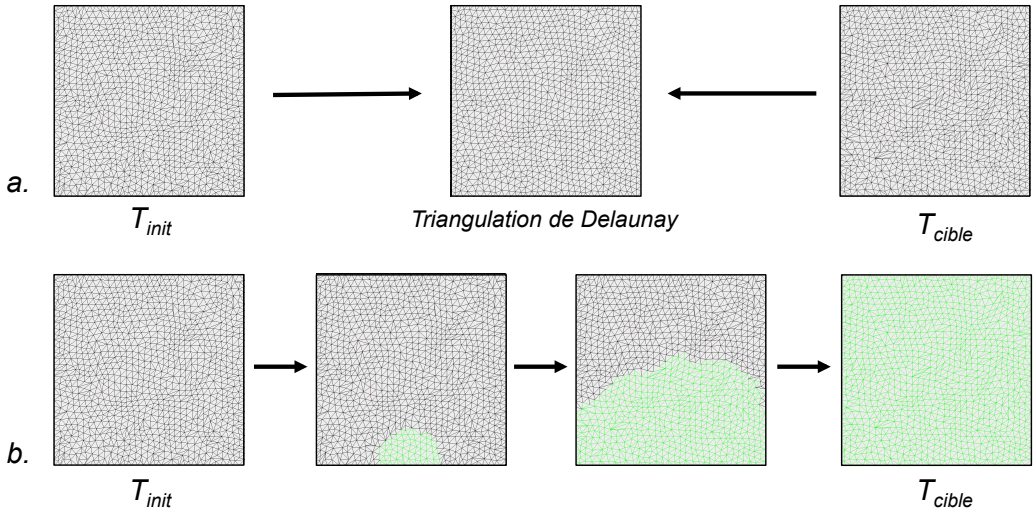


FIGURE 3.30 – Détermination d'une séquence de bascules d'arêtes entre T_{init} et T_{cible} : a) En utilisant une triangulation de Delaunay comme pivot [Lawson 1977], nous obtenons 353 bascules géométriques ; b) Avec notre algorithme, nous obtenons 294 bascules combinatoires.

3.6 Discussion

Étant données deux triangulations connexes de $\mathbb{T}$, possédant un même nombre de sommets, un même genre topologique et des bords consistents, nous venons de montrer les conditions nécessaires et suffisantes assurant l'existence d'une séquence de bascules d'arêtes. Nous en avons déduit un algorithme pratique permettant de générer une telle séquence, sans avoir à connaître le genre topologique commun.

Nous faisons remarquer que l'algorithme pratique peut être légèrement modifié pour être utilisé sans mise en correspondance des sommets de T_{init} et T_{cible}. Soit (abc) une face de T_{cible}, nous choisissons trois sommets de $T_{courant}$ que nous mettons en correspondance avec a, b et c, puis initialisons la croissance de régions en construisant dans $T_{courant}$ la face orientée correspondante. La mise en correspondance des sommets restants s'effectuera uniquement pendant le cas de création d'une face vers un nouveau sommet (NS). Soit (def) une face de T_{cible} tel que les sommets d et e de T_{cible} sont déjà en correspondance avec deux sommets de $T_{courant}$. Nous mettons le sommet f en correspondance avec un nouveau sommet de $T_{courant}$ incident à aucune arête contrainte, puis nous construisons la face orientée correspondante à (def) dans $T_{courant}$. Un nouveau sommet de $T_{courant}$ peut être choisi de manière aléatoire, ou à partir de critères géométriques ou d'une distance géodésique.

Une perspective intéressante est d'utiliser cette nouvelle version de l'algorithme pratique pour générer une mise en correspondance entre deux triangulations.

Chapitre 4

Réduction de séquences de bascules d'arêtes

La bascule d'arête peut-être vue comme un outil permettant de coder la différence entre deux triangulations et il est important de pouvoir contrôler la longueur d'une séquence, dans l'optique de concevoir des algorithmes de compression. Le problème de la détermination d'une séquence minimale de bascules d'arêtes entre deux triangulations a été prouvé être un problème NP-complet dans certaines classes de triangulations, ce qui rend nécessaire le développement d'heuristiques. Dans ce chapitre, nous proposons une solution consistant à réduire les séquences de bascules d'arêtes dans la classe des triangulations orientées combinatoirement *manifold*. Cette approche de réduction est également valable dans la classe des triangulations géométriques orientées ou la classe des triangulations de Wagner.

La nouvelle approche que nous proposons est fondée sur l'affectation d'un indice différent à chaque arête, avec transfert de l'indice d'une arête détruite sur l'arête créée lors d'une bascule. Ceci donne un moyen de suivre une arête pendant une séquence de bascules et permet le développement de propriétés combinatoires très simples :

- Etant donné une triangulation initiale T, et une séquence de bascules d'arêtes ϕ, nous introduisons trois outils simples qui peuvent être utilisés pour générer une nouvelle séquence γ telle que ϕ et γ transforment T en une triangulation similaire, modulo une permutation des indices. Dans le cas où T est une triangulation d'un n-gone convexe, nous prouvons que ces trois outils peuvent être utilisés pour passer d'une séquence ϕ à n'importe quelle autre séquence γ, telle que ϕ et γ transforme T en une même triangulation, modulo une permutation des indices.
- À partir de ces trois outils, nous proposons un algorithme efficace pour réduire une séquence de bascules d'arêtes donnée, de manière à satisfaire un critère d'optimum local. Cet algorithme est polynomial, et s'utilise aussi bien sur la classe des triangulations orientées combinatoirement *manifold* que sur la classe des triangulations géométriques orientées et sur la classe des triangulations de Wagner. Son efficacité est illustrée par des résultats expérimentaux.

4.1 Etat de l'art sur la recherche d'une séquence minimale de bascules d'arêtes

4.1.1 Détermination d'une séquence minimale de bascules d'arêtes

Le problème de la détermination d'une séquence de bascules minimale est délicat. Cette séquence varie en fonction des triangulations considérées, de la classe de triangulation choisie et, en général, elle n'est pas unique. La seule méthode connue consiste, pour une triangulation donnée, à construire le *graphe de bascules*, c'est-à-dire le graphe dont les sommets sont les triangulations possibles et dans lequel deux sommets sont reliés par une arête s'il existe une unique bascule permettant de transformer la première triangulation en la deuxième (*cf* figure 4.1). Les séquences de bascules minimales entre deux triangulations sont donc les chemins les plus courts qui relient les deux triangulations dans le graphe. Cependant, il est difficile de déterminer ces séquences car le nombre de triangulations possibles évolue de façon exponentielle par rapport au nombre de sommets, ce qui rend le graphe de bascules rapidement impossible à construire. De plus, en fonction de la classe de triangulations choisie, nous ne pouvons pas toujours savoir si ce graphe est connexe ou pas. Nous disposons néanmoins de ces informations dans le cas du graphe de bascules d'un n-gone convexe triangulé : nous savons que si $n > 4$, il est $(n-3)$-sommet-connexe [Hurtado 1999], qu'il a pour squelette un $(n-3)$-polytope convexe appelé associahedron, et que son groupe d'automorphisme est le groupe dihedral D_n [Lee 1988, Hurtado 1999].

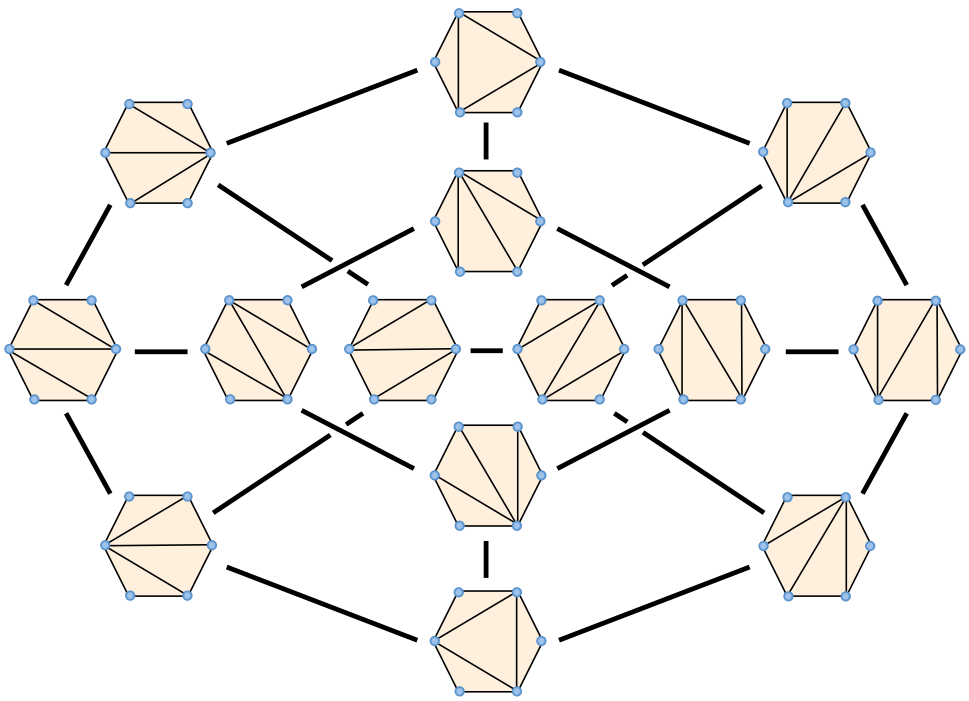


FIGURE 4.1 – Graphe de bascules des triangulations d'un 6-gone convexe.

4.1.2 La distance de bascules d'arêtes

Il existe des résultats concernant un problème sous-jacent à la recherche de séquences minimales : la détermination de la *distance de bascules* entre deux triangulations, à savoir le nombre de bascules constituant la séquence minimale. La distance de bascules a pour particularité de définir un espace métrique sur l'ensemble des triangulations, et elle peut permettre la détermination du *diamètre* d'un ensemble de triangulations, c'est-à-dire la valeur maximale que peut atteindre la distance de bascules pour un couple de triangulations du même ensemble. Cette valeur donne des informations sur le nombre de bits minimal nécessaire pour coder une séquence de bascules d'arêtes entre deux triangulations.

Il n'existe pas d'algorithme efficace pour déterminer la distance de bascule, mais de nombreux travaux en fournissent une approximation. Dans le classe des triangulations géométrique orientées, et plus précisément dans le cas d'une triangulation d'un n-gone convexe, la question consistant à savoir si la détermination de cette distance est un problème NP-complet est toujours ouverte [Sleator 1987]. Cependant, Pournin [Pournin 2012] montre que pour un n-gone avec $n > 13$, $2n - 10$ bascules sont suffisantes et parfois nécessaires pour transformer une triangulation en une autre. Dans le cas des triangulations géométriques planaires, la détermination de la distance de bascules est un problème NP-complet [Lubiw 2012, Pilz 2012], ainsi que pour les polygones simples [Aichholzer 2012]. Lawson[Lawson 1977] montre que pour deux triangulations planaires composées de n sommets, $O(n^2)$ bascules sont suffisantes. Hanke *et al.* [Hanke 1996] montrent que la distance de bascules est inférieure au nombre d'intersections d'arêtes lorsque l'on superpose les deux triangulations. Si la triangulation ne possède pas cinq sommets formant un pentagone vide, Eppstein [Eppstein 2010] montre qu'il est possible de la déterminer en temps $O(n^2)$. Pour le cas général, il propose un algorithme polynomial calculant une borne inférieure.

Dans la classe de triangulations de Wagner, il existe des résultats sur la détermination de la distance de bascules entre deux triangulations d'une surface de genre 0. La meilleure borne supérieure connue est $5.3n - 24.4$ bascules d'arêtes [Bose 2011] et en ce qui concerne la borne inférieure du diamètre, il existe des paires de triangulations qui requièrent $2n - 15$ bascules [Komuro 1997].

La détermination d'une séquence minimale est un problème de difficulté au moins NP-complet car la détermination de la distance de bascules est NP-complet dans de nombreux cas. Il est donc naturel de s'intéresser au développement d'algorithmes réduisant efficacement une séquence plutôt que de la minimiser.

4.2 Bascules d'arêtes et permutation d'indices

4.2.1 Cadre de l'étude

4.2.1.1 Triangulations possédant des arêtes indexées

Dans ce chapitre, nous exploitons l'idée qui consiste à identifier une arête d'une triangulation par un indice alors qu'habituellement on la désigne par ses extrémités. Comme il est spécifié dans l'introduction, chaque arête n'appartenant pas au bord (car elles ne sont jamais basculables) possède un indice différent, qu'elle conservera après une bascule. Les résultats que nous présentons sont appliqués dans la classe générale des triangulations combinatoires introduite en 2.1.1, notée $\mathcal{T}$, et se restreignent sans problème à toutes ses sous-classes : triangulations orientées combinatoirement *manifold*, des triangulations géométriques orientées et des triangulations de Wagner. Nous nommons par $\mathcal{T}_{ind}$ la classe générale des triangulations combinatoires composées d'arêtes indexées.

Dans la suite, nous dirons que deux triangulations aux arêtes indexées T_1 et $T_2 \in \mathcal{T}_{ind}$ sont égales, noté $T_1 = T_2$, si elles sont identiques et possèdent les mêmes indices pour chaque arête.

4.2.1.2 Opération de bascule d'arêtes indexées

Soit $T \in \mathcal{T}_{ind}$ une triangulation contenant une arête i entre les sommets v_0 et v_1. L'opération de bascule de l'arête i est notée $\mathcal{F}_i$ et la triangulation résultante après bascule de l'arête i dans la triangulation T est notée $\mathcal{F}_i(T)$. L'indice i de l'arête (v_0v_1) basculée est transféré sur la nouvelle arête créée (*cf* figure 4.2).

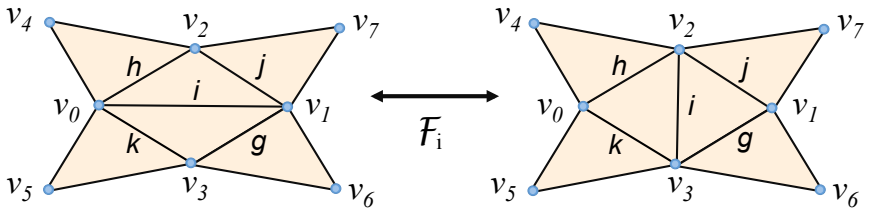


FIGURE 4.2 – Bascule de l'arête i.

Si l'arête i est basculable dans $T \in \mathcal{T}_{ind}$, alors i est aussi basculable dans $\mathcal{F}_i(T)$, et $\mathcal{F}_i(\mathcal{F}_i(T)) = T$. Autrement dit, $\mathcal{F}_i$ est une involution.

4.2.1.3 Composition de bascules d'arêtes

Soit $T \in \mathcal{T}_{ind}$ contenant les arêtes $i_1, i_2, \ldots i_n$. Nous définissons la *composition* de deux bascules comme $\mathcal{F}_{i_2}(\mathcal{F}_{i_1}(T)) = (\mathcal{F}_{i_2} \circ \mathcal{F}_{i_1})(T)$, c'est-à-dire que l'on effectue dans un premier temps la bascule de l'arête i_1, puis celle de l'arête i_2. La *séquence de bascules* ϕ des arêtes $i_1, \ldots, i_n$ de T correspond à $\phi = (\mathcal{F}_{i_n} \circ \mathcal{F}_{i_{n-1}} \circ \ldots \circ \mathcal{F}_{i_1})$ dans laquelle chaque bascule est autorisée. La *séquence inverse de* ϕ est $\phi^{-1} =$

$(\mathcal{F}_{i_1} \circ \mathcal{F}_{i_2} \circ \ldots \circ \mathcal{F}_{i_n})$, et elle vérifie la propriété que $(\phi^{-1} \circ \phi)(T) = (\phi \circ \phi^{-1})(T) = T$ pour toute triangulation $T \in \mathcal{T}_{ind}$.

En effet, on observe que $(\phi^{-1} \circ \phi)(T) = (\mathcal{F}_{i_1} \circ \ldots \circ \mathcal{F}_{i_n} \circ \mathcal{F}_{i_n} \circ \ldots \circ \mathcal{F}_{i_1})(T)$ se réduit à $(\mathcal{F}_{i_1} \circ \ldots \circ \mathcal{F}_{i_{n-1}} \circ \mathcal{F}_{i_{n-1}} \circ \ldots \circ \mathcal{F}_{i_1})(T)$ quelle que soit la triangulation $T \in \mathcal{T}_{ind}$, car $\mathcal{F}_{i_n}$ est une involution et $(\mathcal{F}_{i_n} \circ \mathcal{F}_{i_n})(T) = T$. Par induction sur la sous-séquence, on obtient $(\phi \circ \phi^{-1})(T) = (\phi^{-1} \circ \phi)(T) = T$.

Dans la suite, nous dirons que *la séquence* $(\mathcal{F}_{i_f} \circ \mathcal{F}_{i_{f-1}} \circ \ldots \circ \mathcal{F}_{i_1})$ *est basculable dans* T si $\mathcal{F}_{i_1}$ est basculable dans T, $\mathcal{F}_{i_2}$ est basculable dans $\mathcal{F}_{i_1}(T)$, ..., et $\mathcal{F}_{i_f}$ est basculable dans $(\mathcal{F}_{i_{f-1}} \circ \ldots \circ \mathcal{F}_{i_1})(T)$.

4.2.1.4 Commutativité au sein d'une séquence

Si i est une arête de T, on définit le *support* de la bascule $\mathcal{F}_i$, noté $supp(i, T)$ comme l'ensemble composé uniquement des deux faces incidentes à l'arête i dans la triangulation T.

Soient i et j deux arêtes basculables de T, et l'on suppose qu'il n'existe aucune face de T incidente à la fois à i et j, autrement dit, le cardinal de $supp(i, T) \bigcap supp(j, T)$ est nul en terme de nombre de faces : $|supp(i, T) \bigcap supp(j, T)| = 0$. On observe que la séquence $(\mathcal{F}_i \circ \mathcal{F}_j)$ peut être appliquée à T et $(\mathcal{F}_i \circ \mathcal{F}_j(T)) = (\mathcal{F}_j \circ \mathcal{F}_i)(T)$, c'est-à-dire que $\mathcal{F}_j$ et $\mathcal{F}_i$ *commutent dans* T (*cf* figure 4.3).

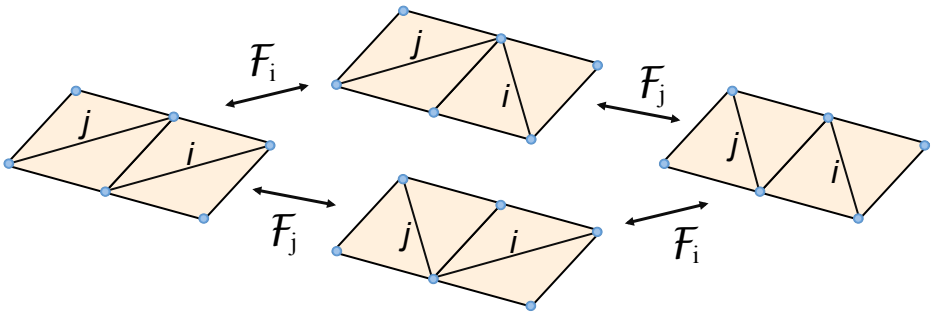


FIGURE 4.3 – $(\mathcal{F}_i \circ \mathcal{F}_j)(T) = (\mathcal{F}_j \circ \mathcal{F}_i)(T)$ lorsque $|supp(i, T) \bigcap supp(j, T)| = 0$.

On peut déduire des règles de commutativité entre deux bascules au sein d'une séquence :
soient ϕ, μ et γ trois séquences de bascules telles que $\phi(T) = (\mu \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \gamma)(T)$, on a $\phi(T) = (\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \gamma)(T)$ si $|supp(i, \gamma(T)) \bigcap supp(j, \gamma(T))| = 0$.

Dans la suite, nous dirons que deux arêtes i et j sont voisines dans la triangulation T, si $|supp(i, T) \bigcap supp(j, T)| > 0$.

4.2.2 Opération de transposition d'indices d'arêtes

Nous introduisons maintenant $\mathcal{P}_{(i,j)}$, l'opération de *transposition d'indices* appliquée aux arêtes i et j : si $T \in \mathcal{T}_{ind}$, $\mathcal{P}_{(i,j)}(T)$ est la triangulation obtenue à partir de T en transposant les indices i et j. De plus, nous définissons la

composition de deux transpositions par $(\mathcal{P}_{(i,j)} \circ \mathcal{P}_{(k,l)})(T) = \mathcal{P}_{(i,j)}(\mathcal{P}_{(k,l)}(T))$.

Nous allons travailler avec des séquences composées à la fois d'opérations de bascules et de transpositions, et nous pourrons les commuter de la manière suivante :

$$(\mathcal{P}_{(i,j)} \circ F_k)(T) = (F_{\theta_{(i,j)}(k)} \circ \mathcal{P}_{(i,j)})(T)$$

où $\theta_{(i,j)}(k)$ correspond à j si $k = i$, à i si $k = j$ et à k sinon (*cf* figure 4.4).

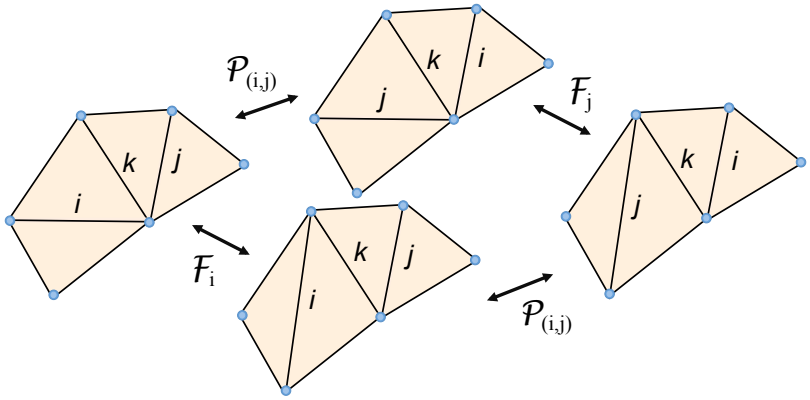


FIGURE 4.4 – $(\mathcal{P}_{(i,j)} \circ F_k)(T) = (F_{\theta_{(i,j)}(k)} \circ \mathcal{P}_{(i,j)})(T)$ où $\theta_{(i,j)}(k)$ correspond à j si $k = i$, à i si $k = j$ et k sinon.

En effet, si $k \neq i$ et $k \neq j$, le résultat est équivalent à dire que la bascule de k et la transposition de i et j commutent car les deux opérations sont clairement indépendantes. Les cas $k = i$ et $k = j$ sont symétriques, et en considérant par exemple que $k = i$, l'égalité devient $(\mathcal{P}_{(i,j)} \circ \mathcal{F}_i)(T) = (\mathcal{F}_j \circ \mathcal{P}_{(i,j)})(T)$, et reste vraie car la bascule de l'arête est une opération combinatoire qui ne dépend clairement pas de l'indice de l'arête.

Nous déduisons aussi directement que $(F_k \circ \mathcal{P}_{(i,j)})(T) = (\mathcal{P}_{(i,j)} \circ F_{\theta_{(i,j)}(k)})(T)$.

Nous avons introduit la notion de transposition d'indices car il existe une relation entre cette opération et les bascules d'arêtes :

Proposition IV.1 : Soit T_5 une triangulation d'un polygone simple composé de 5 sommets tel que ses arêtes internes sont basculables pour toutes les configurations possibles. Si i et j sont les deux arêtes internes de T_5, on a :

$$(\mathcal{P}_{(i,j)})(T_5) = (\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)(T_5) = (\mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j)(T_5).$$

Nous proposons deux démonstrations différentes pour cette proposition :

Démonstration. Le polygone choisi est intéressant car il n'y a aucune restriction structurelle sur la propriété d'une arête d'être basculable ou non, dans ses triangulations. La figure 4.7 illustre le fait que $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)$ engendre une

transposition des indices i et j lorsqu'on est dans un polygone contenant 5 sommets, ce qui prouve le résultat car les triangulations d'un polygone simple sont similaires modulo une rotation. En particulier, ce résultat est toujours vrai dans le cas d'un 5-gone convexe. □

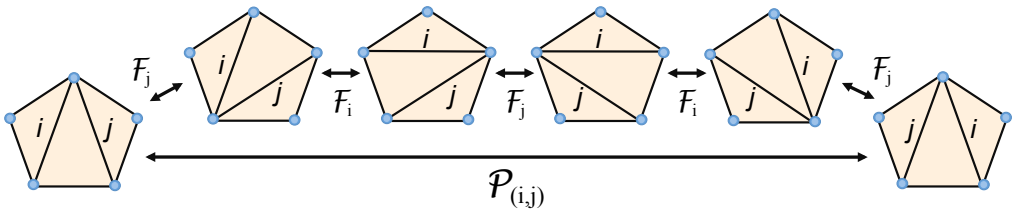


FIGURE 4.5 – Illustration de $\mathcal{P}_{(i,j)}(T_5) = (\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)(T_5)$.

Démonstration. On peut aussi démontrer cette proposition de manière plus formelle, en montrant le lien avec la théorie des groupes. On définit une relation d'équivalence entre deux séquences de bascules d'arêtes : ϕ_1 et ϕ_2 sont équivalentes si $\phi_1(T) = \phi_2(T)$ quelque soit la triangulation T_5 du 5-gone. Avec cette relation, les classes d'équivalences générées par l'ensemble des séquences de bascules d'arêtes appliquées à un 5-gone triangulé forment un groupe (au sens algébrique) avec l'opération de composition. Ce groupe est isomorphe au groupe dihédral D_5 et peut être généré par la séquence $(\mathcal{F}_j \circ \mathcal{F}_i)$ qui est un élément d'ordre 5 correspondant aux rotations des deux arêtes internes et $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)$ qui est un élément d'ordre 2 qui transpose les indices i et j. L'équivalence entre $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)$ et $\mathcal{P}_{(i,j)}$ est vérifiée pour toutes les triangulations du 5-gone. □

Proposition IV.2 : Soit $T \in \mathcal{T}_{ind}$ contenant deux arêtes internes voisines indexées par i et j. Les deux indices d'arêtes i et j peuvent être transposés si $|supp(i,T) \bigcap supp(j,T)| = 1$ et si l'une des séquences $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)$ ou $(\mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j)$ est basculable par rapport à la triangulation $T \in \mathcal{T}_{ind}$.

Démonstration. Nous savons que $|supp(i,T) \bigcap supp(j,T)| = 1$ et les arêtes i et j sont des arêtes n'appartenant pas au bord, donc $|supp(i,T) \bigcup supp(j,T)| = 3$. Nous en déduisons que la sous-triangulation formée des faces de $supp(i,T) \bigcup supp(j,T)$ ainsi que des arêtes et des sommets incidents, est une triangulation d'un polygone simple composé de 5 sommets qui ne sont pas forcément distincts et dont ses arêtes internes sont i et j. De plus, comme la séquence $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)$ (resp. $(\mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j)$) est basculable, elle transpose les indices i et j. □

Il est donc possible de transposer des arêtes voisines dans des configurations plus générales que celles énoncées dans la proposition IV.1 (*cf* figure 4.6).

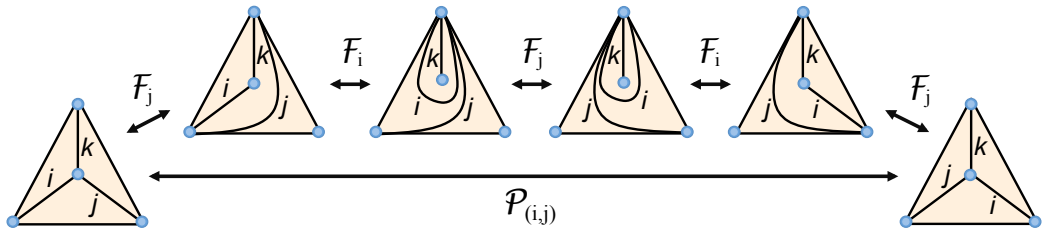


FIGURE 4.6 – Si $\mathcal{T}_{ind}$ autorise les triangulations de type combinatoires pouvant posséder des arêtes reliant une même extrémités on a : $\mathcal{P}_{(i,j)}(T) = (\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)(T)$ pour $T \in \mathcal{T}_{ind}$.

4.2.3 Algorithme de permutation d'indices d'arêtes

Les résultats précédents montrent qu'une séquence de bascules d'arêtes ne permet pas uniquement de modifier la connectivité d'une triangulation, mais transposent également les indices. Nous présentons maintenant un algorithme simple pour permuter l'ensemble des indices d'une triangulation.

Nous avons besoin de transposer les indices de deux arêtes i et j qui ne sont pas voisines, et nous procédons de la manière suivante : soient $T \in \mathcal{T}_{ind}$, et une séquence d'arêtes $(i_1, i_2, ..., i_k)$ telle que $i_1 = i$, $i_k = j$ et pour $l \in \{i_1 ... i_k\}$, les arêtes e_l et e_{l+1} sont voisines et transposables par la séquence $\mathcal{P}_{(e_l,e_{l+1})}(T) = (\mathcal{F}_{e_l} \circ \mathcal{F}_{e_{l+1}} \circ \mathcal{F}_{e_l} \circ \mathcal{F}_{e_{l+1}} \circ \mathcal{F}_{e_l})(T)$. On peut transposer les indices des arêtes i et j par la séquence suivante :

$$\mathcal{P}_{(i_1,i_k)}(T) = (\mathcal{P}_{(i_1,i_2)} \circ ... \circ \mathcal{P}_{(i_{k-2},i_{k-1})} \circ \mathcal{P}_{(i_{k-1},i_k)} \circ \mathcal{P}_{(i_{k-2},i_{k-1})} \circ ... \circ \mathcal{P}_{(i_1,i_2)})(T)$$

avec $\mathcal{P}_{(e_l,e_{l+1})} = (\mathcal{F}_{e_l} \circ \mathcal{F}_{e_{l+1}} \circ \mathcal{F}_{e_l} \circ \mathcal{F}_{e_{l+1}} \circ \mathcal{F}_{e_l})$ pour $l \in i_1 ... i_k$.

L'existence d'une séquence de bascules d'arêtes traduisant n'importe quelle permutation d'indice dépend de la classe de triangulation considérée. Nous avons les résultats suivants :

Proposition IV.3 : Soit T est une triangulation orientée combinatoirement *manifold* à laquelle on indexe ses arêtes internes. Si chaque sommet de T est de valence strictement supérieure à 3, alors nous pouvons générer n'importe quelle permutation d'indices.

Démonstration. Si pour chaque couple d'arêtes voisines internes, on peut transposer leur indice, alors nous pouvons générer n'importe quelle transposition et donc n'importe quelle permutation.

Soient deux arêtes internes voisines i et j. On a $|supp(i,T) \bigcap supp(j,T)| = 1$ car la valence de chaque sommet est supérieure ou égale à 3. De plus, durant l'exécution

des bascules de la séquence $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)$ la valence des sommets adjacents à $(supp(i,T) \bigcup supp(j,T))$ pourra être soustrée de 2 au maximum. On en déduit qu'après chaque bascule de la séquence, la valence de chaque sommet est supérieure ou égale 2, ce qui signifie qu'aucune bascule est interdite. D'après la Proposition IV.2, la séquence génère une transposition laissant invariant la valence de chaque sommet. □

Proposition IV.4 : Soit T une triangulation géométrique planaire auquelle on indexe ses arêtes internes. Si l'union des supports de tous les couples d'arêtes voisines forment un 5-gone convexe dans T, alors nous pouvons générer n'importe quelle permutation d'indices.

Démonstration. Si pour chaque couple d'arêtes voisines internes, on peut transposer leur indice, alors nous pouvons générer n'importe quelle transposition et n'importe quelle permutation.

Nous savons que pour tout couple d'arête i et j, il existe une triangulation d'un 5-gone convexe incluse dans T tel que i et j sont les arêtes internes. Nous en déduisons d'après la Proposition IV.1, que les arêtes i et j sont transposables. □

Dans chacun des cas précédents, on peut appliquer l'algorithme 9.

Algorithme 9 Détermination de séquences de bascules d'arêtes entre deux triangulations identiques modulo une permutation des indices

```
Insérer toutes les arêtes de la triangulation initiale dans une pile
tant que la pile n'est pas vide faire
  dépiler la première arête (ab) d'indice i
  si l'indice souhaitée de (ab) est j alors
    trouver l'arête (cd) indexée par j
    transposer les indices des arêtes (ab) et (cd)
  fin si
fin tant que
Retourner la séquence de bascules d'arêtes
```

En combinant l'utilisation d'un algorithme de détermination de séquences de bascules entre deux connectivités différentes et l'algorithme 9, on peut transformer une triangulation possédant des arêtes indexées T_{init} en une nouvelle triangulation T_{cible} lorsque T_{init} et T_{cible} satisfont les hypothèses de la proposition IV.3 ou IV.4.

La possibilité de "*transporter*" des arêtes sur une triangulation signifie que les bascules d'arêtes peuvent servir à coder davantage d'information qu'un simple changement de connectivité. Les indices peuvent être utilisés pour coder des attributs tels que la couleur. Dans la partie suivante, nous présentons une application : la réduction de séquences de bascules.

4.3 Séquences de bascules d'arêtes équivalentes

Nous reprenons les notations introduites dans le chapitre précédent et nous travaillons dans la classe de triangulation $\mathcal{T}_{ind}$ présentée dans la partie 4.2.1.1.

4.3.1 Séquences faiblement et fortement équivalentes

Nous définissons deux relations d'équivalence entre les séquences de bascules appliquées à une triangulation possédant des arêtes indexées T.

Soient ϕ_1 et ϕ_2 deux séquences de bascules d'arêtes telles que $\phi_1(T)$ et $\phi_2(T)$ sont deux triangulations d'un même ensemble de sommets.

- ϕ_1 est *fortement équivalente à* ϕ_2 lorsqu'elles sont appliquées à T si $\phi_1(T) = \phi_2(T)$,

- ϕ_1 est *faiblement équivalente à* ϕ_2 lorsqu'elles sont appliquées à T si $\phi_1(T)$ et $\phi_2(T)$ correspondent à la même triangulation modulo une permutation des indices des arêtes ; nous notons $\phi_1(T) \simeq \phi_2(T)$ dans ce cas.

Dans la partie suivante, nous nous intéressons à la génération de séquences faiblement et fortement équivalentes pour une triangulation T donnée. Nous faisons remarquer que l'équivalence forte implique l'équivalence faible et que les séquences minimales entre les triangulations à arêtes non indexées T et $\phi(T)$ sont faiblement équivalentes à ϕ par rapport à T.

4.3.2 Outils de génération de séquences équivalentes

Nous présentons trois outils dérivant des propriétés de l'opération de bascule d'arête présentées dans la partie 4.2. Nous les utilisons pour générer des séquences faiblement ou fortement équivalentes à une séquence donnée ϕ appliquée à une triangulation $T \in \mathcal{T}_{ind}$. Nous montrons que si T est une triangulation d'un n-gone convexe, les trois outils permettent de générer toutes les séquences faiblement équivalentes par rapport à T à une séquence donnée ϕ. Dans la suite, nous considérons que toutes les séquences sont composées d'arêtes basculables.

1er outil (exploitant la commutativité) :

S'il existe deux séquences μ et ν telles que $\phi(T) = (\mu \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)(T)$ et que $|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 0$, alors la séquence $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)$ est fortement équivalente à ϕ par rapport à T. Donc si la séquence ϕ appliquée à T contient des bascules que l'on peut commuter, on peut générer de nouvelles séquences fortement équivalentes à ϕ. Dans la suite, si l'on peut transformer une première séquence en une seconde en utilisant ce premier outil, on dira de ces séquences qu'elles sont *fortement équivalentes par commutativité.*

2ème outil (exploitant l'involution) :

Nous savons que la propriété $(\mathcal{F}_i \circ \mathcal{F}_i)(T) = T$ est vraie pour toute triangulation $T \in T_{ind}$ contenant une arête indexée par i. Nous pouvons donc générer de nouvelles séquences fortement équivalentes à ϕ en insérant la séquence $(\mathcal{F}_i \circ \mathcal{F}_i)$ au sein de ϕ (augmentation de la séquence de deux bascules). Inversement, si ϕ contient deux occurrences de $\mathcal{F}_i$, et qu'il existe des séquences de bascules μ et ν telles que $\phi(T)$ est fortement équivalente par commutativité à $(\mu \circ \mathcal{F}_i \circ \mathcal{F}_i \circ \nu)(T)$, alors on peut générer une nouvelle séquence $(\mu \circ \nu)$ fortement équivalente à ϕ et contenant deux bascules de moins.

3ème outil (exploitant la transposition des indices) :

Proposition IV.5 : S'il existe μ et ν telles que ϕ est fortement équivalente par commutativité à $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)$ par rapport à T, avec $|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 1$, et que la séquence $(\mathcal{F}_j \circ \mathcal{F}_i)$ est basculable dans $\nu(T)$, alors $\phi(T) = (\mu \circ \mathcal{P}_{(i,j)} \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)(T)$.

Démonstration. : $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)(T) = (\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ (\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_j \circ \mathcal{F}_i) \circ \nu)(T) = (\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)(T)$ avec $|supp(i, (\mathcal{F}_j \circ \mathcal{F}_i \circ \nu)(T)) \bigcap supp(j, (\mathcal{F}_j \circ \mathcal{F}_i \circ \nu)(T))| = 1$ car $|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 1$. La configuration locale des supports est un polygone simple composé de 5 sommets dans lequel les arêtes i et j sont basculables à chaque instant. D'après la proposition IV.1, nous avons donc $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)(T) = (\mu \circ \mathcal{P}_{(i,j)} \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)(T)$ (*cf* figure 4.7). □

Nous notons $\mathcal{P}_{(i,j)}(\mu)$ la séquence μ pour laquelle toutes les occurrences de $\mathcal{F}_i$ ont été remplacées par $\mathcal{F}_j$ et réciproquement.

La propriété précédente implique que si ϕ est fortement équivalente par commutativité à $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)$, alors ϕ est faiblement équivalente à $(\mathcal{P}_{(i,j)}(\mu) \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)$ par rapport à T. La séquence générée possède une bascule de moins que ϕ.

Inversement, s'il existe μ et ν telles que ϕ est fortement équivalente par commutativité à $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)$ par rapport à T, avec $|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 1$ et la séquence $(\mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j)$ est basculable dans $\nu(T)$, alors ϕ est faiblement équivalente à $(\mathcal{P}_{(i,j)}(\mu) \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)(T)$ par rapport à T. La séquence générée possède une bascule de plus que ϕ.

4.3.3 Séquences de bascules d'arêtes directement réductibles

Nous présentons un cas particulier de séquences de bascules d'arêtes. Soient $T \in \mathcal{T}_{ind}$ contenant les arête indexées i et j, et ϕ_1, ϕ_2 deux séquences de bascules. On dira que ϕ_2 est une *réduction directe* de ϕ_1 s'il existe μ et ν vérifiant l'une des conditions suivantes :

- ϕ_1 est fortement équivalente par commutativité à $(\mu \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \nu)$ avec $|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 1$ et $\phi_2 = (\mathcal{P}_{(i,j)}(\mu) \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)$ (équivalence faible entre ϕ_1 et ϕ_2 dans T).

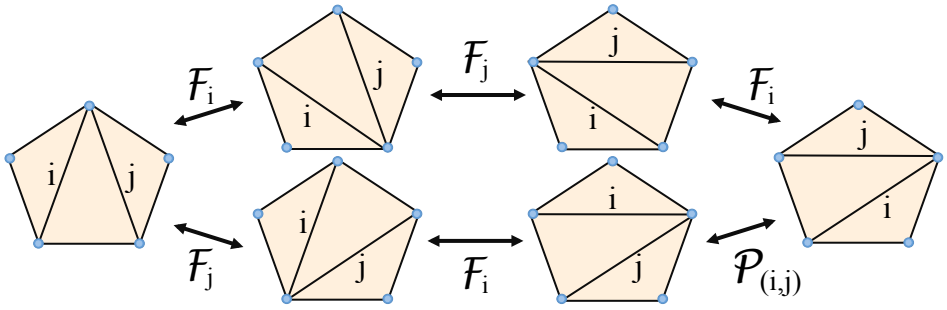


FIGURE 4.7 – Dans le cas du 5-gone convexe triangulé, $(\mathcal{F}_i \circ \mathcal{F}_j \circ \mathcal{F}_i)(T) = (\mathcal{P}_{(i,j)} \circ \mathcal{F}_i \circ \mathcal{F}_j)(T)$

- ϕ_1 est fortement équivalente par commutativité à $(\mu \circ \mathcal{F}_i \circ \mathcal{F}_i \circ \nu)$ et $\phi_2 = (\mu \circ \nu)$ (équivalence forte entre ϕ_1 et ϕ_2 dans T).

Nous dirons qu'une séquence est *directement réduite* si elle ne possède aucune réduction directe. À partir des outils présentés et d'une séquence donnée, nous sommes toujours capable de générer une séquence réduite en construisant successivement des réductions directes. Cependant il existe plusieurs séquences réduites faiblement équivalentes et les réductions directes peuvent générer des séquences directement réduites différentes en fonction de l'ordre et des bascules utilisées lors des réductions. De plus, les séquences directement réduites ne sont pas toutes égales à une séquence minimale (*cf* figure 4.8).

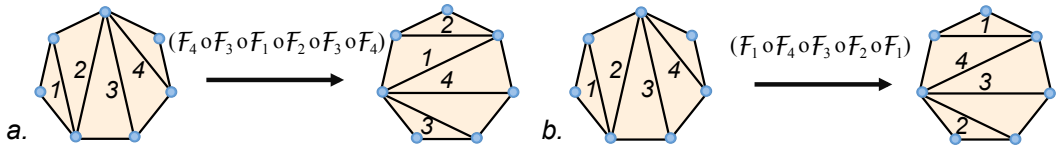


FIGURE 4.8 – En a) la séquence est directement réduite mais elle n'est pas minimale ; en b) la séquence contient une bascule de moins.

4.3.4 Séquences équivalentes dans le cas d'un n-gone convexe

Dans cette partie, nous nous intéressons au cas particulier des triangulations d'un n-gone convexe, qui ont la particularité d'être composées d'arêtes internes basculables à chaque instant. Dans le cas des triangulations combinatoires, les résultats présentés sont aussi valables dans la classe des triangulations combinatoires d'un polygone simple. Nous énonçons un théorème illustrant les possibilités offertes par les trois outils de génération de séquence.

4.3.4.1 Théorème sur la génération de la totalité des séquences équivalentes

Théorème IV.1 : Soient T une triangulation indexée d'un n-gone convexe et ϕ_1 une séquence de bascules d'arêtes. Les trois outils exploitant la commutativité, l'involution de la bascule d'arête et les transpositions d'indices, permettent de générer toutes les bascules d'arêtes faiblement équivalentes à ϕ_1 par rapport à T.

Le théorème nous permet de comprendre le passage d'une séquence de bascules à une autre, mais il n'est pas toujours aisé de trouver comment utiliser ces outils entre deux séquences faiblement équivalentes. En particulier, il est difficile de générer les séquences minimales. Si nous considérons l'exemple présenté par la figure 4.8, voici le cheminement à suivre pour transformer la séquence en a) jusqu'à la séquence minimale en b) :

$$\begin{aligned}(\mathcal{F}_4 \circ \mathcal{F}_3 \circ \mathcal{F}_1 \circ \mathcal{F}_2 \circ \mathcal{F}_3 \circ \mathcal{F}_4)(T) &= (\mathcal{P}_{(1,2)} \circ \mathcal{F}_4 \circ \mathcal{F}_3 \circ \mathcal{F}_1 \circ \mathcal{F}_2 \circ \mathcal{F}_1 \circ \mathcal{F}_3 \circ \mathcal{F}_4)(T)\\ &= (\mathcal{P}_{(1,2)} \circ \mathcal{F}_4 \circ \mathcal{F}_1 \circ \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_3 \circ \mathcal{F}_1 \circ \mathcal{F}_4)(T)\\ &= (\mathcal{P}_{(1,2)} \circ \mathcal{P}_{(2,3)} \circ \mathcal{F}_4 \circ \mathcal{F}_1 \circ \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1 \circ \mathcal{F}_4)(T)\\ &= (\mathcal{P}_{(1,2)} \circ \mathcal{P}_{(2,3)} \circ \mathcal{F}_1 \circ \mathcal{F}_4 \circ \mathcal{F}_3 \circ \mathcal{F}_4 \circ \mathcal{F}_2 \circ \mathcal{F}_1)(T)\\ &= (\mathcal{P}_{(1,2)} \circ \mathcal{P}_{(2,3)} \circ \mathcal{P}_{(3,4)} \circ \mathcal{F}_1 \circ \mathcal{F}_4 \circ \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1)(T)\\ &\simeq (\mathcal{F}_1 \circ \mathcal{F}_4 \circ \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1)(T)\end{aligned}$$

4.3.4.2 Preuve du théorème IV.1

Il existe plusieurs approches possibles pour démontrer le théorème IV.1, dont certaines ne font pas intervenir de bascules d'arêtes indexées. Cependant, nous avons choisi d'utiliser ces dernières, car cela nous permet de présenter d'autres résultats sur les bascules d'arêtes appliquées aux triangulations avec arêtes indexées.

Notre démonstration du théorème IV.1 nécessite l'introduction des trois lemmes suivants.

Lemme IV.1 : Soit T une triangulation d'un n-gone convexe. Si la séquence de bascules ϕ ne contient pas plus d'une occurrence de chaque bascule, alors elle est minimale. De plus, toutes les séquences minimales faiblement équivalentes à ϕ par rapport à T sont composées des mêmes bascules d'arêtes (modulo l'ordre au sein de la séquence).

Démonstration. : Le résultat est évident si ϕ est vide. Supposons que ϕ est de longueur non nulle et qu'elle ne contient pas plus d'une occurrence pour chaque bascule d'arête. Si ϕ n'est pas minimale, il existe une séquence ϕ_{min} faiblement équivalente à ϕ par rapport à T mais dont la longueur est strictement inférieure. Soit $\mathcal{F}_i$ une bascule présente dans la séquence ϕ mais pas dans la séquence ϕ_{min} ($\mathcal{F}_i$ existe nécessairement d'après la différence de longueur). Soient v_1 et v_2 les extrémités de l'arête i dans T (donc aussi dans $\phi_{min}(T)$). Soient v_3 et v_4 deux sommets incidents à l'arête i dans $\phi(T)$ (ce sont les deux sommets opposés incidents au support de

l'arête i et différents de v_1 et v_2). Dans la triangulation initiale T, l'arête i divise le n-gone en deux parties disjointes, dont l'une contient le sommet v_3 et le second le sommet v_4. Cela signifie que v_3 et v_4 sont séparés par l'arête i et qu'il est impossible de les relier sans basculer l'arête i. Cependant, v_3 et v_4 devraient être reliés dans $\phi_{min}(T)$, ce qui signifie que les triangulations $\phi(T)$ et $\phi_{min}(T)$ ont des connectivités différentes et contredit l'existence de ϕ_{min}. Donc ϕ est minimale.

Supposons maintenant une séquence minimale ϕ' telle que $\phi'(T) \simeq \phi(T)$, mais ϕ' n'est pas composée des mêmes bascules d'arêtes que ϕ. Soit $\mathcal{F}_i$ une bascule présente dans ϕ mais pas dans ϕ'. Avec un raisonnement analogue au précédent, on déduit que ϕ et ϕ' ne sont pas équivalentes par rapport à T, ce qui est contradictoire. □

Nous faisons remarquer que le lemme IV.1 n'est pas valide lorsque la triangulation possède au moins un sommet interne. La figure 4.9 illustre ce point, avec un contre-exemple d'une séquence non minimale possédant pourtant au plus une occurrence des arêtes basculées.

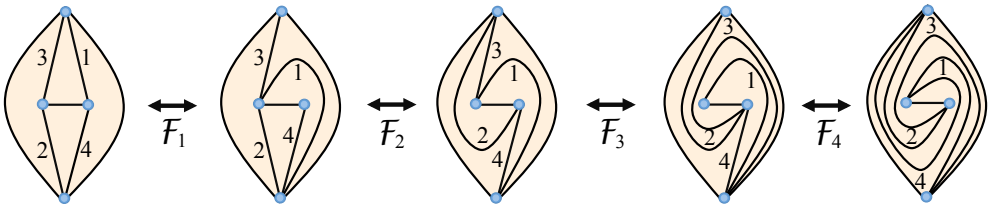


FIGURE 4.9 – Les triangulations situées aux extrémités gauche et droite sont composées des mêmes arêtes et des mêmes faces orientées mais les indices des arêtes sont différents. La séquence $(\mathcal{F}_4 \circ \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1)$ est donc faiblement équivalente à la séquence identité par rapport à la triangulation de droite. Cette séquence n'est pas minimale et possède au plus une occurrence pour chaque bascule d'arête.

Lemme IV.2 : Soit T une triangulation d'un n-gone convexe. Si une séquence minimale de bascules ϕ_1 ne contient pas plus d'une occurrence de chaque bascule, alors toutes les séquences minimales ϕ_2 faiblement équivalentes à ϕ_1 par rapport à T sont fortement équivalentes par commutativité.

Démonstration. : Le lemme IV.1 assure que ϕ_1 et ϕ_2 sont composées des mêmes bascules d'arêtes. On cherche maintenant à réordonner par commutativité chaque bascule de la séquence ϕ_2 en fonction de sa position dans ϕ_1. Si la première arête basculée dans la séquence ϕ_1 est $\mathcal{F}_i$, alors on déplace $\mathcal{F}_i$ au sein de ϕ_2 jusqu'à la première position, tant que $\mathcal{F}_i$ peut commuter avec les bascules d'arêtes qui la précèdent. Si $\mathcal{F}_i$ peut être déplacée jusqu'à la première place, on répète ce procédé pour la bascule d'arête situé à la position $p = 2$, et ainsi de suite. Cette opération continue jusqu'à ce que l'on rencontre une bascule d'arête $\mathcal{F}_j$ qui ne peut pas être déplacée à la position p par commutativité. On en déduit qu'il existe deux séquences μ_1 et ν avec $|\nu| = p - 1$, telles que $\phi_1(T) = (\mu_1 \circ \mathcal{F}_j \circ \nu)(T)$, et il existe μ_2 et μ_3 telles que $\phi_2(T) = (\mu_3 \circ \mathcal{F}_j \circ \mathcal{F}_k \circ \mu_2 \circ \nu)(T)$ sachant que $\mathcal{F}_j$ et $\mathcal{F}_k$ ne commutent pas

dans la triangulation $(\mu_2 \circ \nu)(T)$. Cela signifie que j et k sont incidents à une même face (*cf* figure 4.10a). Soient v_1 et v_2 les deux extrémités de k dans la triangulation $(\mu_2 \circ \nu)(T)$, et supposons k incident aux faces $(v_1v_2v_3)$ et $(v_2v_1v_4)$. Sans perte de généralité, on peut supposer que l'arête j est incidente aux sommets v_1 et v_3 ainsi qu'aux faces $(v_3v_1v_2)$ et $(v_1v_3v_5)$.

Donc l'arête k est incidente à v_3 et v_4 dans $(\mathcal{F}_j \circ \mathcal{F}_k \circ \mu_2 \circ \nu)(T)$, mais aussi dans $\phi_2(T)$ car ces arêtes ne seront pas rebasculées (*cf* figure 4.10a). On peut remarquer que j est déjà incident à v_1 et v_3 dans $\nu(T)$ car il n'existe pas d'occurrence de $\mathcal{F}_j$ dans la séquence $(\mu_2 \circ \nu)(T)$. On en déduit qu'après avoir effectué toutes les bascules de la séquence $(\mathcal{F}_j \circ \nu)$, l'arête j sépare le n-gone en deux parties. La première contient le sommet v_3 et la seconde contient à la fois les sommets v_1, v_4 et l'arête k (*cf* figure 4.10b). Les sommets v_3 et v_4 ne peuvent donc pas être reliés dans $\phi_1(T)$ car ils sont séparés par l'arête j déjà basculée, d'où la contradiction. □

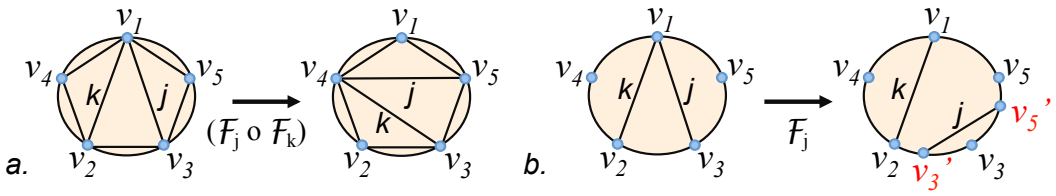


FIGURE 4.10 – a) Position de k et j dans $(\mathcal{F}_j \circ \mathcal{F}_k \circ \mu_2 \circ \nu)(T)$; b) évolution de j dans $(\mathcal{F}_j \circ \nu)(T)$. On peut remarquer que v_3' peut coïncider avec v_2 et que la position de v_5' n'est pas spécifiée.

Lemme IV.3 : Soient T_H une triangulation en *hélice* (c'est-à-dire que toutes les arêtes ont une extrémité commune v_i) d'un n-gone convexe et ϕ une séquence de bascules d'arêtes appliquée à T_H. Si ϕ est directement réduit par rapport à T_H alors ϕ est une séquence minimale. De plus, toutes les séquences minimales faiblement équivalentes à ϕ par rapport à T_H sont fortement équivalentes à ϕ par commutativité.

Démonstration. Par les lemmes IV.1 et IV.2, il suffit de démontrer que toute séquence directement réduite par rapport à T_H, possède au plus une occurrence de chaque bascule. Nous le faisons par induction sur le nombre n de sommets de la triangulation en hélice.

• Le résultat est trivial lorsque $n = 3$ ou $n = 4$. Pour $n = 5$, on remarque que les séquences directement réduites sont Id, $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_1 \circ \mathcal{F}_2$ et $\mathcal{F}_2 \circ \mathcal{F}_1$, et aucune d'elle ne contient plus d'une occurrence des bascules $\mathcal{F}_1$ et $\mathcal{F}_2$.

• Supposons que la propriété soit vérifiée pour un n-gone convexe, et montrons qu'elle est toujours vraie pour un $(n+1)$-gone convexe.

Soit ϕ une séquence directement réduite appliquée à une triangulation d'un $(n+1)$-gone convexe et triangulé en hélice T_H, et soit k avec $(0 < k < n-2)$ la première arête basculée dans ϕ. Il existe donc une séquence ϕ' telle que $\phi = (\phi' \circ \mathcal{F}_k)$ et ϕ' est directement réduite par rapport à $\mathcal{F}_k(T_H)$ car ϕ est supposée directement réduite par rapport à T_H.

Dans la triangulation $\mathcal{F}_k(T_H)$, l'arête k divise le $(n+1)$-gone en une face unique et un n-gone triangulé en hélice par rapport au sommet v_0 (*cf* figure 4.11).

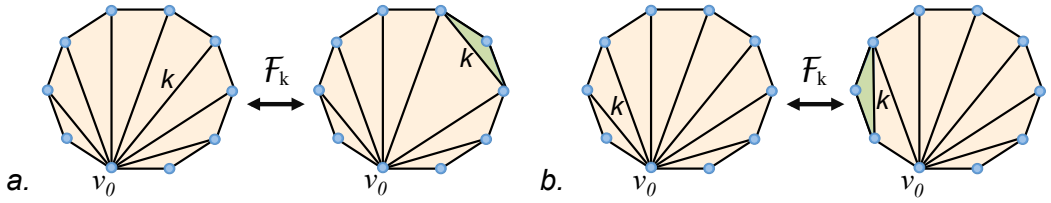


FIGURE 4.11 – Pour $n > 0$, si T_H est un $(n+1)$-gone convexe et triangulé en hélice, $\mathcal{F}_k(T_H)$ contient un n-gone convexe et triangulé en hélice.

Si ϕ' ne contient aucune occurrence de $\mathcal{F}_k$, alors la séquence ϕ' est appliquée au n-gone convexe et triangulé en hélice (privé de l'arête k), et par hypothèse de récurrence, ϕ' possède au plus une occurrence de chaque bascule. Donc $\phi = (\phi' \circ \mathcal{F}_k)$ possède une unique occurrence de chaque bascule.

Supposons que ϕ' contienne au moins une occurrence de $\mathcal{F}_k$. Soient μ et ν deux séquences telles que $\phi = (\nu \circ \mathcal{F}_k \circ \mu \circ \mathcal{F}_k)$ et μ ne contient aucune occurrence de $\mathcal{F}_k$. $\mathcal{F}_k$ ne commute pas avec μ car ϕ est directement réduite.

Il existe donc au moins une bascule $\mathcal{F}_{k'}$ telle que $\phi(T) = (\nu \circ \mathcal{F}_k \circ \mu_2 \circ \mathcal{F}_{k'} \circ \mu_1 \circ \mathcal{F}_k)(T)$ et $\mathcal{F}_{k'}$ est la première bascule qui ne commute pas avec $\mathcal{F}_k$, c'est-à-dire $(\mathcal{F}_{k'} \circ \mu_1 \circ \mathcal{F}_k)(T) = (\mathcal{F}_{k'} \circ \mathcal{F}_k \circ \mu_1)(T)$ et k est voisin de k' dans $\mu_1(T)$ $(|supp(k, \mu_1(T)) \bigcap supp(k', \mu_1(T))| > 1)$.

La séquence directement réduite $\mu = (\mu_2 \circ \mathcal{F}_{k'} \circ \mu_1)$ s'applique uniquement à la sous-triangulation en hélice de $\mathcal{F}_k(T)$ car elle ne possède aucune occurrence de $\mathcal{F}_k$. D'après l'hypothèse de récurrence, μ possède au plus une occurrence de chaque bascule d'arête.

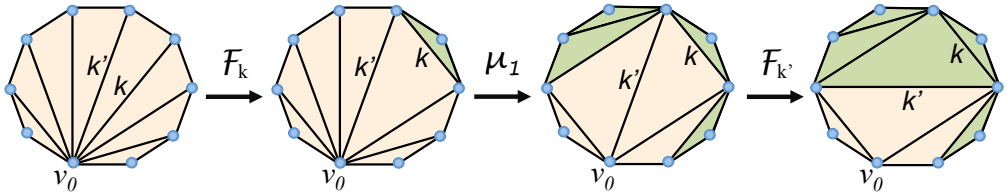


FIGURE 4.12 – k' divise la triangulation $(\mathcal{F}_{k'} \circ \mu_1 \circ \mathcal{F}_k)(T_H)$ en deux composantes disjointes : un p-gone convexe et triangulé en hélice, et une composante dans laquelle k est une arête interne.

k' divise donc $(\mathcal{F}_{k'} \circ \mu_1 \circ \mathcal{F}_k)(T_H) = (\mathcal{F}_{k'} \circ \mathcal{F}_k \circ \mu_1)(T_H)$ en deux composantes disjointes (*cf* figure 4.12). Soit T_{H_p}, la composante correspondant au n-gone convexe avec $p < n$ et triangulé en hélice, et T_k la seconde composante contenant l'arête interne k. On remarque que toutes les arêtes incluses dans T_k différentes de k ont déjà été basculées dans μ_1. Par conséquent, les arêtes basculées dans μ_2 sont localisées au sein de T_{H_p} et $\mathcal{F}_k$ commute donc avec μ_2 dans la triangulation $(\mathcal{F}_{k'} \circ \mathcal{F}_k \circ \mu_1)(T_H)$. On en déduit que $\phi(T) = (\nu \circ \mu_2 \circ \mathcal{F}_k \circ \mathcal{F}_{k'} \circ \mathcal{F}_k \circ \mu_1)(T_H)$ avec $|supp(k, \mu_1(T)) \bigcap supp(k', \mu_1(T))| = 1$, ce qui n'est pas possible car ϕ est une séquence directement réduite. ϕ' ne contient aucune occurrence de $\mathcal{F}_k$ et contient au plus une occurrence de chaque bascule. Comme $\phi = (\phi' \circ \mathcal{F}_k)$, la propriété est vérifiée. Les lemmes IV.1 et IV.2 terminent la démonstration. □

Nous pouvons à présent démontrer le théorème IV.1 :

Démonstration. : Soient ϕ_1 et ϕ_2 deux séquences de bascules telles que $\phi_1(T) \simeq \phi_2(T)$. Nous allons montrer qu'il est possible de générer une troisième séquence faiblement équivalente commune en utilisant uniquement les trois outils présentés.

Soient T_H un n-gone convexe et triangulé en hélice, et μ une séquence de bascules telle que $\mu(T) = T_H$. On a $\phi_1(T) = (\phi_1 \circ \mu^{-1} \circ \mu)(T)$ et $\phi_2(T) = (\phi_2 \circ \mu^{-1} \circ \mu)(T)$. Soient γ_1 et γ_2 deux séquences directement réduites obtenues à partir de $(\phi_1 \circ \mu^{-1})$ et $(\phi_2 \circ \mu^{-1})$ respectivement, par rapport à la triangulation $\mu(T)$. D'après le lemme IV.3, γ_1 et γ_2 sont deux séquences minimales fortement équivalentes par commutativité par rapport à T_H. On peut donc générer γ_1 à partir de γ_2 par l'outil de commutativité. On en déduit que la combinaison des trois outils permet de générer, à partir de ϕ_1 et ϕ_2, une séquence faiblement équivalente par rapport à T, $(\gamma_1 \circ \mu)$. □

4.3.5 Séquences équivalentes dans le cas général

Dans le cas général, le résultat précédent ne peut probablement pas être maintenu et nous ne pouvons pas assurer qu'une séquence minimale de bascules peut être atteinte en utilisant uniquement les trois outils présentés. En particulier, dans la figure 4.9, nous ne pensons pas que les trois outils permettent de générer une séquence faiblement équivalente à la séquence Id. Cependant, nous n'avons actuellement ni preuve ni contre-exemple.

4.4 Réduction de séquences de bascules d'arêtes

En s'appuyant sur les résultats précédents et sur les trois outils introduits dans la partie 4.3.2, nous présentons un algorithme générant une séquence réduite faiblement équivalente à une séquence originale donnée. Nous fournissons une heuristique alternative au problème NP-complet de détermination de séquences de longueur minimale entre deux triangulations, et nous montrons que nous pouvons l'utiliser pour

des configurations combinatoires et géométriques. La séquence obtenue est une séquence directement réduite, et correspond donc à un minimum local par rapport à une énergie minimisant la distance de bascules, mais pas nécessairement à un minimum global.

4.4.1 Algorithme de réduction

Soient une triangulation T et une séquence ϕ contenant b bascules, l'algorithme va tenter de retirer les bascules d'arêtes les unes après les autres tant que cela est possible.

Tentative de réduction de ϕ en déplaçant $\mathcal{F}_i$ au sein de ϕ :

Soit $\mathcal{F}_i$ une bacsule d'arête de ϕ. On cherche si ϕ contient une autre occurrence de $\mathcal{F}_i$ parmi les bascules suivantes dans la séquence. Cela correspond à une recherche parmi au plus $b-1$ éléments, avec une complexité en $O(b)$.

S'il n'existe pas d'autre occurrence parmi les bascules suivantes dans ϕ, on ne pourra pas supprimer $\mathcal{F}_i$. Sinon, on note $\mathcal{F}_i'$ la prochaine occurrence de $\mathcal{F}_i$ dans ϕ, et on essaye de les rapprocher au sein de la séquence : $\mathcal{F}_i$ est déplacée vers la gauche, (c'est-à-dire en direction des bascules de la séquence qui n'ont pas encore été appliquées), en direction de $\mathcal{F}_i'$, tant que $\mathcal{F}_i$ commute avec la bascule suivante au sein de la triangulation intermédiaire correspondante. De même, $\mathcal{F}_i'$ est déplacée vers la droite (c'est-à-dire en direction des bascules de la séquence qui ont déjà été appliquées), en direction de $\mathcal{F}_i$, tant que $\mathcal{F}_i$ commute avec la bascule précédente dans la triangulation intermédiaire correspondante (*cf* algorithme 10).

Algorithme 10 Réduction de $\mathcal{F}_i$ par déplacement de bascules au sein de Φ

```
Φ_Initial ← Φ
F_i' ← TrouverSuivante(Φ, F_i)
Φ ← DéplacementMaximalGauche(Φ, F_i)
Φ ← DéplacementMaximalDroite(Φ, F_i')
si TentativeRéduction(Φ) == FALSE alors
  Φ ← Φ_Initial
```

Plus formellement, soit ϕ_{cr} la séquence de bascules courante initialisée par ϕ. À chaque $\mathcal{F}_i$, on considère μ_1, μ_2 et μ_3 trois séquences telles que $\phi_{cr} = (\mu_3 \circ \mathcal{F}_i' \circ \mu_2 \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mu_1)$. Si $|supp(i, \mu_1(T)) \bigcap supp(j, \mu_1(T))| = 0$ et les arêtes peuvent être basculées, $\mathcal{F}_i$ est déplacée sur la gauche et ϕ_{cr} est désormais égale à $(\mu_3 \circ \mathcal{F}_i' \circ \mu_2 \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mu_1)$. Sinon, le déplacement est arrêté. La vérification de la disjonction des supports respectifs des arêtes i et j est effectuée par rapport à la triangulation $\mu_1(T)$, ce qui signifie que $\mu_1(T)$ doit être mise à jour en même temps que $\mathcal{F}_i$ est déplacée. On peut remarquer que chaque déplacement vers la gauche nécessite une bascule de la triangulation $\mu_1(T)$ afin de la mette à jour pour la prochaine vérification, et chaque déplacement d'une bascule vers la gauche s'effectue en temps constant. Le coût total du déplacement de $\mathcal{F}_i$ sur la gauche est en $\Theta(l)$ avec $l \leqslant b$.

Ensuite, on déplace de façon similaire $\mathcal{F}'_i$ vers la droite en direction de $\mathcal{F}_i$ tant que cela est possible et que les deux occurrences de $\mathcal{F}_i$ ne se croisent pas. Chaque déplacement vers la droite est vérifié et exécuté en temps constant, excepté pour le premier déplacement de $\mathcal{F}'_i$, qui s'effectue en $\Theta(k)$ où k représente le nombre de bascules entre les deux occurrences de $\mathcal{F}_i$, avec $k \leqslant b - l$ (car la mise à jour de la triangulation nécessite k bascules). Le coût total pour déplacer $\mathcal{F}'_i$ vers la droite est donc en $\Theta(k)$.

Lorsque les déplacements de $\mathcal{F}_i$ et $\mathcal{F}'_i$ sont terminés, trois cas peuvent être rencontrés :

- Si $\phi_{cr} = (\nu \circ \mathcal{F}'_i \circ \mathcal{F}_i \circ \mu)$, alors ϕ_{cr} est réduite à $(\nu \circ \mu)$ (les deux occurrences de $\mathcal{F}_i$ sont supprimées en temps constant).
- Si $\phi_{cr} = (\nu \circ \mathcal{F}'_i \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mu)$, $|supp(i, \mu(T)) \bigcap supp(j, \mu(T))| = 1$ et la séquence $(\mathcal{F}_i \circ \mathcal{F}_j)$ est basculable dans $\mu(T)$, alors ϕ_{cr} est réduite à $(P_{(i,j)}(\nu) \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \mu)$ (une occurrence de $\mathcal{F}_i$ est suprimée en temps $O(b)$, car la séquence ν dans laquelle les indices de certaines bascules sont transposés contient au plus b bascules d'arêtes) ;
- Si aucun des cas précédents n'est rencontré, les deux occurrences de $\mathcal{F}_i$ sont remises à leur position initiale.

Au final, le coût temporel d'une tentative de réduction d'une bascule $\mathcal{F}_i$ au sein de ϕ par réduction directe est $O(b)$, où b est le nombre de bascules d'arêtes présentes dans ϕ.

Algorithme de réduction de séquences de bascules d'arêtes :
L'algorithme itère sur toutes les bascules de ϕ, du début à la fin (c'est-à-dire de droite à gauche), jusqu'à ce qu'il trouve une bascule $\mathcal{F}_i$ pouvant diminuer ϕ par réduction directe. Le retrait d'une occurrence de $\mathcal{F}_i$ peut permettre à une bascule précédemment étudiée de devenir supprimable par réduction directe. Après chaque réduction de ϕ, l'algorithme recommence donc son itération depuis le début de la nouvelle séquence ϕ.

Après chaque réduction, l'algorithme itère parmi un sous-ensemble des $l \leq b$ arêtes restantes jusqu'à l'identification de la prochaine bascule directement réductible, si elle existe. La complexité de cette étape est en $O(l^2)$.

S'il n'existe aucune bascule pouvant mener à une réduction, l'algorithme itère parmi les $l \leq b$ arêtes de ϕ pour une dernière tentative de réduction directe de toutes les bascules restantes (complexité en $O(l^2)$).

La longueur de la séquence est strictement décroissante après chaque réduction directe, donc l'algorithme n'effectuera pas plus de b réductions. On en déduit que la complexité générale de l'algorithme est $O(b^3)$.

On peut remarquer que lorsque l'algorithme est appliqué à une séquence

presque optimale ou si les bascules ont une seule occurrence dans la séquence, l'algorithme procède en $O(b^2)$.

Il est possible de réduire encore plus la séquence en utilisant d'autres outils de réduction, eux-mêmes combinaisons des trois outils de la partie 4.3.2. Cependant, leur intégration est complexe car elle nécessite l'ajout de nombreuses conditions à vérifier avant chaque simplification, ce qui augmente considérablement la complexité de l'algorithme.

4.4.2 Résultats expérimentaux

Les résultats expérimentaux du tableau 4.1 ont été obtenus sur un PC avec un processeur Intel Core 2 Duo 2 GHz et 4 Go de RAM. Nous avons travaillé avec trois classes de triangulations différentes : le n-gone convexe, la classe des triangulations géométriques orientées et la classe des triangulations orientées combinatoirement *manifold*. Les séquences ont été générées aléatoirement de sorte que chaque bascule soit valide au sein des séquences et qu'il existe au moins une arête voisine basculée entre deux occurrences d'une même bascule d'arête. Les séquences sont caractérisées par leur longueur b ainsi que leur *redondance* r qui correspond au ratio entre la longueur de la séquence et le nombre d'arêtes différentes basculées dans la séquence. En particulier, si la séquence possède au plus une occurrence de chaque bascule, r est égale à 1. Par ailleurs, $b(1 - 1/r)$ donne une estimation du nombre de bascules pouvant être retirées par l'algorithme.

On peut remarquer que la complexité observée de l'algorithme est directement reliée à la redondance r : linéaire par rapport à la longueur de la séquence initiale lorsque r est proche de 1, elle devient quadratique lorsque r augmente. De même, le gain dépend essentiellement de la classe de triangulations et de la redondance r, plutôt que de la longueur de la séquence. En pratique, la complexité globale de l'algorithme semble être $O(b^{3-2/r})$. En particulier, il est instructif d'observer que lorsqu'on applique l'algorithme dans la classe des triangulations orientées combinatoirement *manifold* et à une séquence de bascules ϕ initialement générée dans une configuration géométrique, la séquence obtenue est plus courte que celle produite par l'algorithme appliqué à ϕ dans la classe des triangulations géométriques. Cela s'explique par le fait qu'il existe davantage de triangulations intermédiaires possibles dans le premier cas.

Nous ne connaissons pas d'autres algorithmes de réduction de séquence auxquels comparer nos résultats. Cependant, on peut remarquer que la longueur des séquences obtenues dans le cas d'un n-gone convexe est inférieure à $2n - 10$ avec $n > 13$, et donc inférieure à la borne supérieure établie par Pournin [Pournin 2012].

	n-gone convexe			Triangulations géométriques orientées			Triangulations orientées combinatoirement *manifold*			
Maillage Nombre d'arêtes	2 000	10 000	30 000	2 000	10 000	30 000	2 000	10 000	30 000	100 000
Longueur Initiale (r=1,1)	200	500	2 000	200	500	2 000	200	500	2 000	10 000
Gain (%)	27 (13)	64 (12)	216 (10)	2 (1)	7 (1)	70 (3)	8 (4)	31 (6)	157 (7)	1 043 (10)
Durée	1ms	13ms	76ms	1ms	4ms	31ms	5ms	24ms	541ms	2s
Longueur Initiale (r=2)	2 000	6 000	10 000	2 000	6 000	10 000	2 000	6 000	10 000	20 000
Gain (%)	973 (48)	2 950 (49)	4 964 (49)	596 (29)	1 062 (17)	2698 (26)	854 (42)	2 288 (38)	4 608 (46)	9 152 (45)
Durée	276ms	7s	1min22s	101ms	588ms	1s	3s	57s	1m28s	5min36s
Longueur Initiale (r=10)	6 000	10 000	20 000	6 000	10 000	20 000	6 000	10 000	20 000	30 000
Gain (%)	5 277 (87)	8 666 (86)	17 759 (88)	4 212 (70)	5 686 (57)	10 353 (51)	4 617 (76)	5 804 (58)	15 060 (75)	22 530 (75)
Durée	14s	1min15s	3min53s	281ms	946ms	3s	8s	5min52s	20min14s	1h32min

TABLE 4.1 – Résultats expérimentaux : dans la classe des triangulations géométriques et celle des triangulations orientées combinatoirement *manifold*, nous avons utilisé les même séquences, en nous assurant que toutes les bascules satisfont les contraintes de chaque configuration. Cela permet de mettre en valeur les conséquences du choix de la classe sur les résultats obtenus. La dernière colonne correspond à des séquences générées et réduites dans le classe des triangulations orientées combinatoirement *manifold*.

Enfin, on constate qu'avant et après réduction, l'évolution de la triangulation est assez similaire (*cf* figure 4.13). En effet, les outils de réduction sont locaux et la position des bascules au sein de la séquence est peu modifiée.

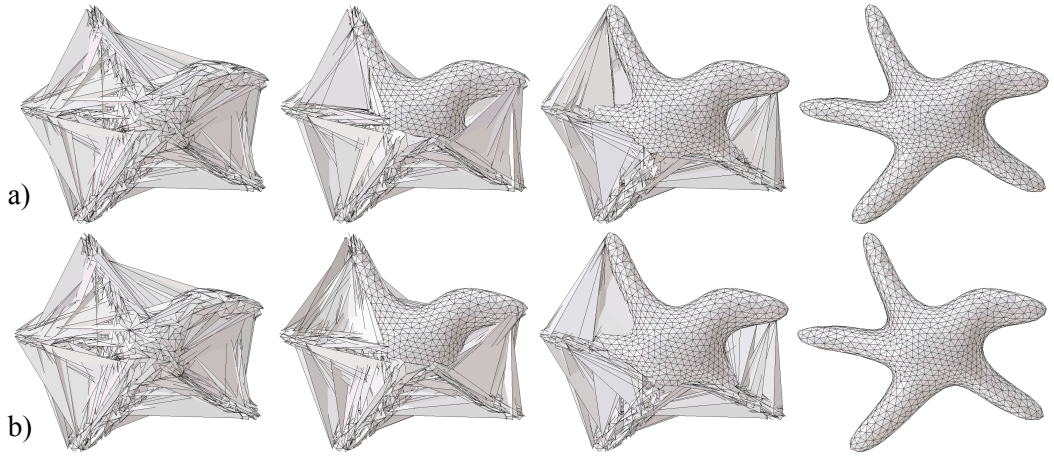


FIGURE 4.13 – a) Évolution d'une séquence composée de 40 000 bascules d'arêtes avec une redondance $r = 1,5$ (de type combinatoire) ; b) La séquence réduite composée de 29 942 bascules. Pour chacune des deux séquences, la première figure illustre la triangulation initiale, la triangulation où 1/3 des bascules ont été appliquées, puis 2/3 des bascules, et enfin la triangulation finale.

4.5 Discussion

Nous avons développé des outils de génération de séquences équivalentes basés sur l'étude des déplacements d'une arête au sein d'une triangulation. Malgré leur faible nombre, nous avons montré que dans le cas des triangulations d'un n-gone convexe, quelque soit la séquence de bascules donnée, ils permettent de générer toutes les séquences de bascules faiblement équivalentes. Nous en avons déduit un algorithme efficace de réduction, minimisant une séquence donnée jusqu'à atteindre un optimum local.

En ce qui concerne les perspectives, nous pouvons chercher à améliorer l'algorithme de réduction afin de générer des séquences faiblement équivalentes plus courtes, sans augmenter la complexité temporelle.

De plus, il serait intéressant de généraliser la bascule d'arête afin de coder une modification de connectivité de maillages dont les faces ne sont pas forcément triangulaires. Il est alors nécessaire de plonger le maillage dans une triangulation en remplaçant chaque face polygonale par un n-gone simple triangulé. Ce plongement peut être généré de façon automatique et déterministe à partir de la géométrie. On distinguerait ensuite les indices concernant les arêtes ajoutées dans le cadre du plon-

gement, de celles appartenant au maillage d'origine. Si deux maillages possèdent le même nombre de sommets et d'arêtes, ainsi que le même genre topologique, on peut coder le passage du premier maillage vers le second car cela équivaut à la détermination d'une séquence de bascules d'arêtes entre deux triangulations possédant des arêtes indexées (*cf* figure 4.14).

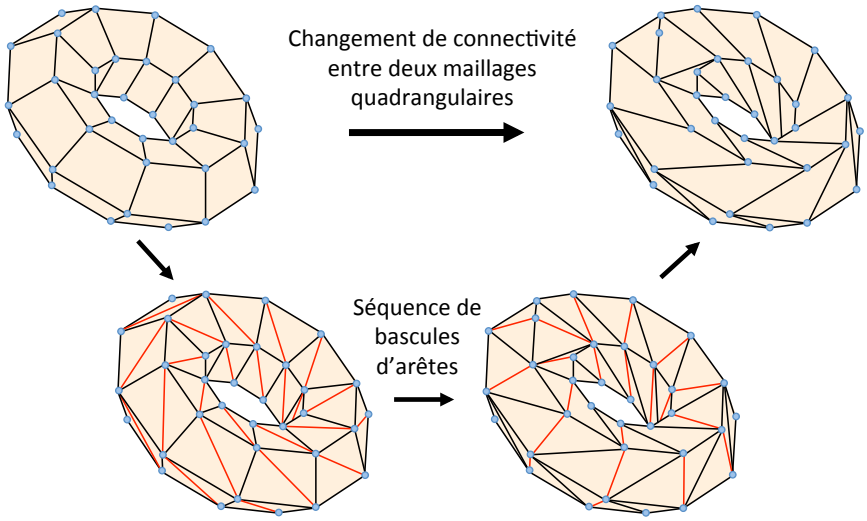


FIGURE 4.14 – Le codage d'une modification de connectivité entre deux maillages quadrangulaires peut être défini à partir de bascules d'arêtes indexées. Dans un premier temps, on plonge les maillages de départ et d'arrivé dans une triangulation en ajoutant de nouvelles arêtes (en rouge). Ensuite, on détermine une séquence de bascules d'arêtes entre les deux triangulations possédant des arêtes indexées.

Chapitre 5

Codage compact d'une séquence de bascules d'arêtes

Dans ce chapitre, nous revenons sur notre préoccupation initiale de compression de la connectivité de surfaces triangulées, en développant une transformation compacte de leur connectivité. Etant données deux triangulations T_1 et T_2 possédant un même nombre de sommets et un même genre topologique, une transformation de T_1 vers T_2 est compacte, si le coût du codage de la connectivité de T_2 à partir de la connectivité de T_1, est inférieur au codage de T_2 par un algorithme mono-résolution. En particulier, si le coût de la transformation de T_1 à T_2 est supérieur au seuil standart $3,2s$ bits avec s le nombre de sommets de T_2, alors il est plus intéressant de coder T_2 indépendamment. Cette constatation nous sert ensuite de repère pour élaborer une méthode de modification de la connectivité à faible coût mémoire.

En exploitant les résultats des chapitres 3 et 4, nous sommes capables de générer une séquence de bascules d'arêtes de taille raisonnable entre deux triangulations orientées combinatoirement *manifold*. Dans cette partie, nous présentons différentes méthodes pour coder cette séquence de bascules d'arêtes et offrir une transformation potentiellement compacte de connectivité :

- Nous présentons différentes méthodes de codage d'une séquence de bascules d'arêtes avec pour objectif de générer une séquence de bits la plus petite possible. Etant données deux triangulations T_1 et T_2 et une séquence de bascules transformant T_1 en T_2, le principe est basé sur une réorganisation de la séquence en n sous-ensembles tels que :

  1. les bascules d'un sous-ensemble sont exécutées après les bascules des sous-ensembles précédents ;

  2. les bascules d'un sous-ensemble commutent deux à deux dans la triangulation courante ;

  3. chaque arête basculée dans le sous-ensemble $k+1$ est voisine d'au moins une arête basculée dans le sous-ensemble k par rapport à la triangulation courante ;

  4. l'action sur T_1 des bascules présentes dans les n sous-ensembles génère T_2.

  Le codage d'un sous-ensemble nécessite a bits si la triangulation possède a arêtes. Cependant, certains bits peuvent être inférés à partir des précédents, et nous exploitons cette propriété pour générer des séquences de bits réduites.
- Nous proposons ensuite une transformation compacte déterminée par les

étapes suivantes : nous commençons par générer une séquence de bascules d'arêtes, nous la réduisons, nous la codons en une séquence de bits, et nous compressons le résultat avec d'un codeur arithmétique. Des résultats expérimentaux montrent que dans certains cas, notre transformation peut potentiellement concurrencer les algorithmes de compression mono-résolution. Nous l'utiliserons pour élaborer une nouvelle version de l'algorithme de compression multi-résolution *IPR* [Valette 2009].

– Enfin, nous associons la transformation compacte avec de nouveaux outils de modification de connectivité, pour coder la différence de connectivité entre deux triangulations ne possédant pas forcément le même nombre de sommets et le même genre topologique.

5.1 État de l'art sur le codage de la connectivité d'un maillage surfacique

Nous présentons un état de l'art sur les algorithmes de compression mono-résolution. Cette présentation donne une idée du coût de codage d'une connectivité à partir de ces algorithmes, et permet de déterminer quand est-ce que le coût d'une transformation peut-être dite compacte. Cet état de l'art présente aussi les différentes méthodes utilisées pour compresser, pouvant servir de point de départ à des améliorations de notre transformation de connectivité.

La description des méthodes de codage de la connectivité n'est pas exhaustive, et le lecteur peut se référer aux états de l'art de Gandoin [Gandoin 2001], d'Alliez et Gotsman [Alliez 2005], ou de Peng *et al.* [Peng 2005] pour obtenir plus de précisions.

5.1.1 Enumération et entropie de triangulations

Pour comparer les méthodes de codage de la connectivité d'une triangulation donnée, nous utilisons le nombre de bits nécessaires pour retrouver la connectivité totale, divisé par le nombre de sommets, ou d'arêtes. Nous obtenons un résultat traduisant le coût de codage d'une connectivité en *bits/sommet* ou *bits/arête*.

Nous utilisons aussi la notion d'*entropie* au sens de Shannon [Shannon 1951], qui désigne le nombre minimal de bits par symbole requis pour coder une séquence sans perte et fournit une limite théorique du volume de données nécessaires à ce codage. Cette valeur dépend à la fois du nombre de symboles distincts N et de la probabilité p_i d'occurrence pour chaque symbole s_i dans la séquence :

$$Entropie = -\sum_{i=1}^{N} p_i log_2(p_i)$$

Nous utiliserons la probabilité uniforme, c'est-à-dire $p_i = \frac{N_i}{\sum_{j=1}^{N} N_j}$ avec N_i le nombre d'occurrences du symbole s_i.

Si tous les éléments sont équiprobables, c'est-à-dire $p_i = \frac{1}{N}$, alors $Entropie = -\sum_{i=1}^{N} \frac{1}{N} log_2(\frac{1}{N}) = log_2(N)$ bits. L'entropie se détermine uniquement à partir du nombre de triangulations différentes pour un nuage de points donné, et correspond au nombre de bits nécessaires pour les énumérer. C'est la méthode utilisée par Tutte dans le lemme V.1. Nous rappelons qu'une triangulation est 3-connexe, s'il faut supprimer 3 sommets pour qu'elle ne soit plus connexe, et enracinée si elle possède une arête orientée et une face infinie à sa droite.

Lemme V.1 [Tutte 1962] : Le nombre de triangulations planaires 3-connexes enracinées ayant $s+2$ sommets est égal à $N_{Tr} = \frac{2(4s-3)!}{(3s-1)!s!}$.

Pour n assez grand, cette valeur vaut asymptotiquement $\frac{16}{27}\sqrt{\frac{3}{2\pi}} s^{-5/2} (\frac{256}{27})^s$ et on déduit que l'entropie des triangulations planaires vaut $\frac{1}{s} log_2(N_{Tr}) \approx log_2(\frac{256}{27}) \approx$ 3.2451 bits/sommet.

Ce résultat donne des repères pour juger de la performance d'une méthode de codage de connectivité.

5.1.2 Compression de données standard

La compression d'un maillage ne peut pas se contenter de l'utilisation d'algorithmes de compression de données standard. Même si ces méthodes permettent de réduire la redondance présente dans un ensemble de données informatiques arbitraires, elles ne tiennent pas compte de la structure particulière d'un maillage et des relations entre ses primitives. Cependant, elles sont parfois utilisées par les algorithmes de compression mono-résolution, et nous présentons des méthodes de codage dites *entropiques* : *méthodes de substitutions* et *méthodes statistiques*.

5.1.2.1 Méthodes par substitution

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

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

5.1.2.2 Méthodes statistiques

Nous présentons ici deux méthodes statistiques : le codage de Huffman et le codage arithmétique.

Le codage de Huffman, qui date des années 50 [Huffman 1952], repose sur une analyse statistique préalable des données à compresser. A l'issue de cette analyse, un arbre est construit permettant d'attribuer à chaque symbole un code dont le nombre de bits est inversement proportionnel à sa probabilité d'apparition. Ainsi, aux symboles les plus fréquents sont affectés des codes plus courts, et les codes les plus longs sont attribués aux symboles rares. En outre, ces codes sont séparables, c'est-à-dire qu'un code donné ne peut pas être le préfixe d'un autre code. Bien sûr, pour permettre au décodeur de reconnaître les symboles, il est nécessaire de lui transmettre le dictionnaire obtenu après la phase d'analyse statistique des données.

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

Le codage arithmétique a été développé dans les années 80 [Rissanen 1979, Rissanen 1983, Witten 1987], et permet de coder un symbole en fonction de sa probabilité d'occurrence, sur un nombre de bits non nécessairement entier, ce qui constitue un avantage considérable sur la méthode de Huffman. Fondamentalement, le principe de la compression arithmétique est de coder une séquence de symboles par un unique nombre réel appartenant à l'intervalle $[0, 1[$. Pour chaque symbole rencontré, l'intervalle initial $[0, 1[$ est raffiné en fonction de la probabilité estimée du symbole, conduisant finalement à un petit intervalle dont n'importe quel nombre code l'ensemble de la séquence. Cette méthode permet de coder chaque symbole s de la séquence sur $log_2(\frac{1}{P}) + \varepsilon$ bits, où P est la probabilité estimée de s, et ε une quantité négligeable devant $log_2(\frac{1}{P})$. Comme pour le codage de Huffman, cette modélisation peut être statique ou dynamique, mais dans le cas d'une modélisation dynamique, la complexité du codage arithmétique est meilleure puisque aucun arbre n'est reconstruit après la mise à jour du modèle statique.

5.1.3 Codage direct de la connectivité

La manière la plus directe et la plus couramment utilisée pour coder la connectivité d'un maillage consiste à indexer ses s sommets par un nombre compris entre 0 et $s-1$ et à décrire chaque face composant la structure géométrique par un code identifiant la nature du polygone et une liste d'indices pour les points correspondants aux sommets du polygone (*cf* figure 5.1 pour un exemple de codage d'une triangulation). Lorsque le maillage est orienté, pour chaque face, les sommets sont énumérés dans le sens trigonométrique. Dans une triangulation, le coût du codage d'une face nécessite au moins $3.log_2(s)$ bits, et si elle ne possède aucun bord, le

nombre de faces est deux fois supérieur au nombre de sommets pour s assez grand. Le coût du codage de la connectivité est donc de $6s.log_2(s)$ bits, soit $6.log_2(s)$ bits/sommet. Il existe de nombreux formats décrivant les maillages de cette façon : *OFF*, *OBJ*, *VRML*, *etc.*

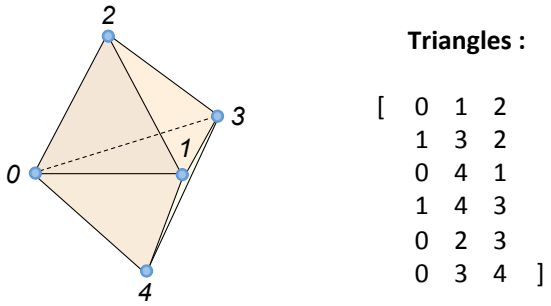


FIGURE 5.1 – Codage direct de la connectivité d'une triangulation.

Ces formats présentent l'avantage d'être rapidement accessibles et très généraux, mais ils sont redondants puisque chaque sommet est référencé autant de fois qu'il possède de faces incidentes. Le but est donc de proposer une manière directe d'accéder aux informations, au détriment du coût de son codage, ce qui n'est pas le cas des approches suivantes.

5.1.4 Codage optimal d'une triangulation 3-planaire

Il est possible de coder de façon optimale la classe des triangulations planaires 3-connexes enracinées ayant $s+2$ sommets. En effet, Poulalhon et Schaeffer [Poulalhon 2003] montrent qu'il existe une bijection entre cette structure et les *arbres bourgeonnants*. Lorsque ces arbres possèdent s noeuds, ils peuvent être codés par un mot binaire de longueur $4s-2$ et de poids $s-1$ (ce qui est équivalent à dire que la triangulation possède $s+2$ sommets), et à partir d'un codeur entropique tel que le codeur arithmétique, il est prouvé que l'on atteint la limite effective. Cependant, la longueur du code ne s'adapte pas à la régularité du maillage, et en pratique, un algorithme qui ne code pas de façon optimale la structure mais qui possède un coût beaucoup plus faible pour des maillages couramment manipulés sera plus intéressant.

5.1.5 Codage par parcours canonique des faces et des arêtes

C'est l'approche la plus couramment utilisée. Elle est basée sur la détermination d'un ordre d'énumération d'éléments de la connectivité (face, arête, ou sommet) jusqu'à l'obtention de l'intégralité du maillage.

Deering [Deering 1995] introduit en 1995 le concept de compression de maillage. L'objectif de l'auteur est d'obtenir une représentation des maillages triangulaires à la fois *compacte* et compatible avec une décompression et un affichage rapide.

L'algorithme utilise des *bandes de triangles généralisées* permettant de spécifier implicitement un triangle à chaque occurrence d'un sommet dans la séquence codante : le sommet courant engendre un nouveau triangle, connecté par la droite ou par la gauche au dernier triangle construit. Un bit additionnel est donc nécessaire pour préciser la position relative du nouveau triangle (*cf* figure 5.2). Puisqu'un sommet appartient à deux bandes, cette construction implique que chaque sommet du maillage apparait en moyenne deux fois dans la séquence. Le codage de la connectivité s'effectue en $7,5 + log_2(\frac{s}{8})$ bits/sommet.

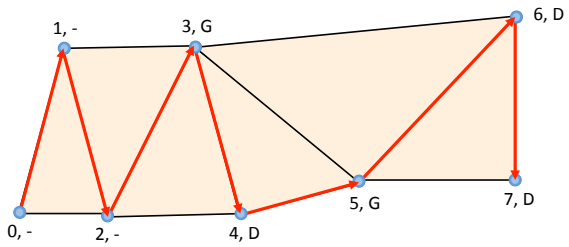


FIGURE 5.2 – Bande de triangles généralisée. Les flêches rouges définissent l'ordre des sommets découverts.

En 1998, Taubin et Rossignac [Taubin 1998] développent un algorithme de compression dans lequel la connectivité d'un maillage triangulaire est représentée par un codage explicite de deux arbres entrelacés. Le premier est un arbre de sommets, tel que le découpage du maillage suivant les arêtes de l'arbre engendre un polygone simple triangulé, qui peut être représenté par un second arbre dont les noeuds correspondent aux faces. Les deux arbres sont codés à l'aide d'un codeur arithmétique, et sur des maillages très réguliers, le coût de cette double représentation peut être inférieur à 2 bits/sommet. Cependant, il n'est pas borné dans le cas de connectivités plus complexes.

Rossignac propose une méthode appelée *Edgebreaker* [Rossignac 1999], conçue pour comprimer la connectivité de maillages triangulaires homéormorphes à une sphère, en réalisant une conquête des triangles par un parcours en profondeur. Chaque triangle donne lieu à une commande (ou étiquette) indiquant au décodeur comment le rattacher au reste du maillage. La figure 5.3 illustre les 5 commandes C, L, E, R, S utilisées. King et Rossignac [King 1999] proposent une extension de cette méthode permettant de traiter les surfaces possédant un ou plusieurs bords et un genre quelconque tout en garantissant un coût de codage maximal de 3.67 bits/sommet. Szymczak *et al.* [Szymczak 2001] ont également présenté une autre extension permettant de coder de manière plus efficace les maillages réguliers : cette méthode assure un débit maximal de 1.622 bits/sommet en cas de maillages larges et réguliers.

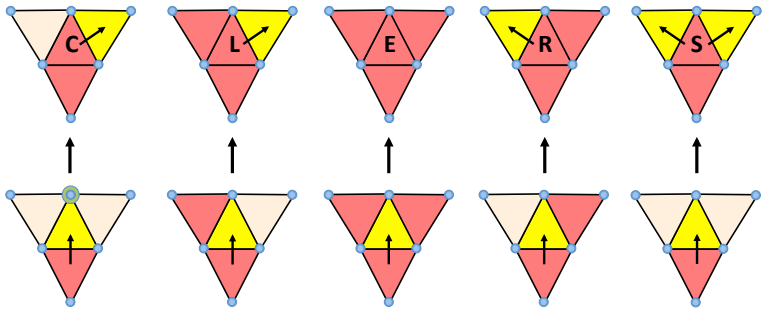


FIGURE 5.3 – Les 5 étiquettes de l'algorithme *Edgebreaker*. Le triangle courant est en jaune, les triangles déjà codés en rouge et les triangles non codés en beige ; le disque vert représente un point qui n'est pas encore conquis.

5.1.6 Codage par valence des sommets

Dans ce cas, on propose de coder la valence de chaque sommet (c'est-à-dire son nombre de voisins), plus un nombre limité de codes "d'accidents" pour retrouver la connectivité des maillages triangulaires. Dans de nombreux maillages, la distribution des valences étant faiblement dispersée, il en résulte des taux de compression extrêmement compétitifs.

Touma et Gotsman [Touma 1998] introduisent un algorithme basé sur la conquête de faces. Une face de départ est choisie arbitrairement, et sépare la surface en une région interne (constituée des éléments déjà traités) et une région externe. Ensuite, l'algorithme maintient une liste L composée des sommets adjacents aux arêtes de bord de la région interne. L est parcourue dans le sens direct, et chacun de ses sommets devient un pivot à tour de rôle. Le région interne s'agrandit par conquête de tous les triangles incidents au pivot qui ne sont pas codés (*cf* figure 5.4). Cette conquête détermine un ordre sur les sommets du maillage permettant de reconstruire sa connectivité avec peu d'informations supplémentaires : chacun des sommets de la séquence est accompagné de son degré et de l'une des trois commandes *ajouter*, *séparer* et *fusionner*. Ces commandes permettent de gérer toutes les modifications liées à L, c'est-à-dire le rebouchage, la séparation de listes et la fusion de celles-ci. L'efficacité de cet algorithme repose sur la distribution homogène des valences des sommets, qui est en général centrée sur 6. Le codage entropique des valences fournit un coût moyen de l'ordre de 2 bits/sommet sur des maillages usuels, et ce coût qui tend vers 0 pour des maillages très réguliers (c'est-à-dire que les sommets ont quasiment tous une valence de 6).

Une extension de l'algorithme de Touma et Gotsman est proposée par Alliez et Desbrun [Alliez 2001]. Les auteurs énoncent un ensemble de règles pour réduire le coût final en minimisant le nombre de séparations de liste. Par rapport à la méthode initiale, celle-ci présente un meilleur taux de compression, en particulier pour les maillages irréguliers. De plus, les auteurs démontrent que la borne supérieure du

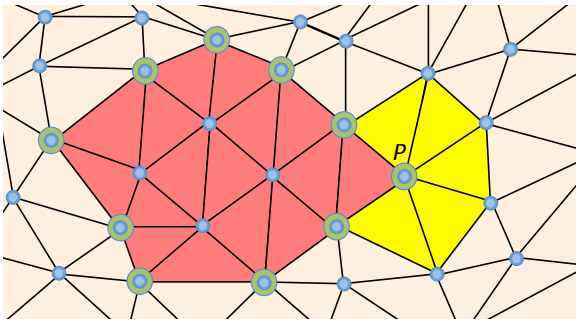


FIGURE 5.4 – Codage par conquête de triangles. En rouge, les triangles déjà codés ; en vert, les sommets qui deviendront des pivots ; en jaune, les triangles qui seront codés à partir du pivot P.

codage de la connectivité est de 3.24 bits/sommet, soit quasiment identique au coût optimal présenté par Tutte [Tutte 1962].

5.1.7 Codage dirigé par la géométrie

En pratique, il existe une certaine cohérence entre la géométrie des sommets et la connectivité du maillage. Les méthodes suivantes retrouvent la connectivité d'un maillage à partir de sa géométrie et d'un ensemble de symboles.

Lee *et al.* [Lee 2002] propose une méthode dirigée par la géométrie, permettant de traiter des maillages composés de triangles et de quadrangles. Ils revisitent la définition des cinq descripteurs du *Edgebreaker* et déterminent un ordre de parcours à partir de la géométrie, en choisissant la prochaine arête à manipuler afin que la région traitée soit la plus convexe possible. De cette manière le nombre de codes incidents est réduit, optimisant le codage de la connectivité. Cette approche permet d'obtenir des résultats en moyenne 40% inférieurs à la méthode pionnière de Touma et Gotsman [Touma 1998]. Kälberer *et al.* [Kälberer 2005] proposent aussi une méthode de codage par valence dont le parcours est déterminé par la géométrie. Leur méthode donne en moyenne des coûts de compression 20 à 30% inférieurs à celle de Lee *et al.*.

La géométrie peut également être utilisée pour guider un algorithme de reconstruction et prédire la connectivité. Cela nécessite que l'algorithme ait accès à la géométrie des sommets avant la construction de la connectivité. C'est le cas des travaux de Lewiner *et al.* [Lewiner 2005] utilisant l'algorithme de reconstruction par stratégie de balle pivotante [Bernardini 1999], et des travaux de Chaine *et al.* [Chaine 2007] utilisant l'algorithme de reconstruction par convection basé sur la tétraèdrisation de Delaunay [Chaine 2003]. Les méthodes sont utilisables pour une surface de genre topologique quelconque et comportant plusieurs composantes

connexes. En ce qui concerne les travaux de Chaine *et al.*, après l'utilisation d'un codeur arithmétique, le codage de la connectivité est proche de 0 lorsque le maillage est entièrement inclus dans une triangulation de Delaunay, et reste compétitif dans le cas général.

5.1.8 Définition de la notion de compacité pour une transformation

Etant données deux triangulation T_1 et T_2, nous dirons qu'une transformation de T_1 vers T_2 est compacte, si le coût du codage de la connectivité de T_2 à partir de la connectivité de T_1, peut concurrencer son codage par un algorithme de compression mono-résolution. Dans l'état de l'art, nous observons que pour une triangulation possédant s sommets, si le codage de la transformation est supérieur ou égale à $3,2s$ bits [Tutte 1962], il existe des méthodes pour lesquelles il est plus intéressant de coder indépendamment T_2 [Alliez 2001, Poulalhon 2003]. De plus, on remarque que si la triangulation est régulière ou incluse dans un Delaunay, la transformation peut-être compacte si le coût est inférieur à s bits, voir $\frac{s}{2}$ bits. Si la triangulation est moins régulière, les algorithmes de compression présentés génèrent des résultats plus coûteux et la transformation est potentiellement compacte lorsque le coût est inférieur ou égal à $2s$ bits.

Dans la suite de ce chapitre, nous construisons l'opération de transformation compacte de connectivité à partir des bascules d'arêtes et nous présentons une méthode de codage efficace des séquences.

5.2 Codage d'une séquence de bascules d'arêtes

Le but de cette partie est de présenter différentes méthodes de codage d'une séquence de bascules d'arêtes donnée. Nous travaillons avec des maillages pour lesquels les a arêtes sont indexées de 1 à a pour désigner les arêtes que l'on bascule. Si les arêtes ne sont pas indexées, il est possible de donner un indice par une méthode déterministe utilisant la géométrie ou l'indexation des sommets.

5.2.1 Codage simple

Cette première méthode consiste simplement à coder une séquence ordonnée de nombres entiers correspondant à l'arête à basculer (*cf* figure 5.5). Si la triangulation possède a arêtes, une bascule est désignée par un nombre compris entre 1 et a. Nous appliquons ensuite un codeur entropique sur la séquence.

Cette méthode est intéressante si la séquence possède très peu de bascules. Le codage d'une bascule nécessite $log_2(a)$ bits avant utilisation du codeur arithmétique.

5.2.2 Codage par bascules simultanées

Cette méthode est basée sur le concept de bascules simultanées introduit pas Hurtado [Hurtado 1998]. Etant données deux triangulations T_1 et T_2 et une séquence

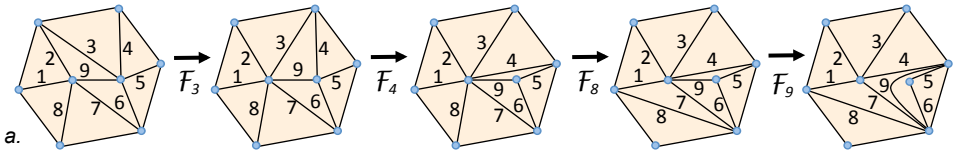


FIGURE 5.5 – Exemple de codage simple. En a) la séquence de bascules d'arêtes et en b) la séquence codée.

de bascules transformant T_1 en T_2, le principe est basé sur une réorganisation de la séquence en n sous-ensembles tels que :

- les bascules d'un sous-ensemble sont exécutées après les bascules des sous-ensembles précédents ;
- les bascules d'un sous-ensemble commutent deux à deux dans la triangulation courante ;
- chaque arête basculée dans le sous-ensemble $k+1$ est voisine d'au moins une arête basculée dans le sous-ensemble k par rapport à la triangulation courante ;
- l'action sur T_1 des bascules présentes dans les n sous-ensembles génère T_2.

Si la triangulation possède a arêtes, nous appelons *niveau* une séquence composée de a bits telle que si le i^{ime} bit vaut 1, alors on bascule l'arête indexée par i, sinon elle reste inchangée. Nous faisons remarquer que les bascules peuvent être effectuées dans n'importe quel ordre au sein d'un sous-ensemble, car elles commutent toutes entre elles.

De plus, nous ajoutons un bit d'en-tête supplémentaire au début de chaque niveau, de valeur 1 si le niveau qui le précéde possède au moins une bascule d'arête (autrement dit, s'il reste des bascules à coder). Ce bit vaut 0 dans le cas inverse et il n'est pas utile de coder le niveau en question et ses successeurs. Soient une séquence de bascules d'arêtes non vide et une triangulation courante initialisée par la triangulation de départ. On détermine la séquence de bits de la manière suivante : on met le bit d'en-tête du niveau 0 à 1 (car la séquence est non vide), puis on initialise le niveau en mettant à 0 tous les bits qui le composent.
On parcourt ensuite la séquence du début à la fin (sans effectuer de mise à jour de la triangulation courante) et pour chaque bascule appliquée à une arête i, on met le i^{me} bit à 1 et on retire la bascule de la séquence, si le i^{ime} bit est à 0 et si aucune des bascules précédemment rencontrées dans la sous-séquence est voisine de l'arête i. Une fois que l'on a terminé le parcours, on bascule les arêtes de la triangulation courante contenant un bit 1 dans le niveau 0 puis on recommence le parcours avec le niveau 1, et ainsi de suite jusqu'à ce que l'on obtienne un niveau composé exclusivement du bit 0. La concaténation de tous les niveaux donne la séquence de bits codant la séquence de bascules initiale (*cf* figure 5.6). Nous utilisons ensuite un codeur arithmétique pour compresser la séquence obtenue.

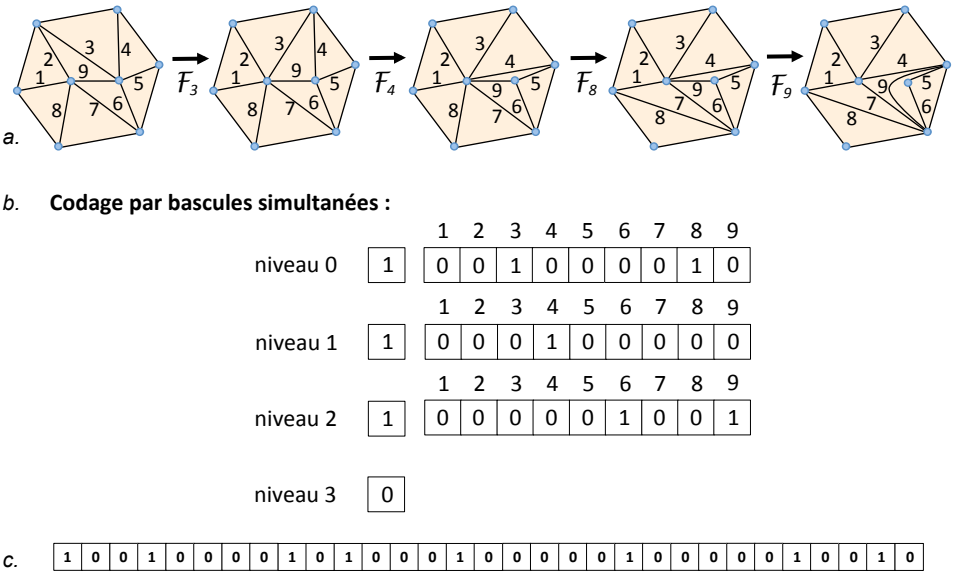


FIGURE 5.6 – Codage par bascules simultanées. En a) séquence de bascules d'arêtes, en b) l'ensemble des niveaux, en c) la séquence finale.

5.2.3 Codage par réduction de niveaux

Nous proposons une amélioration du codage par bascules d'arêtes simultanées en supprimant de la séquence de bits résultante, les bits qui peuvent être inférés à partir des précédents, car nous seront capables de les retrouver lors de la relecture. Nous présentons plusieurs cas de bits qui peuvent être inférés, en exploitant les particularités du codage par des bascules simultanées d'une séquence de bascules quelconque, puis d'une séquence de bascules réduite.

5.2.3.1 Codage appliqué à une séquence quelconque

Nous utilisons l'algorithme par bascules simultanées pour concevoir les niveaux précédents, puis nous réduisons les niveaux à partir des trois remarques suivantes :

- Dans un niveau, nous savons que si le bit correspondant à l'arête i vaut 1, alors au sein du même niveau, toutes les arêtes j voisines de i qui n'ont pas été traitées car $j > i$, ne seront pas affectées à 1 (car deux bascules d'un même niveau doivent commuter entre elles et ne peuvent donc pas être voisines). Nous pouvons donc retirer ces bits de la séquence.

- Nous considérons que nous travaillons avec une séquence composée de bascules autorisées sur les arêtes. On en déduit que si une arête n'est pas basculable par rapport à la classe de triangulation choisie, alors le bit correspondant ne vaudra pas 1. Nous retirons donc, sur chaque niveau, tous les bits codant la bascule d'une arête non basculable.

- Nous savons que toutes les bascules d'un sous-ensemble commutent entre elles et que chaque bascule i du sous-ensemble k est voisine d'au moins une bascule

de l'ensemble $k-1$. Donc, si une arête i, ainsi que toutes ses arêtes voisines, n'ont pas été basculées au niveau $k-1$, alors i ne sera pas basculée au niveau k. Nous retirons donc les bits correspondants à une arête non basculée avec ses arêtes voisines dans le niveau précédent.

La concaténation des niveaux réduits donne la séquence de bits codant la séquence de bascules initiale (*cf* figure 5.7) et nous utilisons ensuite un codeur arithmétique pour compresser la séquence obtenue.

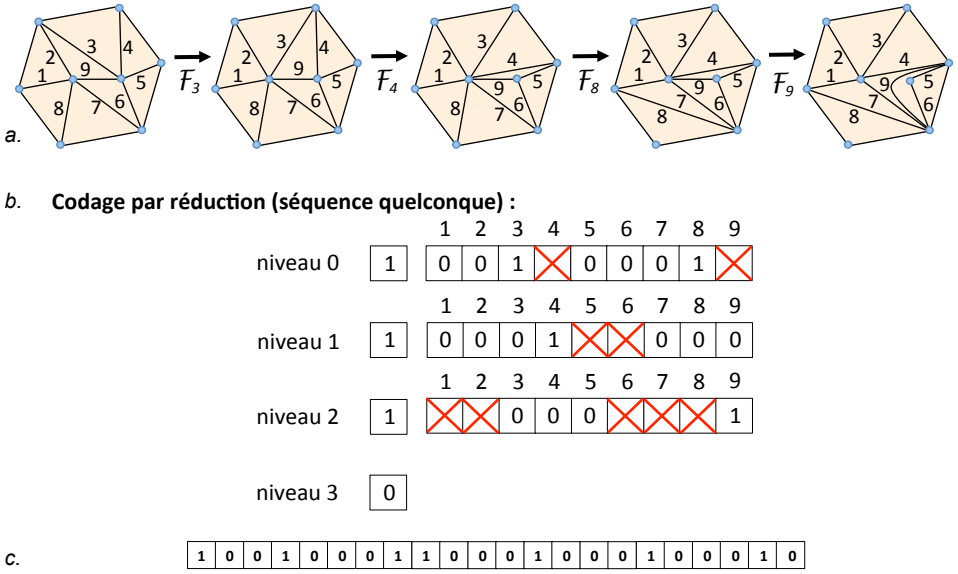


FIGURE 5.7 – Codage par réduction de niveaux appliqué à une séquence quelconque. En a) la séquence de bascules d'arêtes, en b) l'ensemble des niveaux et en c) la séquence finale.

Si les bascules d'arêtes sont localisées sur une petite zone de la triangulation, à partir du niveau 1, les bits restants correspondront uniquement aux arêtes situées dans la zone en question.

5.2.3.2 Codage appliqué à une séquence réduite

Lorsque la séquence de bascules d'arêtes a été réduite par l'algorithme présenté dans le chapitre 4, nous pouvons proposer une séquence de bits plus petite à partir des trois remarques suivantes :

- Nous savons qu'au sein d'une séquence de bascules d'arêtes réduite, il n'est pas possible de rapprocher par commutativité deux occurrences d'une même bascule. On en déduit que si le bit correspondant à l'arête i au niveau $k-1$ vaut 1, alors le bit correspondant à l'arête i au niveau k vaut forcément 0. Nous pouvons donc retirer ce bit au niveau k.

- Pour une triangulation T, nous savons aussi qu'une séquence réduite ne peut pas être équivalente par commutativité à $(\mu \circ \mathcal{F}_j \circ \mathcal{F}_i \circ \mathcal{F}_j \circ \nu)$, avec

$|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 1$, et $(\mathcal{F}_i \circ \mathcal{F}_i)$ est basculable dans $\nu(T)$. On en déduit que si le bit correspondant à l'arête i au niveau $k-2$ vaut 1, le bit correspondant à l'arête j au niveau k vaut 1, on a $|supp(i, \nu(T)) \bigcap supp(j, \nu(T))| = 1$ et $(\mathcal{F}_j \circ \mathcal{F}_i)$ est basculable dans la triangulation $\nu(T)$, alors le bit correspondant à l'arête i au niveau k sera toujours égal à 0. Nous pouvons donc retirer ce bit de la séquence au niveau k.

La concaténation des niveaux réduits génère une nouvelle séquence de bits (*cf* figure 5.8) et nous utilisons ensuite un codeur arithmétique pour compresser la séquence obtenue.

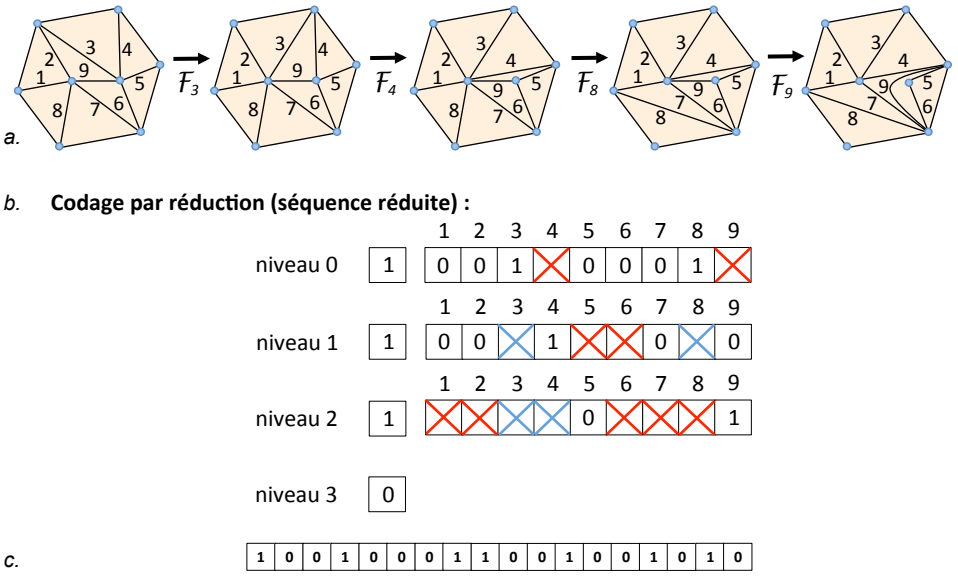


FIGURE 5.8 – Codage par réduction de niveaux appliqué à une séquence réduite. En a) la séquence de bascules d'arêtes, en b) l'ensemble des niveaux et en c) la séquence finale.

5.2.4 Résultats expérimentaux de la transformation compacte

Nous pouvons enfin présenter une transformation potentiellement compacte. Etant données deux triangulations orientées combinatoirement *manifold*, possédant un même nombre de sommets et un même genre topologique, nous déterminons une transformation par les étapes suivantes : nous commençons par générer une séquence de bascules d'arêtes, nous la réduisons, nous la codons en une séquence de bits et nous compressons le résultat avec le codeur arithmétique. Nous présentons au tableau 5.1, le coût d'une transformation en fonction de la méthode de codage d'une séquences de bascules d'arêtes et de la répartition des bascules d'arêtes sur une triangulation.

Pour un seuil tel que $3,2$ bits/s, il est difficile de définir l'ensemble des couples de triangulations générant une transformation dont le coût est inférieur à cette valeur.

	Codage simple	Codage par bascules	Codage par réduction (séquence quelconque)	Codage par réduction (séquence réduite)
Bascules diffuses	4,99 b/s	1,62 b/s	1,47 b/s	1,47 b/s
$n/a = 0,15$ et $r = 1$	1,67 b/a	0,54 b/a	0,49 b/a	0,49 b/a
Bascules localisées	3,94 b/s	2,64 b/s	1,06 b/s	1,00 b/s
$n/a = 0,15$ et $r = 1,2$	1,31 b/a	0,88 b/a	0,35 b/a	0,33 b/a
Bascules globales	12,6 b/s	4,32 b/s	3,49 b/s	3,39 b/s
$n/a = 0,3$ et $r = 1,03$	4,22 b/a	1,44 b/a	1,16 b/a	1,13 b/a
Bascules globales	36,4 b/s	12,1 b/s	7,84 b/s	7,12 b/s
$n/a = 1$ et $r = 1,25$	12,1 b/a	4,06 b/a	2,61 b/a	2,37 b/a

TABLE 5.1 – Résultats expérimentaux : coût d'une transformation compacte en fonction des méthodes de codage d'une séquence de bascules d'arêtes et de la répartition des bascules d'arêtes sur la triangulation (maillage : $s = 2432$, $a = 7290$ et $f = 4860$). *Bascules diffuses* signifie qu'il n'y a pas deux arêtes voisines basculées, *Bascules localisées* signifie que les bascules sont localisées sur une petite zone (1/5 des arêtes sont basculées) , et *Bascules globales* signifie que l'intégralité du maillage est basculé.

Cependant, les résultats expérimentaux montrent que la séquence de bascules ne doit être de longueur inférieure à $\frac{a}{3}$, avec a le nombre d'arêtes des triangulations. De plus, parmi l'ensemble des séquences de bascules de longueur k, les séquences possédant une faible redondance propose un codage plus intéressant que les autres. Le coût d'une transformation est donc fonction du nombre de bascules ainsi que du nombre d'arêtes basculées différentes au sein de la séquence de bascules d'arêtes.

5.3 Application à un algorithme de compression multi-résolution

Dans cette partie, nous nous servons de la transformation compacte précédente dans le but de proposer une nouvelle version plus efficace de l'algorithme *IPR* (pour *Incremental Parametric Refinement*) développé par Sébastien Valette *et al.* [Valette 2009]. Les auteurs présentent un algorithme de compression sans perte multi-résolution dans lequel le raffinement est dirigé par des critères géométriques, dans l'esprit des algorithmes de reconstruction de surfaces. Soient M_0, M_1, ..., M_n la séquence de maillages obtenue telle que M_0 est le maillage le moins raffiné, et M_n est le maillage souhaité (*cf* figure 5.9). Le genre topologique de ces maillages est identique. Le maillage M_{n-1} correspond à un maillage dans lequel l'ensemble de la géométrie du maillage M_n est entièrement restituée (les maillages ont donc même nombre de sommets). Mais leurs connectivités peuvent différer.

Le codage de la différence résiduelle entre le maillage M_{n-1} et M_n, est alors géré par une séquence de bascules d'arêtes. La séquence est déterminée par optimisation locale d'une énergie $\mathcal{E}n$ décrivant une distance entre les deux maillages. L'énergie $\mathcal{E}n$ correspond à le somme des distances dans M_{n-1} entre les couples de sommets adjacents dans M_n. La distance entre deux sommets M_{n-1} est caractérisée par la longueur du chemin de faces minimal qui les sépare. Le codage de la séquence de

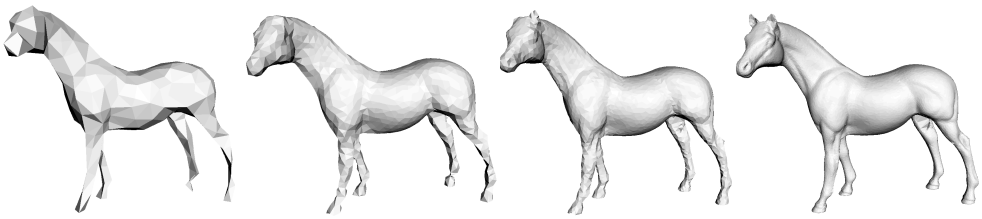


FIGURE 5.9 – Transmission progressive générée par l'algorithme *IPR*. Le premier maillage contient 295 sommets, le second 1540 sommets, le troisième 11 409 sommets et le dernier 19 851 sommets.

bascules est décrit par une séquence de bits : pour chaque arête du maillage, le bit correspondant désigne si elle doit être basculée ou non. La décision est prise en fonction de l'énergie, sachant qu'une arête est basculée lorsque cela engendre une diminution de $\mathcal{E}n$. Lorsque toutes les arêtes du maillage ont été passées en revue, on recommence ce processus jusqu'à ce que l'énergie $\mathcal{E}n$ ne puisse plus être décrémentée. Cependant, l'algorithme ne permet pas toujours d'atteindre le minimum global de cette énergie c'est-à-dire énergie nulle), ce qui signifie que dans certains cas où M_n et M_{n-1} sont trop différents, l'heuristique ne permet pas de déterminer une séquence de bascules d'arêtes entre M_{n-1} et M_n. Nos travaux permettent de remédier à ce problème et nous proposons une amélioration de l'algorithme *IPR*, dans laquelle nous utilisons notre transformation compacte pour coder la différence entre le maillage M_{n-1} et M_n. Nous comparons les résultats dans le tableau 5.2.

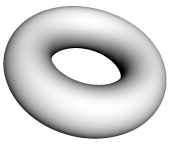
	fandisk	Torus	Rabbit	Fertility
				
Genre	0	1	0	4
Sommets	6 475	36 450	67 039	241 607
Arêtes	19 419	109 350	201 111	724 839
Coût du codage de la transformation (Méthode *IPR*)	1,30 b/s	1,56 b/s	1,06 b/s	2,82 b/s
Coût du codage de la transformation (Méthode compacte)	1,14 b/s	1,38 b/s	1,04 b/s	2,49 b/s

TABLE 5.2 – Résultats expérimentaux. Comparaison des méthodes de codage de la modification de connectivité entre les maillages M_{n-1} et M_n. La méthode *IPR* est la méthode abordée dans la publication de Valette *et al.*, et la méthode compacte est celle correspondant à la transformation compacte développée dans ce manuscrit.

La méthode de transformation compacte présentée dans ce manuscrit permet de toujours trouver une séquence de bascules d'arêtes entre deux triangulations possédant le même nombre de sommets et le même genre topologique. De plus, une analyse des résultats expérimentaux présentés dans la tableau 5.2, montre que le coût du codage de la transformation de connectivité est inférieur aux coûts obtenus par les auteurs dans la version originale d'*IPR*. Au final, nous obtenons un nouvel algorithme de compression multi-résolution plus efficace.

5.4 Vers une transformation compacte plus générale

Dans cette partie, nous montrons que les résultats présentés tout au long de ce mémoire peuvent se combiner avec d'autres opérations agissant sur la connectivité d'un maillage. Nous présentons une transformation entre deux triangulations qui n'ont pas forcément le même nombre de sommets, le même genre topologique et sans mise en correspondance entre les sommets.

Soient T_{init}, T_{cible} et $T_{courant}$ trois triangulations appartenant à $\mathbb{T}$. Nous initialisons $T_{courant}$ par T_{init} et nous considérons que les sommets des triangulations sont indexés de 1 à s et les arêtes de 1 et a.

5.4.1 Opérations supplémentaires

Nous présentons de nouvelles opérations agissant sur la connectivité.

5.4.1.1 Opération d'ajout et de retrait de sommets

L'ajout de sommet au sein d'une face consiste simplement à ajouter un sommet en désignant une face. L'opération de retrait de sommet remplace le polygone simple formé du sommet, des arêtes et des faces incidentes, par un polygone triangulé en hélice dont toutes les arêtes internes sont incidentes au sommet du polygone possédant le plus faible indice (*cf* figure 5.10). Si le sommet à retirer est de valence 2, nous basculons l'arête appartenant au *link* du sommet et possédant le plus petit indice, puis nous supprimons le sommet comme précédemment.

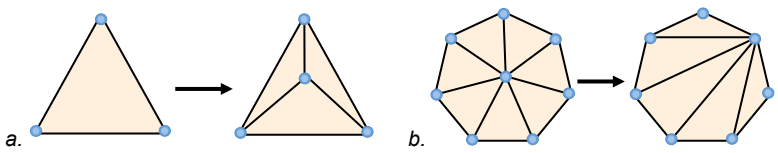


FIGURE 5.10 – En a) l'opération d'ajout de sommet, en b) l'opération de retrait de sommet.

Dans le cas des surfaces simpliciales, les opérations d'ajout et de retrait de sommets au sein d'une face, associées à la bascule d'arête, permettant de générer

toutes les triangulations homéomorphes [Pachner 1991]. Nous présentons une opération supplémentaire permettant de modifier le genre topologique d'une surface sans modifier le nombre de sommets.

5.4.1.2 Opération de changement de genre topologique

Pour incrémenter le genre topologique, nous recherchons deux faces ne possédant aucun sommet commun. Ensuite, nous supprimons ces faces et nous utilisons leurs arêtes de bord pour créer un *tunnel*, comme présenté sur la figure 5.11a. Cette opération retire 2 faces puis rajoute 6 faces et 6 arêtes. Le genre topologique est donc incrémenté de 1.

L'opération inverse va consister à trouver un tunnel, le retirer et remplacer les deux bords par des polygones simples en conservant une orientation consistente, comme présenté sur la figure 5.11b. Pour effectuer cette opération, nous recherchons un cycle de faces non contractile et non séparateur (notions définies dans l'annexe 2, partie B.2.2) pour lui faire jouer le rôle de tunnel. Ensuite, nous supprimons toutes les faces et arêtes de ce tunnel, ce qui engendre deux bords disjoints sur la triangulation que nous cousons avec les polygones correspondants. Si la surface est de genre 0, il n'existe pas de cycle non séparateur et non contractile.

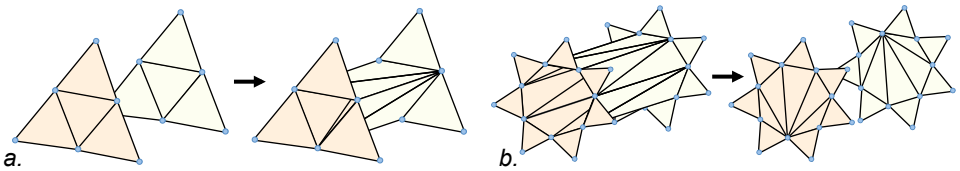


FIGURE 5.11 – En a) augmentation du genre topologique par création d'un tunnel ; en b) diminution du genre par suppression d'un tunnel.

5.4.2 Algorithme

L'algorithme proposé génère une transformation entre deux triangulations qui n'ont pas forcément le même nombre de sommets et le même genre topologique. Même si nous ne codons pas les modifications de la géométrie, les opérations utilisées ainsi que la mise en correspondance générée auraient une cohérence avec la géométrie des deux maillages. Dans la suite, nous appelons *distance combinatoire* entre deux primitives (faces, arêtes ou sommets), le nombre de faces composant le chemin simple le plus court reliant les deux éléments. Nous divisons l'algorithme en quatre étapes :

5.4.2.1 1^{re} étape : modification du genre topologique de $T_{courant}$:

– si le genre de T_{cible} est supérieur à celui de $T_{courant}$: on incrémente de 1 le genre de $T_{courant}$ jusqu'à ce qu'il soit égal à celui de T_{cible}. On recherche le

couple de faces ne possédant aucun sommet en commun et tel que le ratio entre la distance combinatoire et la distance euclidienne soit le plus grand possible. On construit ensuite un tunnel entre ces deux faces (*cf* figure 5.12).

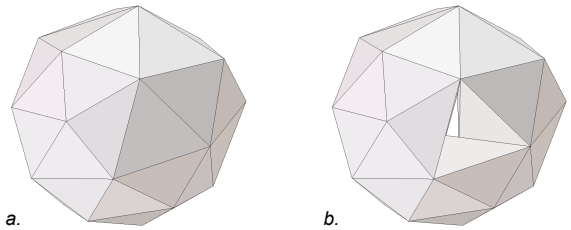


FIGURE 5.12 – Augmentation du genre de la surface par création d'un tunnel.

– si le genre de T_{cible} est inférieur à celui de $T_{courant}$: on décrémente de 1 le genre de $T_{courant}$ jusqu'à ce qu'il soit égal à celui de T_{cible}. On détermine un cycle non contractile traversant la face indexée par 1, et on le définit comme un tunnel pour réduire le genre de la surface.

Au final, le codage d'un nombre entier relatif désignant la différence de genre entre T_{cible} et $T_{courant}$ est suffisant pour changer le genre.

Remarque : parfois, il est nécessaire de rajouter des sommets afin de créer les tunnels nécessaires. Par exemple, il n'est pas possible d'augmenter le genre d'une surface composée d'un tétraèdre.

5.4.2.2 2^{me} étape : modification du nombre de sommets de $T_{courant}$:

– si le nombre de sommets de $T_{courant}$ est supérieur à celui de T_{cible}, on les supprime itérativement jusqu'à ce que $T_{courant}$ possède le même nombre de sommets que T_{cible}. À chaque itération, le sommet s_i supprimé correspond à celui qui minimise la valeur de $|2\pi - \sum \{angles\ \widehat{s_k s_i s_p}\}|$ avec s_k et s_i les sommets différents de s_i dans chacune des faces incidentes à s_i. Ce nombre approxime la valeur absolue de la courbure moyenne d'un maillage au niveau du sommet s_i et on décide d'assimiler localement les zones de faible courbure moyenne à un plan. Si plusieurs sommets minimisent cette valeur, on choisit le sommet possédant le plus petit indice dans le but de rendre l'opération reproductible (*cf* figure 5.13).
– si le nombre de sommets de $T_{courant}$ est inférieur à celui de T_{cible}, on ajoute les sommets itérativement jusqu'à ce que $T_{courant}$ possède le même nombre de sommets que T_{cible}. À chaque itération, le sommet est ajouté dans la face possédant la plus grande aire. Si plusieurs faces partagent ce maximum, on choisit la face incidente à l'arête possédant le plus petit indice (*cf* figure 5.14).

De même que pour le codage de la modification du genre topologique, un nombre entier relatif est suffisant pour que $T_{courant}$ possède le même nombre de sommets

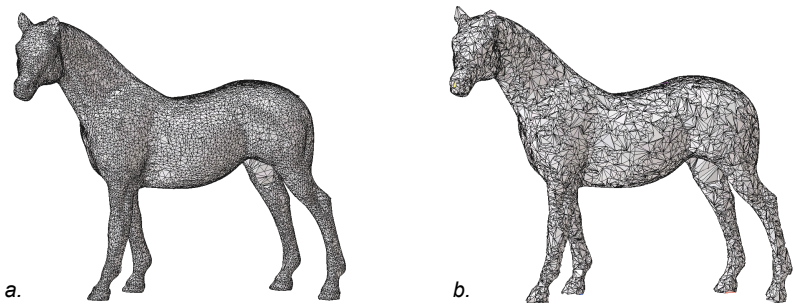


FIGURE 5.13 – Suppression de sommets : en a) 19 850 sommets et en b) 10 000 sommets.

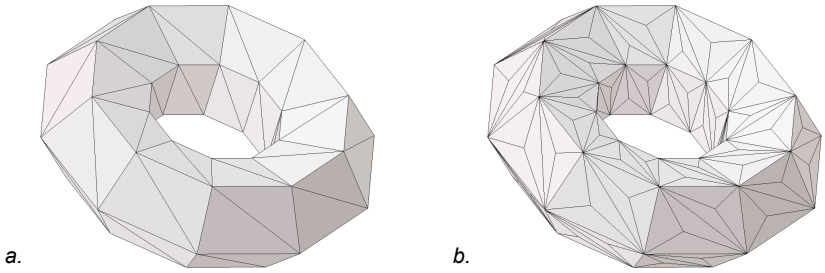


FIGURE 5.14 – Ajout de sommets : en a) 50 sommets et en b) 150 sommets.

que T_{cible}.

5.4.2.3 3^{me} étape : mise en correspondance de faces de départ :

Nous choisissons k faces au sein de chaque triangulation que nous mettons en correspondance. Elles serviront de points de départ à l'algorithme de détermination développé dans la partie 3.5.1.

Dans les deux triangulations, nous commençons par choisir une face au hasard, et nous recherchons la plus éloignée des faces par rapport à la distance combinatoire (elle n'est pas toujours unique). La face sélectionnée est notre première face pour la mise en correspondance. Nous obtenons une seconde face en sélectionnant la plus éloignée de la face précédente par rapport à la distance combinatoire. On détermine une troisième face en maximisant la distance combinatoire minimale avec les deux premières faces, et on continue ainsi jusqu'à obtenir k faces. Ensuite, on met en correspondance les faces sélectionnées ainsi que leurs sommets adjacents en minimisant la somme des distances euclidiennes par l'algorithme hongroise [Munkres 1957], qui s'effectue en temps polynomial. La mise en correspondance obtenue possède une certaine cohérence avec la géométrie.

5.4.2.4 4^{me} étape : mise en correspondance et détermination de la séquence de bascules d'arêtes :

Nous effectuons la mise en correspondance en parallèle de la détermination de bascules, dans le but d'obtenir la plus petite séquence possible. À partir des k faces de départ, nous utilisons l'algorithme de croissance de régions développé dans la partie 3.5, en modifiant le cas de la création d'une face vers un nouveau sommet (NS). En effet, supposons que l'on souhaite créer une face $(a'b'c')$ de T_{cible}, sachant que les sommets a et b de $T_{courant}$ sont déjà en correspondance avec les sommets a' et b', et qu'il n'existe aucune correspondance entre le sommet c' et un sommet de $T_{courant}$. Dans un premier temps, nous choisissons de mettre en correspondance le sommet c de $T_{courant}$ qui est le plus proche de l'arête (ab) (en terme de distance combinatoire) avec le sommet c', puis nous construisons (abc) dans $T_{courant}$. Nous obtenons un algorithme de détermination de séquences de bascules d'arêtes qui génère parallèlement une mise en correspondance des sommets (*cf* figure 5.15).

Nous obtenons une séquence de bascules que nous réduisons par l'algorithme présenté dans le chapitre 4, et nous codons la séquence de bascules d'arêtes en bascules de bits par les méthodes présentées dans le chapitre 5. Enfin, nous compressons le résultat à l'aide d'un codeur arithmétique.

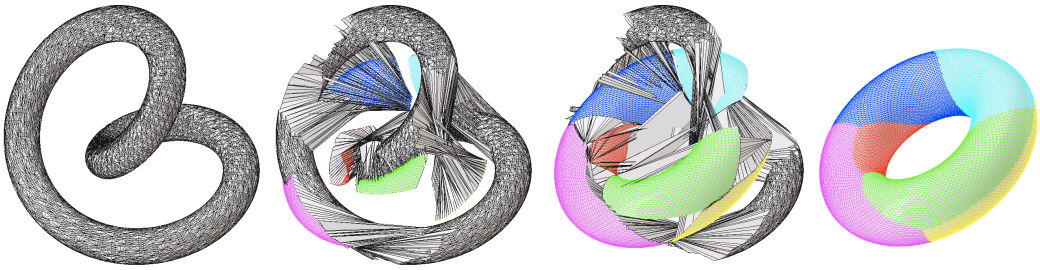


FIGURE 5.15 – Déroulement de la détermination d'une séquence de bascules d'arêtes. Lorsqu'un sommet de $T_{courant}$ est mis en correspondance avec un sommet de T_{cible}, nous lui attribuons les mêmes coordonnées. Les régions colorées correspondent aux k régions sur lesquelles nous effectuons les croissances.

Nous aurions aussi pu utiliser un algorithme de mise en correspondance entre les étapes 2 et 3, c'est-à-dire lorsque les deux triangulations possèdent le même nombre de sommets et le même genre topologique. Dans ce cas, nous ne faisons pas appel à l'étape 3, et l'étape 4 consiste uniquement à appliquer l'algorithme de la partie 3.5, sans effectuer de mise en correspondance.

Au final, la transformation est entièrement déterminée par une séquence de bascules d'arêtes et deux nombres entiers (le premier code le changement de genre topologique, le second l'ajout ou le retrait de sommets).

5.4.3 Résultats expérimentaux

Les résultats expérimentaux de la table 5.3 ont été obtenus sur un PC composé d'un Intel Core 2 Duo à 2 GHZ et 4 Go de RAM. Pour chaque test, nous avons choisi de mettre trois faces de départ en correspondance. Lorsqu'on observe le coût des transformations, on constate qu'il est plus avantageux de coder les deux maillages indépendamment. Si deux triangulations ont une connectivité "proche", les opérations de modification du nombre de sommets et du genre topologique, ainsi que le procédé de mise en correspondance ne favorise pas la conservation de cette proximité. En ce qui concerne le temps de calcul de l'algorithme, le résultat dépend principalement de l'algorithme de réduction de séquence. En effet, si l'on ne cherche pas à réduire la taille des séquences de bascules, les durées sont inférieures à quelques secondes dans chacun des cas présentés.

	Même genre topologique				**Genre topologique différent**			
	Sphere	Fandisk	Twist	Tore	Sphere	2-Tore	Tore	Fandisk
								
Nombre de sommets	21 812	6 476	837	1350	21 812	20 734	1 350	6 476
Séquence avant simplification Nombre de bascules (Redondance)	38 532 (3,2)		7 042 (2,9)		134 407 (5,0)		47 582 (3,4)	
Séquence après simplification Nombre de bascules (Redondance)	22 873 (1,9)		4 608 (1,9)		111 501 (4,1)		23 256 (1,6)	
Coût total	6,18 b/s		6,28 b/s		7,17 b/s		6,11 b/s	
Durée totale	3min 00s		118ms		20min 03s		4min 51s	

TABLE 5.3 – Résultats expérimentaux. Le coût total en bits de la transformation entre les deux triangulations est divisé par le nombre de sommets du maillage cible.

Nous sommes convaincus que l'on peut obtenir de meilleurs résultats en utilisant plus astucieusement l'ensemble des outils présentés, et en donnant un rôle plus important à la géométrie. L'utilisation de bascules automatiques, telles que des bascules de Delaunay, produirait un maillage dont la connectivité est cohérente avec la géométrie et réduiraient peut-être la taille de la séquence de bascules d'arêtes à coder. Nous pensons en particulier aux bascules d'arêtes appliquées aux triangulations de Bobenko [Bobenko 2007].

5.5 Discussion

Dans ce chapitre, nous avons présenté des méthodes de codage de séquences de bascules d'arêtes, et nous avons appliqué l'ensemble des résultats de ce mémoire pour développer une transformation potentiellement compacte entre deux triangulations orientées combinatoirement *manifold* possédant un même nombre de sommets et un même genre topologique. Les résultats expérimentaux montrent que le coût de la transformation dépend du nombre de bascules présentes dans la séquence de bascules ainsi que du nombre d'arêtes différentes basculées. Notre transformation a permis d'améliorer les résultats de l'algorithme de compression multi-résolution *IPR*. Nous avons également proposé une nouvelle transformation entre deux triangulations ne possédant pas le même nombre de sommets et le même genre topologique. Cependant, les résultats n'étant pas optimaux, de nombreuses améliorations sont à l'étude ouvrant ainsi une nouvelle voie de recherche.

En ce qui concerne les perspectives, il serait intéressant d'utiliser cette transformation pour coder un ensemble de triangulations. En effet, cet outil peut être une solution au stockage d'un grand nombre de maillages. Nos résultats pourraient également servir de point de départ à l'élaboration d'algorithmes de compression de séquences de maillages non contraintes (c'est-à-dire avec une connectivité évolutive).

Chapitre 6

Conclusion et perspectives

6.1 Résumé des contributions

Le fil conducteur de ce mémoire est le développement d'une transformation compacte entre deux connectivités à partir de bascules d'arêtes. Les travaux développés s'inscrivent dans la thématique de la compression de maillages. Nous nous sommes intéressés dans un premier temps, au problème de détermination d'une séquence de bascules d'arêtes entre deux triangulations partageant un même nombre de sommets et un même genre topologique. Nous avons introduit une nouvelle classe de triangulations, appelée classe des triangulations orientées combinatoirement *manifold*, et nous avons présenté les conditions nécessaires et suffisantes assurant l'existence d'une séquence de bascules d'arêtes pour un couple de triangulations appartenant à cette classe. Nous en avons déduit un algorithme direct de détermination de séquences, dont la validité est prouvée.

Dans un second temps, nous nous sommes intéressés au problème de minimisation des séquences de bascules d'arêtes. Nous avons cherché à mieux comprendre l'action d'une bascule en appliquant l'opération à des triangulations possédant des arêtes indexées, et nous avons mis en évidence qu'une séquence de bascules d'arêtes peut être utilisée pour transposer des indices. Nous avons déduit trois outils simples générant des séquences de bascules équivalentes et nous avons montré que, dans le cas des triangulations d'un n-gone convexe, ces trois outils permettent de passer d'une première séquence de bascules à une seconde si elles sont faiblement équivalentes. Nous avons ensuite développé un algorithme de réduction de séquences de bascules d'arêtes, dont le temps d'exécution dépend principalement de la longueur et de la redondance de la séquence.

Dans un dernier temps, nous avons proposé des approches de codage d'une séquence de bascules d'arêtes dans le but de générer la plus petite séquence de bits possible. Nous avons ensuite développé une transformation entre deux triangulations pouvant concurrencer les algorithmes de compression mono-résolution et dont le coût dépend du nombre de bascules ainsi que du nombre d'arêtes différentes basculées au sein de la séquence. Notre transformation a permis d'améliorer les résultats de l'algorithme de compression multi-résolution *IPR*. Nous avons également proposé une nouvelle transformation entre deux triangulations ne possédant pas forcément le même nombre de sommets et le même genre topologique.

6.2 Idées à explorer

Trois grandes perspectives peuvent découler de ce travail : une meilleure efficacité de la transformation, une transformation qui ne se limite pas à des couples de triangulations possédant le même nombre de sommets et le même genre topologique, et des applications à la compression de maillages

Nous proposons plusieurs pistes pour rendre la transformation plus compacte. La première perspective consiste à améliorer l'algorithme de détermination afin de générer des séquences de taille plus réduite. Nous pensons qu'une analyse des maillages, basée sur la détection des zones nécessitant beaucoup ou au contraire peu de bascules d'arêtes pour passer d'une connectivité à une autre, permettrait de choisir judicieusement la position de la face de départ, et d'orienter la croissance de régions pour minimiser le nombre de bascules d'arêtes nécessaires. L'algorithme de réduction de séquences nous semble quant à lui plus difficile à améliorer dans la mesure où une meilleure réduction de la taille des séquences s'accompagnerait nécessairement d'une augmentation sensible de la complexité algorithmique. Enfin, le codage d'une séquence de bascules d'arêtes semble aussi difficile à améliorer, mais une utilisation différente du codeur arithmétique en compressant la séquence de bits par paquets, permettrait peut-être d'obtenir de meilleurs résultats.

Il serait également intéressant de développer une transformation compacte entre deux triangulations qui ne possèdent pas forcément le même nombre de sommets et le même genre topologique. Nous avons proposé une telle transformation dans la partie 5.4, mais les coûts de codage obtenus ne permettent pas d'utiliser cette transformation pour compresser des maillages. Une meilleure efficacité nécessite probablement une meilleure utilisation des outils de modification de connectivité, une meilleure mise en correspondance entre les maillages, et peut-être l'utilisation de bascules automatiques telles que des bascules de Delaunay.

Nous suggérons également de généraliser notre transformation à des maillages qui ne sont pas forcément des triangulations. Nous avons montré que le fait d'appliquer une séquence de bascules d'arêtes sur des triangulations possédant des arêtes indexées, peut constituer le point de départ de cette généralisation.

Enfin, nous proposons d'utiliser cette transformation pour développer de nouveaux algorithmes de compression. En effet, nos résultats pourraient servir de point de départ à l'élaboration d'algorithmes de compression de séquences de maillages non contraintes (c'est-à-dire avec une connectivité évolutive). Il serait également intéressant d'utiliser cette transformation pour coder un ensemble de triangulations, et proposer une solution au stockage de très grand nombre de maillages.

Annexe A

Relation entre les rotations sur les arbres binaires et les bascules d'arêtes

Depuis les années 1980, plusieurs travaux ont émergé sur les bascules d'arêtes appliquées aux triangulations d'un n-gone convexe (dans ce cas toutes les arêtes sont basculables à chaque instant) afin de faciliter la compréhension des rotations appliquées aux arbres binaires [Sleator 1986, Sleator 1987]. Nous proposons cette annexe pour présenter le lien entre les bascules d'arêtes d'un n-gone convexe et les rotations.

A.1 Les arbres binaires

Un arbre binaire est une structure de données que l'on représente habituellement sous la forme d'un graphe connexe acyclique, tel que le degré de chaque noeud soit au maximum 3. Le noeud initial appelé *racine*, est au maximum de degré 2, et chaque noeud aura un unique *parent* (sauf pour le noeud racine) et au plus deux *enfants*. On appelle *feuille* un noeud ne possédant pas de fils, et *branche* l'arête reliant deux noeuds.

L'opération de *rotation d'un arbre binaire* est habituellement définie en choisissant un noeud Q qui représente la racine du sous-arbre dans lequel on effectue la rotation et une direction (droite ou gauche) pour orienter la rotation (figure A.1). Elle est en particulier utilisée sur des arbres binaires de recherche car elle permet de changer la structure sans invalider l'ordre des éléments.

Cependant, on peut aussi utiliser uniquement une branche pour désigner une rotation : parmi les deux extrémités de la branche, le noeud parent représente la racine du sous-arbre et si le second noeud est un fils gauche, on effectue une rotation à droite, sinon on effectue une rotation à gauche. On indexe donc les k branches de l'arbre par un entier appartenant à $\{1...k\}$ et différent pour chaque branche. De plus, on désigne par $\mathcal{R}ot_i$ l'opération de rotation au niveau de la branche i (figure A.2).

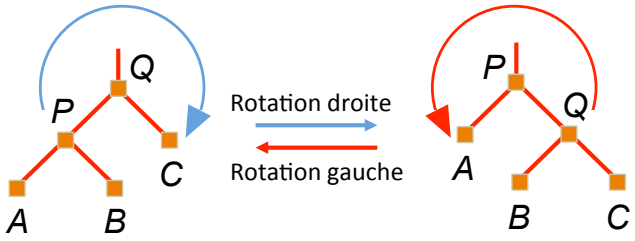


FIGURE A.1 – Rotation à droite au niveau du noeud Q et à gauche au niveau du noeud P.

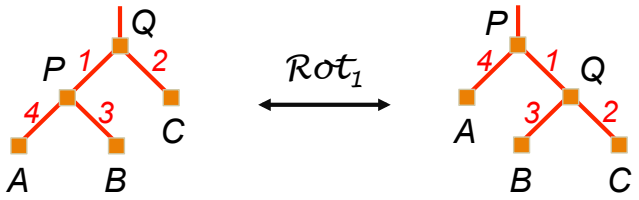


FIGURE A.2 – L'opération $\mathcal{R}ot_1$ est une involution.

A.2 Correspondance entre les rotations sur les arbres binaires et les bascules d'arêtes

Un arbre binaire contenant n noeuds possède une correspondance avec les triangulations d'un $(n+2)$-gone. Il existe plusieurs manières d'effectuer la mise en correspondance, et nous choisissons la suivante : chaque face de la triangulation représente un noeud de l'arbre, et deux noeuds sont reliés par une branche si les deux faces correspondantes sont voisines dans la triangulation. On choisit ensuite une arête du bord telle qu'avant et après chaque rotation, la face incidente à cette arête corresponde au noeud racine de l'arbre (figure A.3).

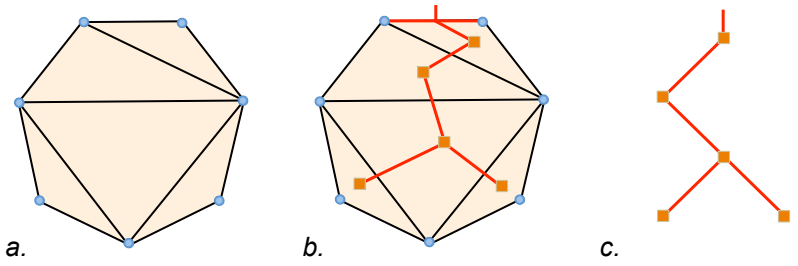


FIGURE A.3 – La triangulation d'un 7-gone en a) et l'arbre binaire correspondant en c) ; leur superposition en b).

Par la mise en correspondance énoncée, une rotation dans un arbre correspond exactement à une bascule d'arête dans la triangulation associée (figure A.4). En particulier, la détermination du nombre minimal de rotations entre deux arbres binaires est équivalente à celle du nombre minimal de bascules d'arêtes entre deux

triangulations d'un n-gone convexe. La question étudiant la NP-complétude du problème est toujours ouverte [Sleator 1987].

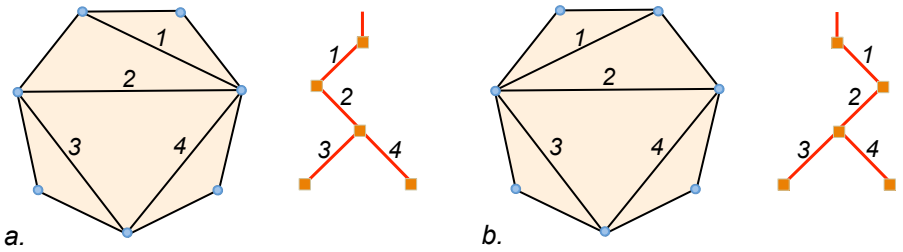


FIGURE A.4 – Equivalence entre bascule d'arête et rotation d'un noeud.

Par analogie avec les résultats traités dans le chapitre 4, nous mettons en évidence que si l'on indexe les branches, une séquence de rotations peut transposer les indices dans un arbre binaire (figure A.5).

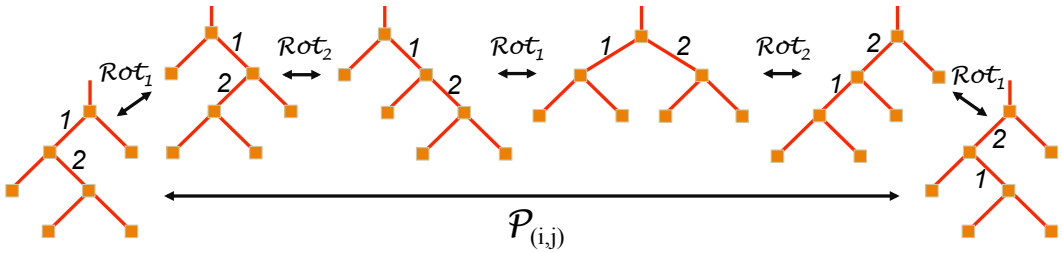


FIGURE A.5 – Une séquence de rotations engendre une transposition des indexes des branches.

Annexe B

Quelques notions sur les graphes et les triangulations surfaciques

B.1 Notions sur les graphes

Un *graphe* $G = (V, E)$ est la donnée d'un ensemble de *sommets* V et d'un ensemble d'*arêtes* E. Chaque arête possède deux sommets, appelés ses *extrémités*. Il existe de nombreuses variantes de la notion de graphe :

- le graphe peut être *orienté*, c'est-à-dire que chaque arête a un sommet initial et un sommet terminal, ou *non-orienté* si les deux extrémités de chaque arête ne sont pas distinguées ;
- les *arêtes multiples*, c'est-à-dire le fait que plusieurs arêtes puissent avoir les mêmes extrémités, peuvent être autorisées ou non ;
- les arêtes ayant leurs extrémités confondues, appelées *boucles*, peuvent être autorisées ou non ;
- le graphe peut être *fini* (au sens où V et E sont des ensembles finis) ou non.

À partir de la notion de graphe, on définit les notions d'*incidence* entre un sommet et une arête, d'*adjacence* entre deux sommets, ainsi que la notion de *valence* (appelée également *degré*) d'un sommet, c'est-à-dire le nombre d'arêtes qui lui sont incidentes.

La structure de graphe peut être complétée par des *faces*, qui correspondes à un polygone bordé par un cycle d'arêtes de E.

B.2 Les invariants topologiques d'une triangulation surfacique

Nous présentons ici brièvement les quantités appelées *invariants topologiques*, calculées pour une triangulation T avec ou sans bord. Elles traduisent le nombre de bords (c'est-à-dire le nombre de cycles disjoints formés d'arêtes de bord), le fait que la surface soit orientable ou non, le nombre de composantes connexes et le genre topologique. Ces résultats ne se limitent pas aux triangulations surfaciques et peuvent être généralisés à tous types de surfaces. Dans ce mémoire, nous travaillons

exclusivement avec des triangulations *compacts*, car composé d'un ensemble fini de sommets, d'arêtes et de faces.

B.2.1 Orientabilité

Le cycle d'arêtes bordant une face de T, peut être orienté de deux façons différentes, et un choix d'orientation de ce polygone donne une orientation à la face (*cf* figure B.1a). Les faces sont orientées de façon *consistante* si chaque arête incidente à deux faces est orientée de façon contraire au sein de chacune d'elles (*cf* figure B.1b). T est *orientable* s'il est possible de trouver une orientation consistante, et T est *orienté* s'il possède une orientation consistante.

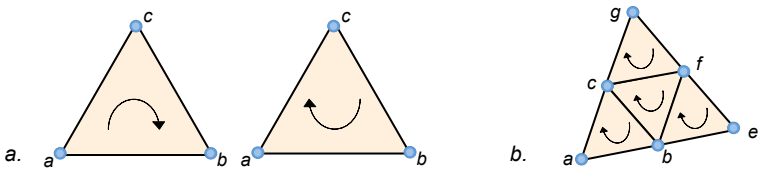


FIGURE B.1 – En a), les deux orientations possibles pour une face. En b), ensemble de faces dont l'orientation est consistante.

B.2.2 Connexité et cycles homotopes

Un *chemin* sur T est une application continue $p : [0,1] \to T$ et ses *extrémités* sont $p(0)$ et $p(1)$. Un chemin est *simple* s'il est injectif, *fermé* sur T (ou un *lacet*) si ses deux extrémités coïncident. Un chemin fermé ne contenant aucune extrémité est appelé un *cycle* et on le définit comme une application continue $Cycl : S^1 \to T$ où S^1 désigne le cercle unité. Un cycle est *simple* s'il est injectif.

• T est dit *connexe* si pour deux points a et b sur le maillage, il existe un chemin sur T d'extrémités a et b. Les *composantes connexes* sont les classes de la relation d'équivalence définie par : a est équivalent à b s'il existe un chemin entre a et b (*cf* figure B.2).

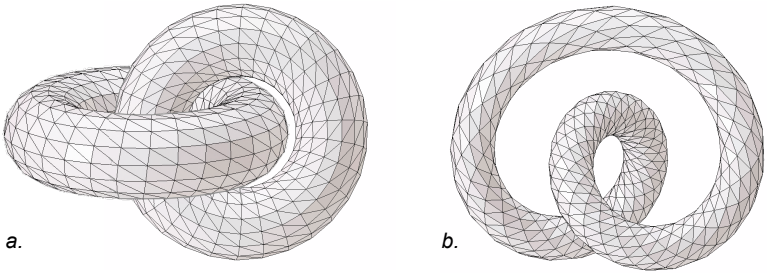


FIGURE B.2 – En a), maillage contenant deux composantes connexes. En b), maillage contenant une seule composante connexe.

• Soient $Cycl_1$ et $Cycl_2$ deux cycles sur T, nous appelons une *homotopie* entre $Cycl_1$ et $Cycl_2$, une application continue $h : [0,1] \times S^1 \to T$ telle que $h(0,.) = Cycl_1$ et $h(1,.) = Cycl_2$. Dans ce cas, $Cycl_1$ et $Cycl_2$ sont dits *homotopes*. Un cycle sur T est *séparateur* si le découpage le long du chemin génère deux composantes connexes et il est *contractile* s'il est homotope à un cycle constant. La notion d'homotopie fournit une relation d'équivalence partitionnant l'ensemble des cycles de T en *classes d'homotopie*.

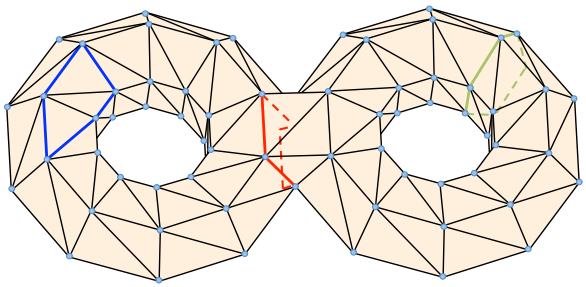


FIGURE B.3 – Surface de genre 2 : en bleu un cycle contractile, en rouge un cycle non contractile et séparateur, en vert un cycle non contractile et non séparateur.

B.2.3 Genre topologique

La *caractéristique d'Euler-Poincaré* de T, notée $\chi(M)$, est un entier relatif égal à $s - a + f$, avec T contenant s sommets, a arêtes et f faces.

Il existe une autre façon de définir la caractéristique d'Euler-Poincaré, pour des surfaces compactes, connexes et orientables. Il s'agit du théorème de classification des surfaces, qui ne sera pas démontré, mais nous renvoyons le lecteur intéressé aux travaux de Brahana [Brahana 1921].

Théorème de classification des surfaces : Soit S une surface compacte, connexe et orientable. Il existe deux entiers positifs ou nuls g et b, uniques, tels que M est homéomorphe à une sphère à laquelle on a collé g poignées et retiré b disques deux à deux disjoints. De plus, $\chi(S) = 2 - 2g - b$.

En particulier, le nombre b représente le nombre de bords de la surface et l'entier g est appelé le *genre topologique* de la surface (*cf* figure B.3). Dans le cas d'une surface S compacte, connexe et orientable, $g = \frac{2-b-\chi(S)}{2}$.

Bibliographie

[Aichholzer 2012] Oswin Aichholzer, Wolfgang Mulzer et Alexander Pilz. *Flip Distance Between Triangulations of a Simple Polygon is NP-Complete*. CoRR, vol. abs/1209.0579, 2012. (Cité en page 55.)

[Alliez 2001] Pierre Alliez et Mathieu Desbrun. *Valence-Driven Connectivity Encoding for 3D Meshes*. In Computer graphics forum, volume 20, pages 480–489. Wiley Online Library, 2001. (Cité en pages 83 et 85.)

[Alliez 2005] Pierre Alliez et Craig Gotsman. *Recent advances in compression of 3D meshes*. In Advances in Multiresolution for Geometric Modelling, pages 3–26. Springer, 2005. (Cité en page 78.)

[Bern 1993] Marshall Bern, Herbert Edelsbrunner, David Eppstein, Scott Mitchell et Tiow Seng Tan. *Edge-Insertion for Optimal Triangulations*. Discrete Computational Geometry, vol. 10, pages 47–65, 1993. (Cité en page 6.)

[Bernardini 1999] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva et Gabriel Taubin. *The ball-pivoting algorithm for surface reconstruction*. Visualization and Computer Graphics, IEEE Transactions on, vol. 5, no. 4, pages 349–359, 1999. (Cité en page 84.)

[Björner 1984] A. Björner. *Posets, Regular CW Complexes and Bruhat Order*. European Journal of Combinatorics, vol. 5, pages 7–16, 1984. (Cité en page 11.)

[Bobenko 2007] Alexander I Bobenko et Boris A Springborn. *A discrete Laplace–Beltrami operator for simplicial surfaces*. Discrete & Computational Geometry, vol. 38, no. 4, pages 740–756, 2007. (Cité en pages 9, 22 et 97.)

[Bose 2011] Prosenjit Bose, Dana Jansens, André van Renssen, Maria Saumell et Sander Verdonschot. *Making triangulations 4-connected using flips*. CoRR, vol. abs/1110.6473, 2011. (Cité en page 55.)

[Brahana 1921] T. Brahana. *Systems of circuits on 2-dimensional manifolds*. Annals of Mathematics, vol. 23, pages 144–168, 1921. (Cité en page 107.)

[Chaine 2003] Raphaëlle Chaine. *A geometric convection approach of 3-D reconstruction*. In Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 218–229. Eurographics Association, 2003. (Cité en page 84.)

[Chaine 2007] Raphaëlle Chaine, Pierre-Marie Gandoin et Céline Roudet. *Mesh connectivity compression using convection reconstruction*. In Proceedings of the 2007 ACM symposium on Solid and physical modeling, pages 41–49. ACM, 2007. (Cité en page 84.)

[Deering 1995] Michael Deering. *Geometry compression*. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pages 13–20. ACM, 1995. (Cité en page 81.)

[Dewdney 1973] A.K. Dewdney. *Wagner's Theorem for Torus Graphs.* Discrete Mathematics, vol. 4, pages 139–149, Fier 1973. (Cité en page 24.)

[Eppstein 2010] David Eppstein. *Happy endings for flip graphs.* Journal of Computational Geometry, vol. 1, pages 3–28, 2010. (Cité en page 55.)

[Gandoin 2001] Pierre-Marie Gandoin. *Compression progressive et sans perte de structures géométriques.* PhD thesis, 2001. (Cité en page 78.)

[Giblin 2010] Peter J Giblin. Graphs, surfaces and homology. Cambridge University Press Cambridge, 2010. (Cité en page 14.)

[Hanke 1996] S. Hanke, T. Ottmann et S. Schuierer. *The Edge-Flipping Distance of Triangulations.* j-jucs, vol. 2, no. 8, pages 570–579, 1996. (Cité en pages 23 et 55.)

[Hatcher 2002] Allen Hatcher. Algebraic topology. Cambridge University Press, 2002. Available at `http://www.math.cornell.edu/~hatcher/`. (Cité en page 9.)

[Huffman 1952] David A Huffman. *A method for the construction of minimum-redundancy codes.* Proceedings of the IRE, vol. 40, no. 9, pages 1098–1101, 1952. (Cité en page 80.)

[Hurtado 1998] Ferran Hurtado, Marc Noyy et Jorge Urrutiaz. *Parallel edge flipping.* In CCCG, 1998. (Cité en page 85.)

[Hurtado 1999] Ferran Hurtado. *Graph of triangulations of a convex polygon and tree of triangulations.* Computational Geometry, vol. 13, no. 3, pages 179–188, 1999. (Cité en pages 18 et 54.)

[Ibarria 2003] Lawrence Ibarria et Jarek Rossignac. *Dynapack : space-time compression of the 3D animations of triangle meshes with fixed connectivity.* In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 126–135. Eurographics Association, 2003. (Cité en page 4.)

[Indermitte 2001] Claude Indermitte, Th M Liebling, Marc Troyanov et Heinz Clémençon. *Voronoi diagrams on piecewise flat surfaces and an application to biological growth.* Theoretical Computer Science, vol. 263, no. 1, pages 263–274, 2001. (Cité en page 23.)

[Kälberer 2005] Felix Kälberer, Konrad Polthier, Ulrich Reitebuch et Max Wardetzky. *FreeLence-Coding with Free Valences.* In Computer Graphics Forum, volume 24, pages 469–478. Wiley Online Library, 2005. (Cité en page 84.)

[King 1999] Davis King et Jaroslaw R Rossignac. *Guaranteed 3.67 v bit encoding of planar triangle graphs.* 1999. (Cité en page 82.)

[Komuro 1997] Hideo Komuro. *The Diagonal Flips of Triangulations on the Sphere.* The Yokohama Mathematical Journal, vol. 44, no. 2, pages 115–122, 1997. (Cité en page 55.)

[Lawson 1972] Charles L. Lawson. *Transforming Triangulations.* Discrete Mathematics, vol. 3, no. 4, pages 365–372, 1972. (Cité en page 21.)

[Lawson 1977] C. L. Lawson. Software for c^1 surface interpolation, pages 161–194. Academic Press, 1977. (Cité en pages 21, 23, 49, 51 et 55.)

[Lazarus 2001] Francis Lazarus, Michel Pocchiola, Gert Vegter et Anne Verroust. *Computing a canonical polygonal schema of an orientable triangulated surface.* In Proceedings of the seventeenth annual symposium on Computational geometry, SCG '01, pages 80–89, 2001. (Cité en page 47.)

[Lee 1988] Carl W. Lee. *The Associahedron and Triangulations of the n-gon.* European Journal Combinatorics, vol. 10, pages 551–560, 1988. (Cité en page 54.)

[Lee 2002] Haeyoung Lee, Pierre Alliez et Mathieu Desbrun. *Angle-Analyzer : A Triangle-Quad Mesh Codec.* In Computer Graphics Forum, volume 21, pages 383–392. Wiley Online Library, 2002. (Cité en page 84.)

[Lewiner 2005] Thomas Lewiner, Marcos Craizer, Hélio Lopes, Sinésio Pesco, Luiz Velho et Esdras Medeiros. *Gencode : Geometry-driven compression in arbitrary dimension and co-dimension.* In Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on, pages 249–256. IEEE, 2005. (Cité en page 84.)

[Lubiw 2012] Anna Lubiw et Vinayak Pathak. *Flip Distance Between Two Triangulations of a Point-Set is NP-complete.* CoRR, vol. abs/1205.2425, 2012. (Cité en page 55.)

[Lucas 1987] Joan M Lucas. *The rotation graph of binary trees is Hamiltonian.* Journal of Algorithms, vol. 8, no. 4, pages 503 – 535, 1987. (Cité en page 18.)

[Muller 2005] Karsten Muller, Aljoscha Smolic, Matthias Kautzner, Peter Eisert et Thomas Wiegand. *Predictive compression of dynamic 3D meshes.* In Image Processing, 2005. ICIP 2005. IEEE International Conference on, volume 1, pages I–621. IEEE, 2005. (Cité en page 4.)

[Munkres 1957] James Munkres. *Algorithms for the assignment and transportation problems.* Journal of the Society for Industrial & Applied Mathematics, vol. 5, no. 1, pages 32–38, 1957. (Cité en page 95.)

[Negami 1990] Seiya Negami et Shin Watanabe. *Diagonal Transformations of Triangulations on Surfaces.* Tsukuba journal of mathematics, vol. 14, pages 155–166, 1990. (Cité en page 25.)

[Negami 1993] Seiya Negami et Atsuhiro Nakamoto. *Diagonal transformations of graphs on closed surfaces.* Sci. Rep. Yokohama Nat. Univ. Sect. I Math. Phys. Chem, vol. 40, pages 71–97, 1993. (Cité en page 25.)

[Negami 1999] Seiya Negami. *Diagonal Transformations of Triangulations on Surfaces, a survey.* Yokohama Mathematical Journal, vol. 47, pages 1–40, 1999. (Cité en page 25.)

[Pachner 1991] Udo Pachner. *P.L. homeomorphic manifolds are equivalent by elementary shellings.* European Journal of Combinatorics, vol. 12, pages 129–8145, 1991. (Cité en page 93.)

[Payan 2005] Frederic Payan, Marc Antonini*et al. Wavelet-based compression of 3d mesh sequences*. Proceedings of ACIDCA-ICMI'2005, 2005. (Cité en page 4.)

[Peng 2005] Jingliang Peng, Chang-Su Kim et C-C Jay Kuo. *Technologies for 3D mesh compression : A survey*. Journal of Visual Communication and Image Representation, vol. 16, no. 6, pages 688–733, 2005. (Cité en page 78.)

[Pilz 2012] Alexander Pilz. *Flip Distance Between Triangulations of a Planar Point Set is NP-Complete*. CoRR, vol. abs/1206.3179, 2012. (Cité en page 55.)

[Poulalhon 2003] Dominique Poulalhon et Gilles Schaeffer. *Optimal coding and sampling of triangulations*. In Automata, languages and programming, pages 1080–1094. Springer, 2003. (Cité en pages 81 et 85.)

[Pournin 2012] Lionel Pournin. *The diameters of associahedra*. arXiv preprint arXiv :1207.6296, 2012. (Cité en pages 55 et 72.)

[Rissanen 1979] Jorma Rissanen et Glen G Langdon. *Arithmetic coding*. IBM Journal of research and development, vol. 23, no. 2, pages 149–162, 1979. (Cité en page 80.)

[Rissanen 1983] Jorma Rissanen. *A universal data compression system*. Information Theory, IEEE Transactions on, vol. 29, no. 5, pages 656–664, 1983. (Cité en page 80.)

[Rossignac 1999] Jarek Rossignac. *Edgebreaker : Connectivity compression for triangle meshes*. Visualization and Computer Graphics, IEEE Transactions on, vol. 5, no. 1, pages 47–61, 1999. (Cité en page 82.)

[Sattler 2005] Mirko Sattler, Ralf Sarlette et Reinhard Klein. *Simple and efficient compression of animation sequences*. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 209–217. ACM, 2005. (Cité en page 4.)

[Shannon 1951] Claude E Shannon. *Prediction and entropy of printed English*. Bell system technical journal, vol. 30, no. 1, pages 50–64, 1951. (Cité en page 78.)

[Sleator 1986] D D Sleator, R E Tarjan et W P Thurston. *Rotation distance, triangulations, and hyperbolic geometry*. In Proceedings of the eighteenth annual ACM symposium on Theory of computing, STOC '86, pages 122–135, New York, NY, USA, 1986. ACM. (Cité en page 101.)

[Sleator 1987] Daniel Sleator, Robert Tarjan et William Thurston. *Rotation distance*. In Open problems in communication and computation. Springer-Verlag, 1987. (Cité en pages 23, 55, 101 et 103.)

[Spanier 1994] Edwin H Spanier. Algebraic topology, volume 55. Springer, 1994. (Cité en page 14.)

[Stanly 1991] Richard P. Stanly. *f-vectors and h-vectors of simplicial posets*. Journal of Pure and Applied Algebra, vol. 71, pages 319–331, 1991. (Cité en page 11.)

[Szymczak 2001] Andrzej Szymczak, Davis King et Jarek Rossignac. *An Edgebreaker-based efficient compression scheme for regular meshes*. Computational Geometry, vol. 20, no. 1, pages 53–68, 2001. (Cité en page 82.)

[Taubin 1998] Gabriel Taubin et Jarek Rossignac. *Geometric compression through topological surgery.* ACM Transactions on Graphics (TOG), vol. 17, no. 2, pages 84–115, 1998. (Cité en page 82.)

[Touma 1998] Costa Touma et Craig Gotsman. *Triangle mesh compression.* In Graphics interface, pages 26–34. Canadian Information Processing Society, 1998. (Cité en pages 83 et 84.)

[Tutte 1962] William Thomas Tutte. *A census of planar triangulations.* Canad. J. Math, vol. 14, no. 1, pages 21–38, 1962. (Cité en pages 79, 84 et 85.)

[Valette 2009] Sébastien Valette, Raphaëlle Chaine et Rémy Prost. *Progressive lossless mesh compression via incremental parametric refinement.* In Proceedings of the Symposium on Geometry Processing, SGP '09, pages 1301–1310, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association. (Cité en pages 7, 31, 78 et 90.)

[Váša 2007] Libor Váša et Václav Skala. *Coddyac : Connectivity driven dynamic mesh compression.* In 3DTV Conference, 2007, pages 1–4. IEEE, 2007. (Cité en page 4.)

[Váša 2009] Libor Váša et Václav Skala. *Combined compression and simplification of dynamic 3D meshes.* Computer Animation and Virtual Worlds, vol. 20, no. 4, pages 447–456, 2009. (Cité en page 4.)

[Wagner 1936] K. Wagner. *Bemekungen zum Vierfarbenproblem.* Jahresber. Deutsch. Math.-Verein., vol. 46, pages 26–32, 1936. (Cité en pages 12 et 23.)

[Witten 1987] Ian H Witten, Radford M Neal et John G Cleary. *Arithmetic coding for data compression.* Communications of the ACM, vol. 30, no. 6, pages 520–540, 1987. (Cité en page 80.)

[Yang 2002] Jeong-Hyu Yang, Chang-Su Kim et Sang-Uk Lee. *Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction.* Circuits and Systems for Video Technology, IEEE Transactions on, vol. 12, no. 12, pages 1178–1184, 2002. (Cité en page 4.)

[Ziv 1977] Jacob Ziv et Abraham Lempel. *A universal algorithm for sequential data compression.* Information Theory, IEEE Transactions on, vol. 23, no. 3, pages 337–343, 1977. (Cité en page 79.)

[Ziv 1978] Jacob Ziv et Abraham Lempel. *Compression of individual sequences via variable-rate coding.* Information Theory, IEEE Transactions on, vol. 24, no. 5, pages 530–536, 1978. (Cité en page 79.)