Topologie Algorithmique
Combinatoire et Plongement

Bruno Lévy

February 7, 2000

Table des matières

À Nathalie, à mes parents.

Résumé

Dans le domaine de la modélisation 3D, l'objectif est de définir des moyens de représenter une réalité physique par des objets informatiques. Afin de permettre des simulations de phénomènes physiques, le modèle informatique doit représenter non seulement la forme des objets concernés, mais aussi des propriétés physiques attachées à ces objets.

La modélisation 3D s'appuie sur deux principales familles de méthodes. L'une de ces familles de méthodes, appelée souvent "courbes et surfaces", se fonde sur une représentation des objets à modéliser par des fonctions (le plus souvent polynomiales). L'autre famille de représentations consiste à discrétiser les objets en cellules (sommets, segments, polygones, polyèdres ...). Nous étudions ici les problèmes liés à ce dernier type de représentation discrète des objets, ainsi que ses relations avec les "courbes et surfaces". En utilisant le formalisme offert par la Topologie, une branche moderne des mathématiques, nous allons étudier les problèmes suivants:

- Définir des structures de données efficaces pour représenter la décomposition des objets en éléments discrets.
- Générer et éditer interactivement des objets, de manière à respecter des données ainsi que des contraintes globales concernant la forme des objets.
- Construire une paramétrisation sous contraintes d'une surface triangulée, afin de pouvoir facilement lui associer des valeurs.

Le premier point sera traité en utilisant certains résultats de topologie combinatoire, et les deux derniers seront étudiés en termes d'homéomorphisme et de transformation continue.

Nous présenterons également plusieurs applications de ces méthodes, permettant de résoudre des problèmes de modélisation en géologie numérique. Par exemple, nous montrerons comment modéliser de manière précise les variations de porosité de la roche à l'intérieur d'un réservoir naturel. Des applications possibles de nos méthodes à l'image de synthèse et au placage de textures seront également évoquées.

Abstract

In the realm of 3D modeling, the principal purpose is to define new models, enabling physical objects to be represented by computer science objects. In order to perform physical simulations involving those objects, they should be provided not only with a geometrical description, but also with physical properties attached to them.

Two main approaches to 3D modeling exist. The first one, often referred to as "curves and surfaces", consists in representing the objects by mathematical functions (mostly polynomials). The other family of approaches consists in decomposing the objects into cells (i.e. vertices, segments, polygons, polyhedra ...). This work focuses on this latter kind of representation, and to the issues raised by such a discrete setting. The relations with "curves and surfaces" methods will be discussed as well. By using the formalism provided by Topology, a recent branch of mathematics, we will study the following issues:

- Defining efficient data structures enabling the decomposition of objects into discrete elements to be represented.
- Interactively generating and editing objects while respecting data as well as global shape constraints.
- Constructing a parameterization of a triangulated surface under constraints, in order to paint it with dense scalar fields.

The first point will be treated by using results from combinatorial topology, and the two other ones will be studied in terms of homeomorphisms and continuous maps.

We will present several possible application of the method, making it possible to solve 3D modeling problems in numerical geology. For instance, it will be shown how to accurately model the variation of rock porosity within a natural reservoir. Other possible applications of our methods to computer graphics and to texture mapping will be also described.

Remerciements

Je tiens à remercier Jean-Laurent Mallet, qui a dirigé et suivi de très près ce travail de recherche. Entre autres choses, il m'a appris à écrire, à dessiner et à parler, ce pour quoi je le remercie vivement. Merci également pour ses nombreuses idées qui m'ont stimulées durant ces trois années.

Merci à *Gaz de France* et au CNRS, qui ont co-financé ce travail. Merci en particulier à Liliane Wietzerbin et à François Verdier, qui m'ont suivi pendant ces trois années. Leurs conseils m'ont permis de mieux cerner les applications géologiques des méthodes développées.

Malgrès leur emploi du temps très chargé, Claude Puech, Jean-Daniel Boissonnat et Jarek Rossignac ont accepté de rédiger un rapport sur cette thèse, ce pour quoi je suis très honorés. Je les remercie également pour l'intérêt qu'ils portent à mes travaux.

Merci à Pascal Lienhardt et à Yves Betrand, dont les précieux conseils ont largement contribué à la réalisation du modèle combinatoire.

Je remercie également Jean-Claude Paul, qui a accepté de faire partie du jury. Lors de mon stage de deuxième année d'école d'ingénieur, il m'a également initié à l'infographie, discipline qui est devenue une passion pour moi. Merci également à Jens Gustedt, pour avoir accepté d'être rapporteur interne.

Un grand merci à Nathalie, pour sa patience et ses conseils avisés, tout au long de ces trois années.

Je suis très reconnaissant envers Sophie Viseur, qui a relu en profondeur le manuscrit, ses remarques et suggestions m'ont permis d'en rendre la lecture plus aisée.

Merci à Stéphane Conreaux qui partage mon bureau (et mon intérêt pour la topologie combinatoire). Merci pour sa bonne humeur constante et pour le fait qu'il soit toujours prêt à donner un coup de main à tout le monde. Merci aussi à Karen Pairazian, qui a supporté nos mauvais jeux de mots, et à Jérome Massot, qui en a commis de pires !

Je remercie également Stéphane Conreaux, Isabelle Duvinage, Magali Lecour, Nathalie et Christelle Lequart, Joël Conraud, ainsi que mes parents, qui ont eu le courage d'affronter la lecture de ce mémoire, et qui ont détecté un nombre considérable d'erreurs et de fautes de frappe.

Merci à Sophie Viseur pour le dessin de la texture du visage, à Corine Schlumberger et Jérôme Maillot (Alias|Wavefront) pour les objets 3D (le visage et le chien) qui apparaissent dans le troisième chapitre. Merci à Nicolas Euler, Olivier Mariez et Stéphane Conreaux pour les modèles géologiques du sous-sol qu'ils ont construits et pour les images qu'ils m'ont autorisées à inclure dans ce mémoire.

Puisque ce travail a été réalisé dans le cadre du projet Gocad, je tiens à remercier ses principaux architectes, à savoir Jean-Laurent Mallet (qui a développé la première version en C de Gocad), Jean-Claude Dulac (qui outre la création de l'entreprise T-Surf, a accompli un travail "titanesque" sur la version C++), Thierry Valentin (qui assure la cohérence du projet, développe et maintient les outils de base), Fabien Bosquet et Pierre Jaquemin (qui ont développé - entre autres - la librairie graphique), Richard Cognot (qui a écrit - entre autres - l'implantation de l'interpolateur D.S.I. dans la version C++) et Arben Shtuka (qui a développé les méthodes géostatistiques). Merci également à Nathalie Flammang qui assure le bon fonctionnement du système informatique, et à Monique Cugurno, notre secrétaire de choc.

Enfin, merci au reste de l'équipe du LIAD, et à l'entreprise T-Surf, a qui je souhaite longue vie. L'environnement très stimulant que forme l'ensemble des deux entités a été - et continuera a être - un facteur de motivation non négligeable.

Introduction

Modélisation 3D en géosciences

Cette thèse s'inscrit dans le domaine de la modélisation 3D, dont l'objectif est de définir des outils pour représenter des objets tridimentionnels à l'aide de l'outil informatique. Plus précisément, dans le cadre de ce travail, nous nous intéresserons à une classe de problèmes posés dans le cadre de la modélisation du sous-sol, encore appelée *géo-modélisation*. Avant de décrire le contexte théorique dans lequel s'inscrit notre approche, nous allons donner une idée des problèmes de modélisation qui se posent en géologie numérique.

Notre étude, co-financée par *Gaz de France* et par le C.N.R.S.[1], s'inscrit dans le cadre du projet Gocad[2]. L'objectif de ce projet est la mise au point d'un ensemble d'outils informatiques permettant de réaliser un modeleur 3D adapté au monde de la géologie. Dans le domaine de la modélisation géologique, nous nous intéressons à une zone du sous-sol, appelée la *zone d'intérêt* (ou encore le domaine d'étude). Cette zone du sous-sol va être représentée par un ensemble de structures informatiques, permettant de représenter les frontières entre les différentes couches géologiques, et de représenter des propriétés physiques attachées à ces couches géologiques (telles que la porosité des roches, leur perméabilité, la nature des roches ...).

C'est ce dernier aspect qui a suscité l'intérêt de *Gaz de France* , commanditaire de cette étude. Plus précisément, à l'origine du projet, un des objectifs de *Gaz de France* est d'étudier les possibilités d'utiliser des structures géologiques naturelles pour stocker du gaz dans le sous-sol. Ainsi, une couche de grès poreux surmontée d'une couche d'argile imperméable peut constituer un piège naturel dans lequel il est envisageable de stocker du gaz. Lors d'une telle opération, il est nécessaire de connaître le mieux possible la structure du réservoir, amené à se vider l'hiver et à se remplir l'été de très nombreuses fois durant la vie du stockage. La localisation et la manière dont le gaz mobilisable se déplace sont essentiels à connaître pour en optimiser le développement et le fonctionnement. En utilisant une représentation

[1] Centre National de la Recherche Scientifique
[2] http://www.ensg.u-nancy.fr/GOCAD/

de la perméabilité de la roche, des méthodes de simulation numérique permettent d'estimer avec quelle efficacité une structure géologique pourrait jouer le rôle de réservoir naturel. Ces méthodes ont pour objectif de connaître la quantité de gaz stockable ainsi que l'efficacité avec laquelle du gaz pourra être soutiré (l'hiver) ou injecté (l'été). Les mêmes simulateurs d'écoulements sont également utilisés par les entreprises pétrolières, afin de prédire la production de pétrole d'un puits.

Afin de comprendre la structure du sous-sol, et à défaut de pouvoir en traduire toute la complexité, les géologues considèrent un découpage de la zone d'intérêt en couches géologiques, où la nature de la roche est homogène. Les interfaces entre ces couches géologiques définissent la structure géométrique du sous sol, et jouent pour cette raison un rôle prépondérant dans le processus de modélisation. Ces interfaces sont de deux types : les *horizons*, surfaces plutôt horizontales (comme leur nom l'indique), correspondent à la zone de contact entre deux couches géologiques, et les *failles* sont des zones de rupture, causées par descontraintes mécaniques que les couches ont subies. De part et d'autre de ces interfaces, les propriétés des roches peuvent varier de manière très brutale et discontinue. Elles peuvent constituer une barrière à l'écoulement du gaz. C'est ce qui rend également le domaine des paramètres géologiques très difficile à modéliser. Ces différentes structures étant souterraines, il est à la fois très difficile et très coûteux d'obtenir des données permettant de les modéliser. Les données en question sont essentiellement de deux types :

- Les données de puits, échantillonées le long de courbes correspondant à des forages. Ces données fournissent une information exacte sur la profondeur des différentes couches géologiques (dans certains cas, le vecteur normal à chaque couche est également disponible). Ces données, de nature ponctuelle, sont en général peu nombreuses, mais relativement fiables, car résultant d'un processus direct d'acquisition.

- Les données sismiques, obtenues en déclenchant une explosion et en enregistrant les vibrations à l'aide de microphones spéciaux. Après regroupement, ces données sont disponibles sous la forme d'un cube, discrétisé en voxels. Des logiciels appelés des *stations d'interprétation sismiques* permettent d'extraire des nuages de points de ces cubes de données, correspondant aux horizons. En général, ces nuages de points couvrent globalement le domaine d'étude, mais sont beaucoup moins fiables que les données de puits. Ceci est dû à la longue chaîne de traitement entre les capteurs et le cube sismique, qui nécessite de plus une connaissance *à-priori* de la vitesse de propagation des ondes sismiques dans le domaine d'étude.

Dans le cadre du projet Gocad, des méthodes ont été définies par Mallet dans [Mal89, Mal92, Mal99] pour construire des surfaces triangulées passant exactement par les données de puits, et approximativement par les horizons extraits

des données sismiques. À partir de ces surfaces, notre objectif est de mettre au point un nouveau type de support géométrique, permettant de modéliser précisément la variation d'une propriété physique sur le domaine d'étude. Ce support géométrique devra respecter précisément les horizons et les failles du modèle, et devra offrir des possiblités d'édition interactive.

La topologie: un formalisme pour la modélisation 3D

Dans un contexte plus général, la définition d'un support géométrique est un problème de modélisation tridimensionnelle. Cette dernière thématique de recherche a connu de nombreux développements ces dix dernières années, dûs à la fois aux grands progrès technologiques réalisés dans le domaine de la puissance brute des ordinateurs, mais surtout à une meilleure maîtrise des structures de données et algorithmes associés. Plus précisément, notre thème de recherche concerne la *topologie algorithmique*[3], une nouvelle discipline introduite pour la première fois par Dey, Edelsbrunner et Guha lors de la conférence *Computational Geometry - Ten Years After* [DEG96]. Dans leur article, ces auteurs définissent cette thématique comme pouvant regrouper sous un formalisme cohérent différentes recherches en cours dans plusieurs disciplines. L'ensemble des disciplines concernées compte par exemple la synthèse d'images, la génération de maillages, la cartographie, la modélisation volumique, ou encore le traitement d'images. Dans toutes ces disciplines, lorsque nous approfondissons l'étude de certains problèmes, les mêmes questions fondamentales ne cessent d'être soulevées. Certaines de ces questions concernent la *géométrie* des problèmes, à savoir la position et la forme des objets mis en jeu, et ont été très bien étudiées dans l'abondante littérature concernant ce sujet, comme nous le verrons plus loin. D'autres questions vont concerner les notions moins connues de connectivité, de continuité, ou encore de transformation de l'espace. Ces dernières notions sont également liées à la géométrie des problèmes, mais d'une manière plus indirecte. Ce sont des notions dites *topologiques*, qui caractérisent les objets en termes de voisinages et connexité plutôt qu'en termes de géométrie.

Datant de la fin du XVIII^{eme} siècle, la topologie est une branche relativement récente des mathématiques [Ago76]. Fondée sur un nombre d'axiomes très faible, elle forme un édifice cohérent, pouvant être défini de manière relativement indépendante du reste des mathématiques. Intuitivement, la topologie peut être considérée comme la science des objets en pâte à modeler, car elle se préoccupe des caractéristiques de ces objets invariantes par rapport à un certain type de déformations continues de l'objet considéré. Par exemple, comme nous le mon-

[3] computational topology

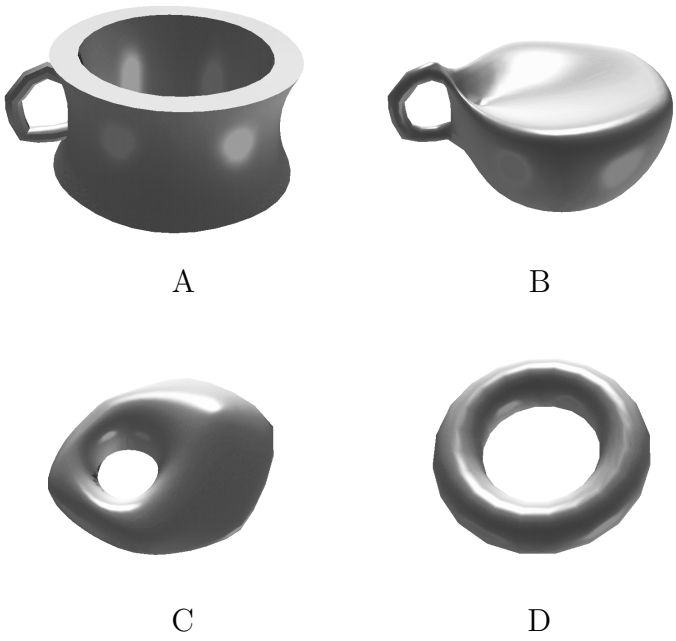


Figure 1: *La topologie peut être considérée comme la science des objets en pâte à modeler. Pour le spécialiste de ce domaine, une tasse à café (A) et un beignet (D) sont deux objets équivalents, car il est possible de passer de l'un à l'autre par une transformation continue appelée un homéomorphisme. Ainsi, la topologie s'intéresse aux caractéristiques des objets invariantes par un homéomorphisme.*

trons Figure 1, du point de vue de la topologie, une tasse à café et un beignet sont deux objets équivalents, car il est possible de passer de l'un à l'autre par une transformation continue. Cette dernière transforme le trou du tore en l'anse de la tasse. Ainsi, le nombre de trous[4] d'un objet est une caractéristique topologique (une transformation qui boucherait un trou n'est pas autorisée par la topologie).

Le point de vue défendu par Dey, Edelsbrunner et Guha dans [DEG96] implique une prise en compte de ce type de considérations, d'ordre purement topologique, comme une problématique à part entière, à séparer des considérations d'ordre géométrique (même si les relations entre les deux peuvent être étudiées par la suite).

Dans le contexte scientifique de la modélisation 3D par ordinateur, Weiler [Wei84] a été l'un des premiers à mentionner l'importance de la topologie en tant que discipline à part entière. Il a décrit des structures combinatoires permettant de représenter des surfaces discrétisées en polygones. Malheureusement, bien qu'il

[4]Il existe en réalité plusieurs types de trous. En topologie, le trou d'un tore est en fait appelé une anse.

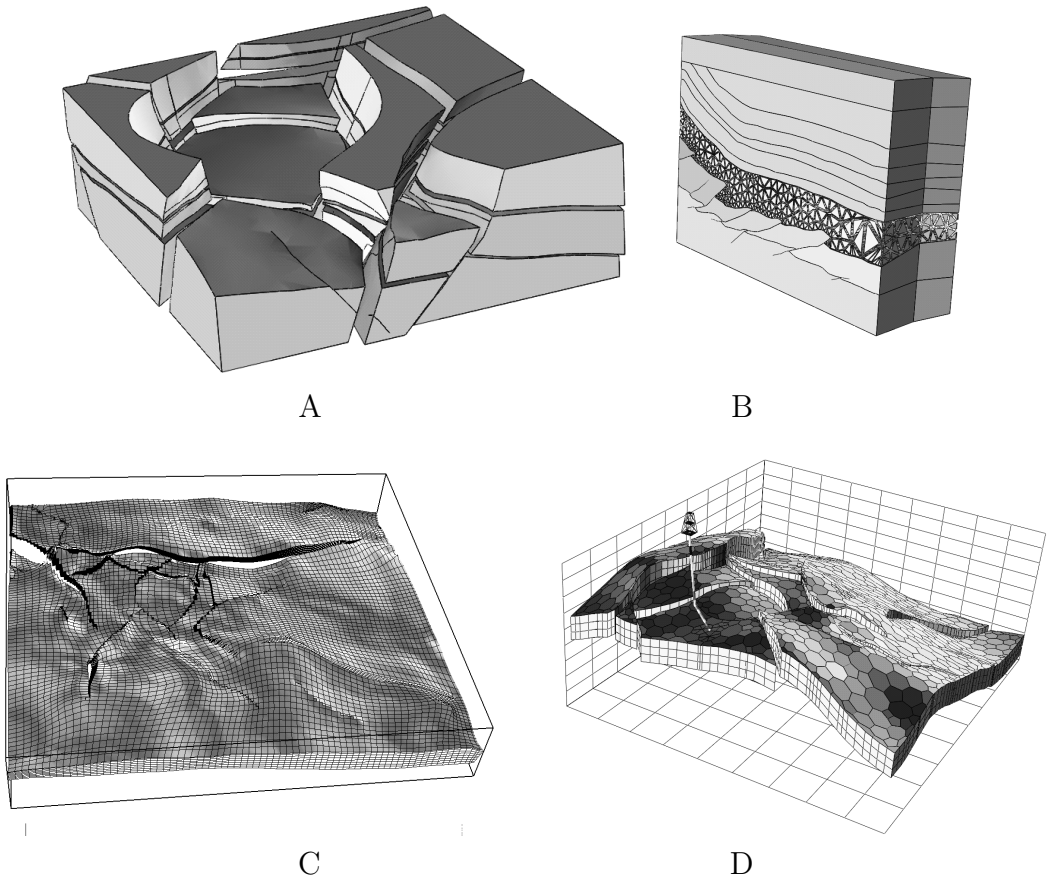


Figure 2: *Exemples d'objets représentés dans le géomodeleur Gocad. A: Modèle 3D du sous-sol. B: Un autre modèle 3D, pour lequel nous montrons la subdivision de la frontière de l'une des régions en polygones, arêtes et sommets. C: Grille 3D structurée, dont les cellules sont des hexaèdres; D: Grille 3D non structurée, dont les cellules sont des polyèdres quelconques.*

ait démontré plusieurs propriétés mathématiques pour la structure qu'il a définie, elle reste très éloignée des autres notions mathématiques relatives à la topologie. Il est vrai que la topologie est une branche des mathématiques impliquant un formalisme relativement complexe ainsi que des notions très abstraites. C'est pourquoi certaines recherches effectuées en informatique ont eu malheureusement pour effet de réinventer d'une manière moins bien formalisée certains résultats déjà connus. Néanmoins, la topologie est également une discipline bénéficiant d'une axiomatique particulièrement compacte, ce qui la rend facilement applicable dans un contexte informatique. Ce dernier aspect a été exploité par Lienhardt [Lie88, Lie94], qui définit un ensemble de structures combinatoires décrivant la discrétisation des objets.

Dans le cadre de notre étude, dont l'objectif est de mettre au point un nouveau support pour la modélisation de propriétés physiques, nous nous proposons de traiter certains points du programme de recherche proposé par Dey, Edelsbrunner et Guha dans [DEG96]. Dans le domaine de la géologie numérique, les utilisateurs s'intéressent à un volume 3D représentant une certaine partie du sous-sol. Ce volume 3D, appellé aussi *zone d'intérêt*, est partitionné en plusieurs zones volumiques, appelées des *régions*, qui correspondent aux couches géologiques au sein d'un même compartiment (voir Figure 2-A). Ces régions sont elles mêmes représentées par leur frontière, c'est-à-dire par un ensemble de surfaces appelées des *interfaces*, qui correspondent aux limites entre les couches géologiques (les *horizons*) et aux failles. Les interfaces sont représentées dans Gocad par des surfaces discrétisées en polygones (voir Figure 2-B). Afin d'effectuer différents calculs numériques, les régions d'un modèle vont être munies de valeurs numériques, définies en tous points de l'espace, correspondant à des valeurs physiques modélisées (par exemple la porosité, la perméabilité, la température ...). Afin de représenter ces champs de valeurs, il est possible de remplir les régions d'un modèle avec des grilles 3D, structurées (Figure 2-B) ou non (Figure 2-C), qui vont servir de support pour les champs scalaires et vectoriels définis sur la zone d'intérêt. Nous verrons dans les différentes parties de ce mémoire un ensemble de méthodes pour représenter et pour construire les différents types d'objets formant un modèle 3D, à savoir les surfaces discrétisées en polygones, ainsi que les grilles structurées ou non.

La topologie se compose de plusieurs sous-domaines: en ce qui concerne les objets discrétisés en sommets, arêtes, triangles ..., la *topologie combinatoire* permet de déterminer certaines propriétés et théorèmes concernant la manière dont ces éléments sont connectés entre eux. La première partie de ce mémoire lui est consacrée. Nous y montrons comment cette discipline permet une définition particulièrement compacte des objets du modeleur, ainsi qu'une étude systématique de certains algorithmes. Ceci va nous permettre d'étudier une manière de considérer un modèle géologique comme une partition de l'espace 3D, et de définir une structure hiérarchique en permettant une représentation efficace. Dans ce premier chapitre, nous introduisons un nouveau modèle topologique, les H-G-Cartes (ou G-Cartes hierarchiques), permettant de modéliser facilement des subdivisions de l'espace 3D (comme des couches geologiques). Notre modèle tient compte du fait que dans notre cas, l'utilisateur préfère manipuler dans un premier temps des surfaces, pour ensuite les assembler sous forme de volumes. L'originalité de notre approche consiste à laisser l'utilisateur croire qu'il manipule des surfaces, alors que du point de vue combinatoire, une telle surface correspond à l'interface entre deux volumes. Ceci fournit aux problèmes de modélisation non-variété une solution plus simple que celles rencontrées dans la litterature (par exemple [Wei85]). Ce chapitre contient également une introduction intuitive aux méthodes de topologie combinatoire, dans l'objectif de rendre le formalisme de

ces methodes plus facilement accessible à la communauté géosciences et C.A.O. Ainsi, nous introduisons la notion de G-Carte décrite par Lienhardt [Lie94], en nous fondant sur la notion de graphe d'incidence, plus facilement accessible que celle de triangulation barycentrique utilisée par Lienhardt.

L'autre principal centre d'intérêt de la topologie est moins bien connu dans le domaine de l'informatique et de la modélisation 3D. Cet aspect concerne les ensembles de points, structurés par une définition de *voisinages*, ainsi qu'une classe de fonctions continues, appelées des *homéomorphismes*, qui préservent la structure des voisinages. Le deuxième et le troisième chapitre de ce mémoire sont consacrées à cette classe de problèmes. Plus précisément, dans le deuxième chapitre, nous nous intéressons à la création et à la modification de surfaces triangulées dont la forme satisfait certains critères. Ceci va nous permettre de représenter les failles et les horizons du modèle géologique, tout en assurant certaines propriétés géométriques utiles. Dans ce deuxième chapitre, nous définissons pour des maillages discrets l'équivalent des méthodes de lissage et d'ajustement aux données connues pour des surfaces polynomiales. Ceci permet de combiner les avantages des deux types de supports, tout en s'affranchissant de certaines limitations des méthodes connues. Par rapport aux surfaces de subdivision [ZSS97], notre approche permet de diminuer le nombre de polygones de la surface finale, et par rapport aux autres methodes de lissage discret [TGHL98, WW92], l'introduction de contraintes linéaires permet l'ajustement aux donnees, ainsi que la réalisation d'autres outils de manipulation interactive. Notre approche généralise les méthodes existantes, en ne faisant aucune supposition a-priori sur les paramétrisation locales utilisees, comme cela est fait dans [TGHL98, KCVS98] (qui suppose une parametrisation uniforme) et dans [WW92] (qui impose l'utilisation de cartes exponentielles). Ceci nous permet de règler certains problemes d'oscillations des bords, qui apparaissent lorsque ces méthodes sont utilisées.

Dans le troisième chapitre, nous montrons comment construire un homéomorphisme mettant en correspondance une surface triangulée avec un sous ensemble de $\mathbb{R}^2$ (autrement dit, une paramétrisation d'une surface). Ce problème a déjà été mentionné comme un problème ouvert dans [DEG96], et a été résolu pour certains cas particuliers, par les mêmes auteurs, dans [EW97]. Les principales applications de ce problème concernent le placage de texture, mais permettent également de résoudre des problèmes de modélisation, (voir Section 3.5). Nous proposons pour ce problème une solution générale, fondée sur la minimisation sous contraintes d'une fonctionnelle. Les contraintes fournissent à l'utilisateur la possibilité de spécifier des informations à prendre en compte dans la construction de l'homéomorphisme. Le probleme de la paramétrisation d'une surface est un probleme "mal posé" (qui admet a-priori une infinité de solutions). Même si plusieurs approches semblaient interessantes [Flo97], [KL96], [MYV93], il man-

quait toujours une formalisation générale du problème. A part celle de Floater [Flo97], aucune de ces méthodes ne permet de construire une paramétrisation de manière automatique, et la méthode de Floater ne permet pas de prendre en compte des contraintes specifiées par l'utilisateur. Nous verrons alors dans le Chapitre 3 comment exprimer ce problème comme la minimisation d'une fonctionelle sous contraintes, correspondant a la minimisation des distortions. Ceci nous permet de définir un formalisme qui englobe tous les autres, d'introduire de nouvelles contraintes de non-distortion, de traiter des surfaces dechirées, ou encore de respecter des correspondances entre le modèle et la texture, specifiées par l'utilisateur. Ces derniers aspects existaient déjà individuellement dans plusieurs approches, sous d'autres formes, mais aucune d'entre elles ne permettait de les combiner de maniere consistante. De plus, notre modèle est "ouvert", et peut facilement être étendu en introduisant de nouvelles contraintes.

Les trois chapitres de ce mémoire peuvent se lire de manière indépendante. Les notions présentées dans le premier chapitre peuvent être utilisées par les deux autres, sans nécessiter de connaître le détail des structures internes. Toutefois, en ce qui concerne les Chapitres 2 et 3, la modélisation variationnelle (Chapitre 2) et la notion de paramétrisation (Chapitre 3) sont deux concepts partageant une certaine base théorique commune.

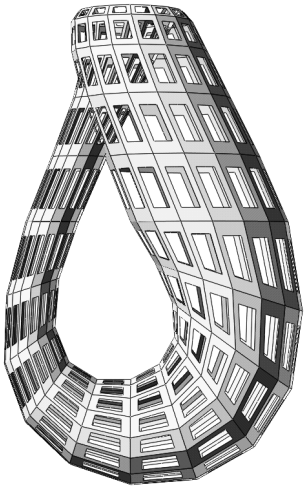
Le premier chapitre se place dans un monde abstrait, et ne concerne que les connections entre les objets, définies indépendamment de toute propriété géométrique. Une fois qu'un tel modèle combinatoire est défini, il peut être utilisé tel quel par les méthodes introduites dans les deux autres chapitres. Il doit ensuite être complété par une définition géométrique, afin de permettre une représentation de la forme des objets du modeleur. Nous verrons dans le Chapitre 2 une manière de construire des surfaces triangulées possédant certaines propriétés géométriques utiles. A ce niveau, un modèle géologique devient un ensemble de boites vides, correspondant aux couches géologiques. Le Chapitre 3 montre une manière de remplir ces couches par des grilles, pouvant servir de support à des propriétés.

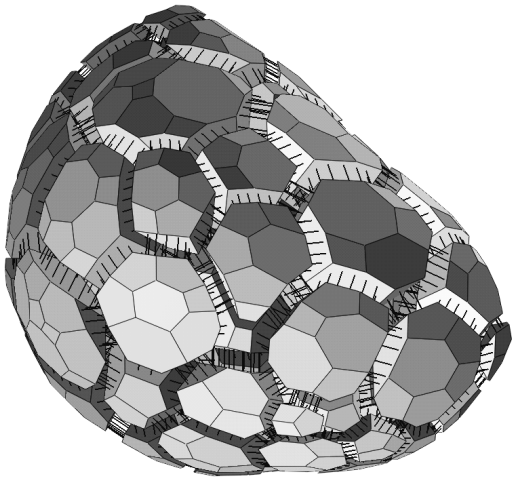
D'un point de vue pratique, nous avons implanté dans le logiciel Gocad les différentes méthodes décrite dans ce mémoire. Le noyau topologique décrit dans le Chapitre 1 sert actuellement de plateforme de développement à plusieurs travaux de recherche effectués au laboratoire. Il est prévu d'en réaliser une version industrielle. Les méthodes introduites dans les Chapitres 2 et 3 ont été intégrées au logiciel Gocad. Ces méthodes permettent respectivement d'améliorer les fonctionalités de lissage sous contraintes déjà offertes par Gocad, et de construire des grilles à partir de surfaces triangulées.

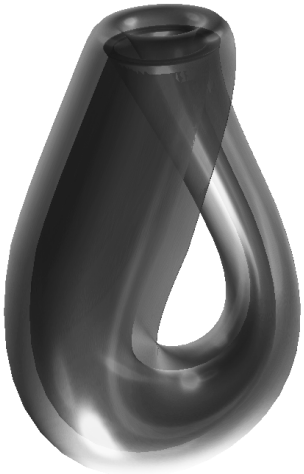
Au cours des chapitres, nous utiliserons des exemples de natures diverses, parfois n'ayant aucun rapport direct avec la géologie (comme un visage humain). Ceci nous permet de faciliter la compréhension de certaines méthodes. En effet, comme le lecteur est en général accoutumé à reconnaître les éléments caractéristiques des visages humains, ceci facilite la perception en trois dimensions des surfaces concernées. De plus, ceci permet de présenter des applications possibles de nos méthodes dans un contexte plus général que celui de la géométrie numérique.

Chapitre 1

Modèles combinatoires





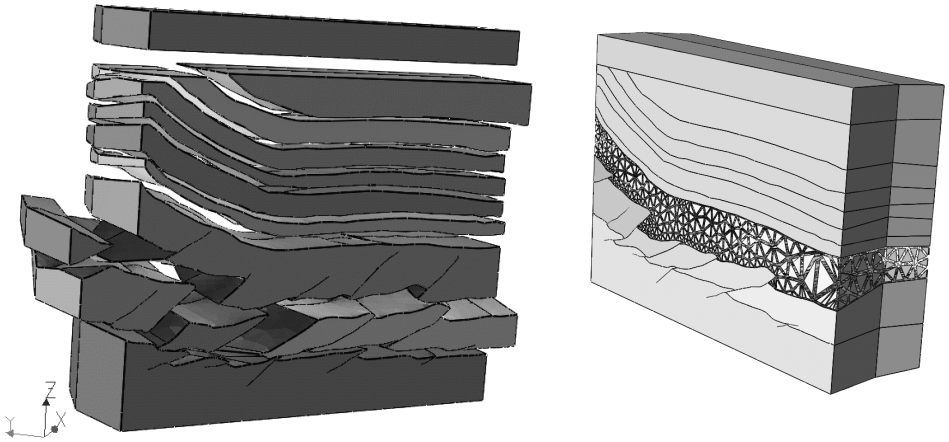


Introduction

La topologie combinatoire est une branche récente des mathématiques, qui promet d'apporter certaines améliorations à la modélisation géométrique et à la C.A.O. Dans cette optique, cette partie introduit la notion de *Carte Généralisée Hiérarchique* (H-G-Carte), qui peut être utilisée pour implanter un noyau de modeleur en dimension arbitraire. Un tel noyau peut être utilisé pour des applications industrielles, comme la résolution d'équations aux dérivées partielles. Pour de telles applications, le noyau du modeleur doit fournir un moyen de discrétiser les objets en un assemblage de formes élémentaires, appelées des *cellules*. Les opérations d'édition des objets, ainsi que les calculs numériques nécessitent de pouvoir "naviguer" efficacement dans la structure considérée. Plus précisément, comme nous le verrons dans la Section 1.1, nous souhaitons pouvoir retrouver efficacement l'ensemble des cellules en contact avec une cellule donnée.

Les approches classiques pour ce type de représentations nécessitent de définir un grand nombre d'entités et de relations les liant entre elles, ou bien ont un domaine de modélisation limité aux objets simpliciaux. Les H-G-Cartes que nous introduisons dans ce chapitre permettent une plus grande flexibilité, tout en nécéssitant des structures de données plus simples. Notre approche se fonde sur la notion de carte généralisée (G-Carte) [Lie94], dont la définition ne requiert qu'un seul type élémentaire. Cette structure permet de représenter la topologie d'objets de dimensions arbitraires (des surfaces, des solides, des hyper-solides ...) et constitués de primitives présentant un nombre arbitraire de faces et de côtés. L'origine mathématique des G-Cartes permet de définir plusieurs façons de caractériser et tester la validité des objets modélisés, ce qui peut être un point crucial dans le cadre d'applications industrielles.

Dans notre contexte applicatif, celui de la modélisation 3D appliquée à la géologie, notre objectif est de créer une base de donnée hiérarchisée permettant de représenter des subdivisions de $\mathbb{R}^3$. Comme nous le montrons Figure 1.1-A, au premier niveau, la zone d'intérêt est subdivisée en régions volumiques correspondant aux couches géologiques. Ces régions sont représentées par leurs frontières, sous la forme de surfaces. Au deuxième niveau, ces surfaces sont subdivisées en polygones, segments et sommets (Figure 1.1-B). Nous verrons dans ce chapitre comment étendre la notion de G-Carte, afin de permettre cette représentation *hiérarchique* des modèles 3D. Les structures que nous allons décrire sont également valables en dimension supérieure. Ce modèle peut aussi trouver des applications intéressantes dans le cadre de calculs intensifs en topologie, comme la compression et l'optimisation de maillages, ou encore l'édition multi-résolution. L'enseignement de notions mathématiques abstraites, telles que les notions d'orientabilité et de partition cellulaire, représente un autre champ applicatif de ce modèle, car il fournit une manière intuitive de représenter graphiquement ces notions.



A B

Figure 1.1: *Modélisation de couches géologiques en utilisant des G-Cartes hiérarchiques. A: modèle du sous-sol présenté en vue "éclatée", montrant les différentes régions volumiques qui le composent; B: ce modèle est structuré hiérarchiquement, en une 3-G-Carte permettant de représenter les relations volumiques liant les régions 3D, et une 2-G-Carte pour discrétiser les surfaces frontières des régions 3D en polygones. La discrétisation de l'une des surfaces est mise en évidence.*

La base de données d'un modeleur géométrique peut être considérée sous deux principaux aspects. La *géométrie*, aussi appelée *plongement*, désigne la position et la forme, dans l'espace 3D, des objets représentés (voir la deuxième partie). La *topologie combinatoire* correspond à la façon dont les objets sont décomposés en primitives, et comment ces primitives sont connectées entre elles. Ce dernier aspect est un champ théorique qui a récemment suscité beaucoup d'intérêt, car des méthodes telles que la compression de maillages [Dee95, TGHL98], l'optimisation de maillages [HDD+93, PH97] ou encore la modélisation multi-résolution [ZSS97, KCVS98] requièrent une représentation topologique efficace. Dans cette partie, l'un de nos objectifs est de montrer comment la modélisation géométrique pourrait bénéficier de notions empruntées à la topologie algébrique [Ago76, Mas67, Mau96], une branche moderne des mathématiques. Ainsi, nous utiliserons ici la notion de *Carte Généralisée* (G-Carte), définie dans [Lie94], comme une manière efficace de représenter la topologie d'objets de dimension arbitraire (surfaces, volumes, hyper-volumes ...). Ces objets peuvent être discrétisés en primitives ayant un nombre quelconque de côtés et de faces. Comme nous le verrons, les complexités des G-Cartes en espace et en temps sont les mêmes que celles des structures plus classiques [Ber96, Ber97], alors que les G-Cartes fournissent plus de flexibilité et un domaine de représentation plus large.

Les travaux précédents liés à ce modèle ont concerné sa définition mathématique [Lie94], ainsi que sa spécification formelle informatique [BDFL93], ce qui a permis à Bertrand de spécifier et d'implanter un premier modeleur expérimental [BDFL93]. Ces travaux insistent surtout sur les implications de ce modèle en termes de topologie combinatoire, spécification formelle et géométrie algorithmique, mais restent difficilement accessibles à la communauté scientifique ayant une culture plus orientée vers la C.A.O. Pour cette raison, cette représentation est encore peu répandue, même si elle offre beaucoup d'avantages, comme nous allons le montrer plus loin. Un de nos objectifs dans cette partie est alors de montrer les connections possibles entre le domaine de la topologie algébrique et celui de la modélisation géométrique, en montrant comment les G-Cartes se positionnent par rapport à des approches plus classiques. Nous montrerons également comment utiliser cette représentation pour implanter un noyau de modeleur pouvant être utilisé dans le cadre d'applications industrielles.

Dans la première section, nous présentons un historique de la modélisation à base topologique. La Section 1.2 introduit la notion de G-Cartes. Nous en verrons une implantation possible dans la Section 1.2.3, tirant partie des possibilités de *généricité* offertes par les langages modernes de programmation. Les algorithmes de base (Section 1.3) ainsi que les tests de validité permettant de vérifier la cohérence des objets (Section 1.4) seront alors présentés. Nous introduirons ensuite la notion de G-Carte hiérarchique dans la Section 1.5, qui va nous permettre une représentation efficace dédiée à la modélisation 3D.

1.1 Fondements théoriques

Avant de présenter la notion de G-Carte, nous donnons ici un historique des différents modèles qui ont été définis en modélisation géométrique. Dans ce contexte, différentes familles de modèles peuvent être utilisées pour représenter la topologie des objets. Les objets sont ainsi discrétisés sous forme d'éléments, appelés des *cellules* . Une base de données topologique correspond alors à cet ensemble de cellules, complété par des structures de données permettant de retrouver efficacement l'ensemble des cellules en contact avec une cellule donnée. Ce type d'opérations va jouer un rôle très important pour les algorithmes d'édition de maillage, ou encore pour les calculs numériques, tels que le lissage variationnel ou la méthode de paramétrisation contrainte, que nous présenterons dans les Chapitres 2 et 3 respectivement.

Le premier modèle introduit a été la représentation *fil de fer*[1], où les objets sont représentés par un ensemble de sommets connectés par des côtés. Cette représentation a été rapidement abandonnée car le même modèle fil de fer peut

[1] wire frame

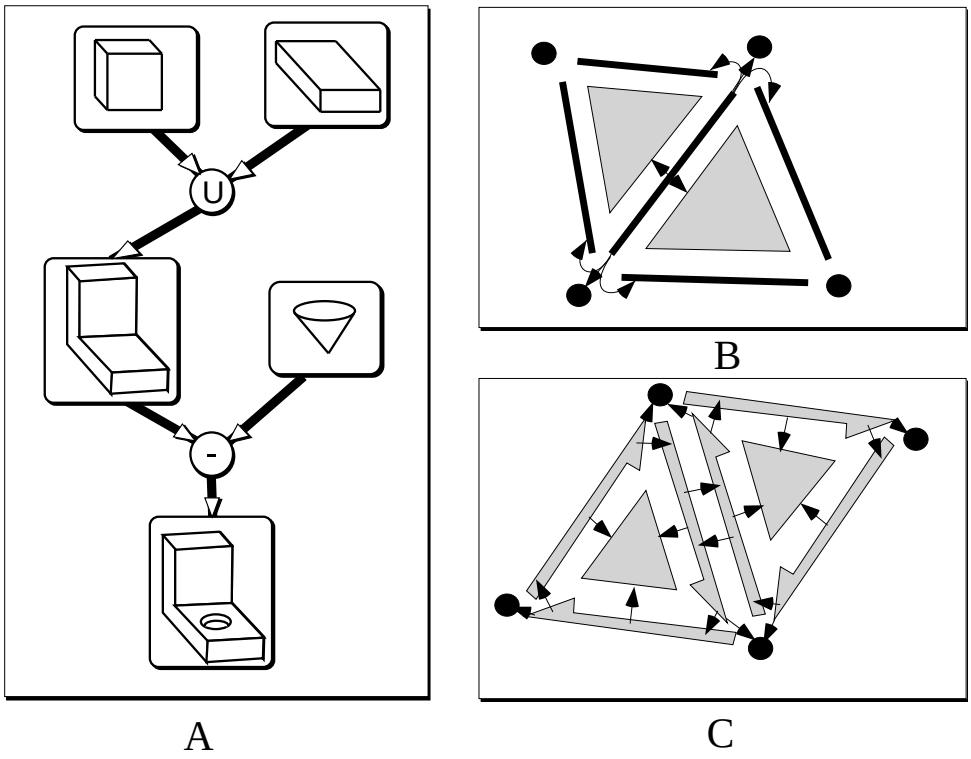


Figure 1.2: *Différentes représentations topologiques. A: Graphe CSG; B: Arêtes ailées (Winged Edge); C: Demi-arêtes (Weiler et DCEL).*

correspondre à plusieurs objets distincts (autrement dit, cette représentation est ambigüe).

Une autre famille de méthodes, connue sous le nom de géométrie constructive solide[2] (CSG) [Req82], définit les objets comme le résultat d'un ensemble d'opérations (union, intersection, différence), appelées *opérations booléennes*, hiérarchiquement appliquées à un ensemble de formes simples appelées des *primitives*. Comme le montre la figure 1.2-A, les objets sont alors stockés sous la forme d'arbres, où les noeuds internes correspondent aux opérations, et les noeuds terminaux représentent les primitives. Dans l'exemple de la Figure 1.2-A, les primitives sont deux parallélépipèdes et un cône, et les opérations sont une union et une différence. Il est important de préciser que ce type de représentation ne définit pas explicitement l'ensemble des éléments (sommets, côtés, faces et volumes) qui compose l'objet, puisque l'objet est le résultat d'un ensemble d'opérations. Il est donc difficile d'attacher des informations à ces éléments lorsque cette représentation est utilisée.

[2]constructive solid geometry

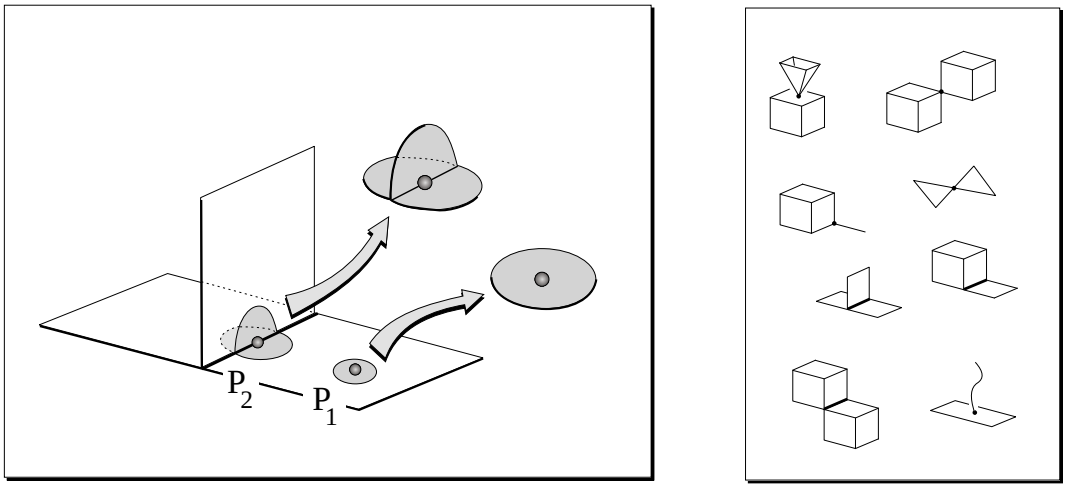


Figure 1.3: *Une variété est un ensemble de points dont les voisinages sont homéomorphe à des disques. A: les voisinages de* $\mathbf{P}_1$ *sont homéomorphes à des 2-disques, donc la surface est variété en* $\mathbf{P}_1$*. En revanche, ce n'est pas le cas des voisinages de* $\mathbf{P}_2$*, donc la surface n'est pas une variété. B: Exemples d'objets qui ne sont pas des variétés, à cause des zones dessinées en gras.*

Dans les représentations par *énumération spatiale*, telles que la notion d'*arbre octal* (ou *octree*) [Mea82] et ses dérivés, un objet est représenté comme un ensemble de cellules connectées les unes aux autres, remplissant l'intérieur de l'objet. Ceci fournit une représentation des objets plus directe que l'approche CSG, mais le domaine de représentation est souvent limité à un type particulier de discrétisation et de connectivité.

Les modèles dits *B-Rep* (Boundary Représentation), ou encore représentation par frontières, consistent à représenter les objets par une discrétisation de leurs bords. La représentation à base d'*arêtes ailées*[3] [Bau75] a été l'une des premières structures de cette famille. Dans cette représentation, l'information est structurée autour des côtés. Chaque côté stocke deux pointeurs vers ses sommets, et deux pointeurs vers les facettes qui le partagent (ses *ailes*, voir Figure 1.2-B). Un ensemble de quatre autres pointeurs permet de parcourir les côtés connectés à un côté donné. Weiler [Wei85] a amélioré cette structure en proposant de couper les arêtes en demi-arêtes, et d'ajouter des pointeurs afin de diminuer la complexité en temps de certains algorithmes (voir Figure 1.2-C). Cette représentation dite *facette-côté* est bien connue et largement utilisée. Elle a été déclinée sous différentes variantes, connues sous le nom de demi-côtés [Män88, BFH95], ou encore de liste

[3]winged edge representation

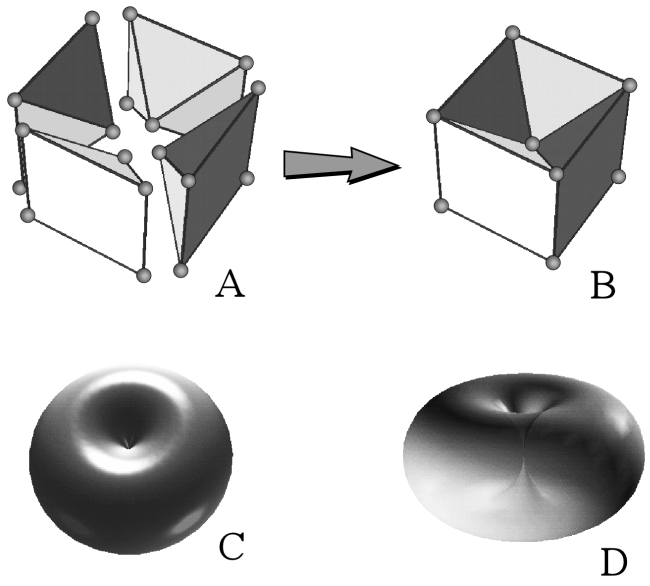


Figure 1.4: *Les G-Cartes permettent de représenter une classe d'objets appellés des quasi-variétés. En dimension 2 (à savoir pour des surfaces), cette classe est équivalente aux variétés. Les différences commencent à apparaître pour des volumes. Ainsi, l'assemblage de quatre pyramides à base carrée se touchant en leur sommet (A) forme un objet volumique (B) qui n'est pas une variété, à cause du point central dont les voisinages ne sont pas équivalents à une boule. Cet objet, ainsi qu'un tore dont le rayon central est nul (C,D) sont des quasi-variétés, pouvant être décrites comme l'assemblage de n-cellules le long de* $n-1$*-cellules.*

doublement chaînée de côtés (DCEL)[4]. C'est cette représentation qui est utilisée dans le projet CGAL [CGA] pour représenter des polyèdres [5]. Il est également possible de considérer cette structure comme une variante de la représentation dite *quad edge* [GS85], où le primal et le dual d'une triangulation sont représentés en même temps. Du point de vue de la topologie combinatoire, la représentation DCEL est équivalente à la notion de *carte combinatoire*, décrite en 1960 par Edmonds [Edm60]. Les cartes combinatoires sont des structures bien définies mathématiquement, qui permettent de représenter des subdivisions de surfaces orientables. Elles ont ensuite été étendues par Lienhardt [Lie88] pour représenter des volumes. Plusieurs représentations ont ensuite été décrites, en se fondant sur

[4]doubly connected edge list

[5]Comme indiqué dans [Ket98], cette représentation ne doit pas être confondue avec celle décrite dans [MP78], de même nom, mais de nature différente

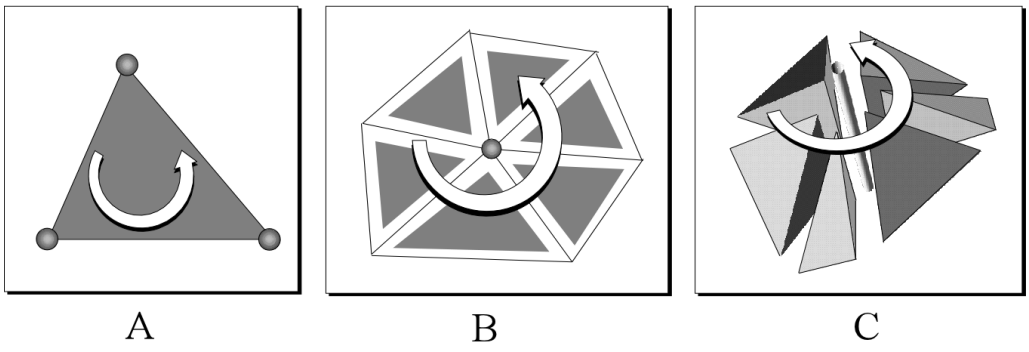


Figure 1.5: *Dans un objet discrétisé en sommets, segments, polygones, polyèdres . . . , les éléments sont* **ordonnés** *autour d'éléments de dimension différente. Ainsi, les sommets d'un polygone sont ordonnés autour du polygone (A), les polygones d'une surface sont ordonnés autour des sommets (B), et dans un solide, les polyèdres sont ordonnés autour des arêtes (C).*

l'idée que la manière dont certains éléments sont ordonnés autour d'éléments de dimension inférieure définit la structure combinatoire du modèle (voir Figure 1.5). Cette idée apparaît de manière informelle dans [AFF85b, FF88], où les objets sont représentés par des hyper-graphes (c'est-à-dire des graphes possédant des multi-arcs, liant un nombre arbitraire de sommets du graphe). Dans cette dernière représentation, les facettes incidentes à un sommet donné (en *éventail*) sont **ordonnées** autour de ce sommet (Figure 1.5-A). De manière similaire, Weiler a étendu sa structure de demi-arêtes afin de représenter des surfaces non-variété, dans [Wei86], où il introduit la représentation par *arêtes radiales* (les notions de variété et de non-variété sont rappelées Figure 1.3). Les arêtes non-variété sont alors définies par un ensemble de facettes **ordonnées** autour de leur côté commun. Toutefois, même si Weiler a pu prouver formellement un ensemble de propriétés de sa représentation, il lui manque toujours un formalisme mathématique unifié pour cette notion d'ordre. Ce manque d'unification se traduit par un très grand nombre de types de structures (onze structures différentes !) devant être définies pour représenter les objets. De plus, toutes les relations entre ces structures doivent être maintenues cohérentes par les algorithmes qui les modifient, ce qui rend la mise au point de nouveaux algorithmes très difficile et source d'erreurs.

Les représentations par arêtes ailées et par arêtes radiales [Wei86] ont été beaucoup utilisées dans l'industrie, sous leur forme originelle, ou sous des formes dérivées. Toutefois, ces structures de complexité croissante ont engendré des applications monolithiques de très grande taille, qui deviennent de plus en plus difficiles à maintenir. Ce problème provient d'un manque de formalisme pour

caractériser la structure algébrique sous-jacente. Afin de s'affranchir de ces problèmes, Brisson a étudié dans [Bri89] la manière dont les éléments sont ordonnés autour d'éléments de dimensions inférieures (voir Figure 1.5), ce qui lui a permis de définir la structure combinatoire dite *tuples de cellules*[6]. Comme nous le montrerons dans la section suivante, l'avantage de ces approches est qu'elles ne requièrent qu'un seul type d'élément ainsi qu'un seul type d'opérateur pour définir la structure combinatoire de n'importe quelle variété. Toutefois, cette étude était incomplète, car Brisson l'a arbitrairement limitée à des variétés formées par l'assemblage d'éléments homéomorphes à des disques. Afin de compléter cette étude, Lienhardt [Lie94] a défini une structure purement combinatoire, appelée G-Carte[7], que nous décrivons dans la section suivante. En utilisant ce formalisme, il lui a été possible de prouver plusieurs propriétés combinatoires des tuples de cellules, et surtout de montrer qu'ils sont en bijection avec une classe d'objets appelés des *quasi-variétés cellulaires* (voir Figure 1.4). Cette classe d'objet est plus large que ce que Brisson avait supposé. Ce type d'approches combinatoires définit une autre classe de représentations, différente de la famille B-Rep, appelée *modèles ordonnés*. Ces représentations sont également appelées *modèles cellulaires* car chaque type de cellule (sommets, arêtes, polygones, polyèdres ...) joue un rôle équivalent dans la définition du modèle. Comme nous le montrerons plus loin, nous pouvons considérer que les modèles cellulaires sont une généralisation des modèles B-Rep (voir Figure 1.11 plus loin): chaque élément est défini par une discrétisation de son bord en éléments de dimension inférieure.

La littérature disponible traitant de ce sujet [Lie88, Lie94, BDFL93, EL92] introduit ce type de représentation en termes de topologie combinatoire, en se fondant sur des notions abstraites telles que les ensembles simpliciaux et la triangulation barycentrique, peu connues dans le domaine de la C.A.O. et de la modélisation géométrique. C'est pourquoi ce type de représentation a été peu utilisé dans l'industrie. Pour cette raison, nous allons présenter dans la prochaine section ces notions d'une manière plus intuitive, et nous allons montrer comment les G-Cartes et les algorithmes associés peuvent être implantés facilement.

1.2 Partitions cellulaires

Dans cette section, nous donnons une vision intuitive de la notion de G-Cartes. Alors que les travaux antérieurs définissent les G-Cartes à partir des ensembles simpliciaux et de la triangulation barycentrique, notre point de départ sera ici la notion de graphe d'incidence, plus répandue dans le domaine de la modélisation 3D.

[6] appelée également "cell-tuple" dans la littérature anglo-saxonne.
[7] Le terme "G-Map" est également employé.

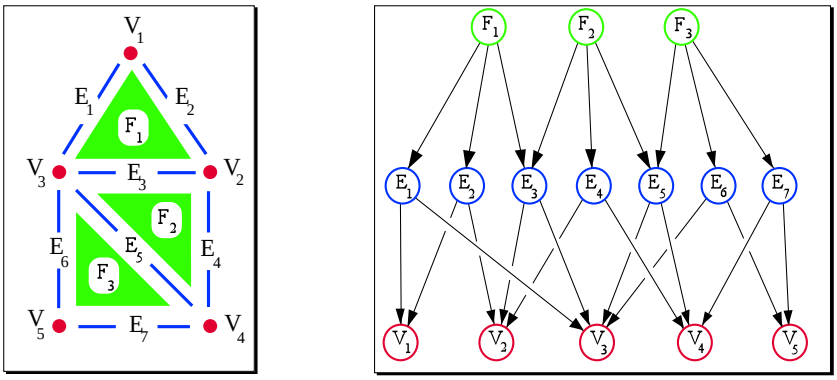


Figure 1.6: *A: Décomposition cellulaire d'un objet simple ; B: Graphe d'incidence associé.*

1.2.1 Notion de graphe d'incidence et tuples de cellules

Nous considérons ici que les modèles sont discrétisés en ensembles ouverts de dimensions $0 \leq k \leq N$ variées, que nous appellerons des *k-cellules*, où $k = dim(c)$ dénote la dimension de la cellule c et où N correspond à la dimension de l'objet à représenter. Ainsi, une 0-cellule est un sommet, une 1-cellule est une arête, une 2-cellule un polygone, une 3-cellule un polyèdre ... Dans [Bri89], une k-cellule est définie comme un ensemble ouvert homéomorphe à un k-disque, alors que Lienhardt a montré dans [Lie94] qu'une plus grande classe de k-cellules peut être utilisée. Une partition d'un objet en cellules est appelée une *partition cellulaire* de l'objet. Une représentation topologique est censée représenter toutes les connections liant les cellules à l'intérieur d'une partition cellulaire, autrement dit les relations d'incidence définies ainsi: Une k-cellule c_1 est dite *incidente* à une $(k-1)$-cellule c_2 si $c_2 \subset \partial c_1$, où ∂c_1 dénote le bord de la cellule c_1. Par exemple, sur la Figure 1.6, la 2-cellule $\mathbf{F}_1$ est incidente aux trois 1-cellules $\mathbf{E}_1$, $\mathbf{E}_2$ et $\mathbf{E}_3$. Dans ce qui suit, $c_1 \mathcal{I} c_2$ dénote que la cellule c_1 est incidente à la cellule c_2. Le graphe d'incidence d'une partition cellulaire est défini comme un graphe orienté, dont les noeuds correspondent aux cellules, et dont chaque arc orienté relie une k-cellule aux $k-1$-cellules auxquelles elle est incidente.

Certaines représentations se fondent directement sur une représentation du graphe d'incidence. Malheureusement, quand ces représentations sont utilisées, retrouver toutes les $(k+1)$-cellules incidentes à une k-cellule donnée nécessite un parcours complet de toute la structure. Il est tout de même possible de compléter la structure afin d'obtenir une implantation efficace de cette opération, mais ceci nécessite des structures à taille variable au niveau de chaque noeud, pour pouvoir parcourir la relation d'incidence dans le sens inverse.

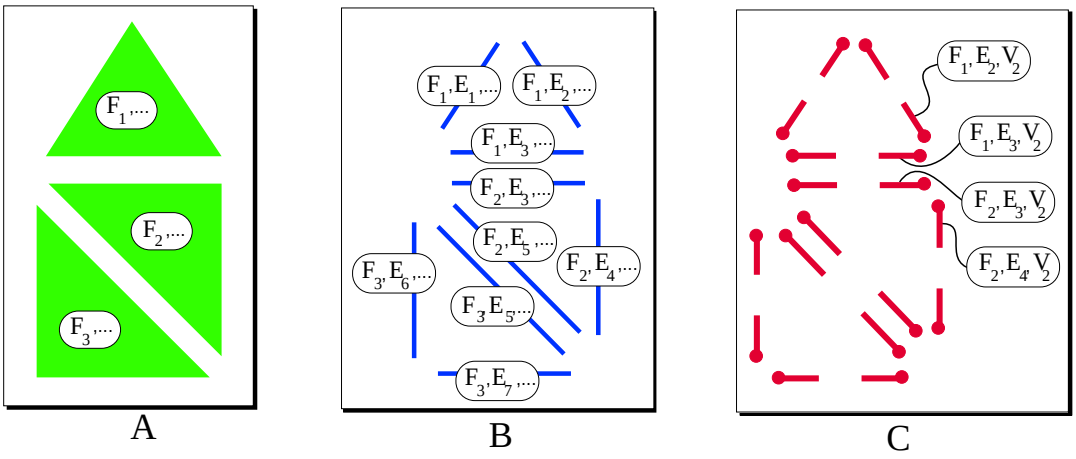


Figure 1.7: *Construction itérative des tuples de cellules engendrés par la partition cellulaire de la Figure 1.6. A: Chaque triangle correspond à l'ensemble des tuples de cellules commençant respectivement par* $\mathbf{F}_1$, $\mathbf{F}_2$ *et* $\mathbf{F_3}$. *B: Si l'on décompose une étape plus loin, on obtient les bâtonnets dont chacun correspond à l'ensemble des tuples de cellules qui commencent par un triangle donné et une arête donnée. C: La dernière étape permet de discriminer tous les tuples de cellules, chacun d'entre eux étant symbolisé par une "tête d'épingle".*

Plutôt que de se focaliser sur les relations liant les cellules, Brisson [Bri89] a introduit la notion de *tuples de cellules*, qui permet de définir une représentation plus élégante de l'information topologique. De plus, comme nous le montrons plus loin, des structures en dimension arbitraire, telles que des surfaces, volumes, hyper-volumes ..., peuvent être définies par cette approche. Les travaux précédents en dimension arbitraire [PBCF93] sont limités à la représentation de complexes simpliciaux, et ne fournissent pas toute l'information topologique. Dans ce modèle, seules les adjacences entre les simplexes de plus haute dimension sont représentées, alors que les tuples de cellules permettent de représenter totalement la topologie de partitions cellulaires.

Un tuple de cellules $\mathcal{C}$ est défini dans [Bri89] comme une séquence ordonnée de cellules $(c_N, c_{N-1}, \ldots, c_1, c_0)$ de dimensions décroissantes, telles que $\forall\ 1 \leq i \leq N$, $c_i\ \mathcal{I}\ c_{i-1}$, où N correspond à la dimension de l'objet considéré. Autrement dit, un tuple de cellules correspond à un chemin dans le graphe d'incidence, ou encore à une séquence de noeuds du graphe de la Figure 1.6-B, connectés par des flèches. Par exemple, pour un objet de dimension 2 tel que celui que nous montrons sur la Figure 1.6-A, un tuple de cellules correspond à un sommet *vu depuis* une arête *vue depuis* un triangle. Comme nous le montrons Figure 1.7, il est possible de construire itérativement une représentation graphique de tous les tuples de cellules engendrés par une partition cellulaire, ce qui peut faciliter la compréhension et la manipulation de cette notion. Nous les représentons Figure 1.7-C comme des "têtes d'épingles" (le segment n'est présent que pour rappeler la forme des

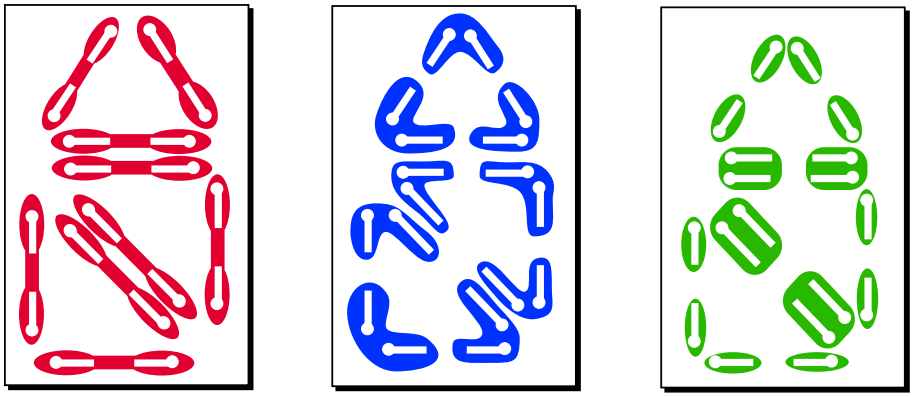


Figure 1.8: *Classes d'équivalence relativement aux relations d'adjacence $\mathcal{A}_0$ (Figure A), $\mathcal{A}_1$ (Figure B) et $\mathcal{A}_2$ (Figure C). Ces classes contiennent soit un seul, soit deux élements, à condition que l'objet considéré soit une variété.*

polygones sous-jacents, nous devrions en toute rigueur représenter uniquement les points).

Ces tuples de cellules sont liés par des relations définies plus loin, que nous allons utiliser pour étudier leur structure combinatoire. Deux tuples de cellules $\mathcal{C}$ et $\mathcal{C}'$ sont dits *i-adjacents*, ce que nous noterons $\mathcal{C}\ \mathcal{A}_i\ \mathcal{C}'$, s'ils vérifient la relation suivante:

$$\mathcal{C}\ \mathcal{A}_i\ \mathcal{C}' \quad \Longleftrightarrow \quad \forall\ j\ /\ 0 \leq j \neq i \leq N,\ c_j = c'_j$$

Autrement dit, deux tuples de cellules sont i-adjacents s'ils correspondent à des chemins dans le graphe d'incidence traversant les mêmes cellules, sauf au niveau i où ils peuvent différer. Par exemple, $(\mathbf{F}_1, \mathbf{E}_1, \mathbf{V}_1)$ et $(\mathbf{F}_1, \mathbf{E}_1, \mathbf{V}_3)$ sont 0-adjacent; $(\mathbf{F}_1, \mathbf{E}_1, \mathbf{V}_1)$ et $(\mathbf{F}_1, \mathbf{E}_2, \mathbf{V}_1)$ sont 1-adjacent; $(\mathbf{F}_1, \mathbf{E}_3, \mathbf{V}_3)$ et $(\mathbf{F}_2, \mathbf{E}_3, \mathbf{V}_3)$ sont 2-adjacent. Ces $N+1$ relations d'adjacences sont des relations d'équivalence (il est facile de vérifier qu'elles sont symétriques, réflexives et transitives). Elles définissent ainsi des classes d'équivalences qui partitionnent l'ensemble des tuples de cellules. Chacune de ces classes, que nous montrons Figure 1.8, contient soit un seul, soit deux tuples de cellules. Brisson a montré dans [Bri89] que c'est toujours le cas pour des partitions cellulaires de variétés. Il définit alors un ensemble de $N+1$ opérateurs appelés $switch_i$, mais nous préfèrerons plutôt introduire les involutions α_i, qui leur sont fondamentalement similaires, mais sont mieux caractérisées formellement. Ces notions définies, nous pouvons alors introduire la gestion de G-Carte comme une façon de décrire des partitions cellulaires en dimension arbitraire.

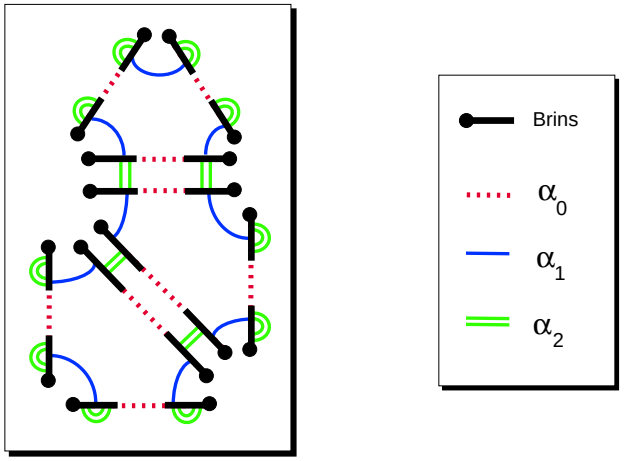


Figure 1.9: *Exemple de G-Carte. Les "têtes d'épingles" correspondent aux brins de la G-Carte. Les involutions α_0, α_1 and α_2 sont symbolisées respectivement par des lignes en pointillés, des lignes et des doubles lignes. Chaque couple de brins connectés par une involution α_i correspond à une classe d'équivalence relativement à une relation d'adjacence $\mathcal{A}_i$ (voir Figure 1.8). Les brins du bord sont connectés à eux-mêmes par l'involution α_2, puisqu'ils sont "orphelins" dans leur classe d'équivalences relative à $\mathcal{A}_2$.*

1.2.2 Structure combinatoire des partitions cellulaires

Dans [Lie94], Lienhardt introduit la notion de cartes généralisées du point de vue de la topologie combinatoire abstraite, ce qui est nécessaire pour définir des théorèmes et des propriétés de cette représentation. Ici, nous préférons prendre comme point de départ la notion de tuple de cellule définie auparavant, qui permet de donner une vue plus intuitive des G-Cartes.

Chaque relation $\mathcal{A}_i$ partitionne l'ensemble des tuples de cellules en sous-ensembles contenant soit un seul, soit deux tuples de cellules (voir Figure 1.8). Ceci suggère de représenter une partition cellulaire par un ensemble d'éléments abstraits, appelés des *brins*[8], correspondant aux tuples de cellules. Ces brins sont munis d'un ensemble de $N+1$ fonctions α_i, définies à partir des relations $\mathcal{A}_i$. Ainsi, pour notre exemple simple, la fonction α_0 connecte chaque paire de brins mises en évidence sur la Figure 1.8-A, i.e. pour une telle paire de brins (d_1, d_2), $\alpha_0(d_1) = d_2$ et inversement, $\alpha_0(d_2) = d_1$. De la même manière, la fonction α_1 connecte chaque paire de brins de la Figure 1.8-B, et la fonction α_2 connecte chaque paire de brins de la Figure 1.8-C. Les brins "orphelins" du bord sont des points fixes pour α_2, c'est-à-dire que pour un tel brin d, $\alpha_2(d) = d$. Nous donnons une représentation graphique d'une G-Carte Figure 1.9.

[8] Dans la terminologie anglaise, ces éléments sont appelés *darts*.

À partir de cette définition, il est clair que les fonctions α_i sont des involutions, i.e. des fonctions telles que $\alpha_i(\alpha_i(d)) = d$ pour tout d. De plus, la contrainte suivante doit être honorée [Lie94]. Nous montrerons plus loin à quoi cette contrainte correspond (voir Figure 1.16 plus loin).

$$\forall\ j\ /\ 0 \leq i < i+2 \leq j \leq N, \quad \alpha_i\ o\ \alpha_j \text{ est une involution.} \tag{1.1}$$

En résumé, une N-G-Carte $\{\mathcal{D}, (\alpha_0, \ldots, \alpha_N)\}$ est définie comme un ensemble $\mathcal{D}$ d'éléments abstraits appelés des brins, muni d'un ensemble de $N+1$ involutions α_i satisfaisant l'Équation 1.1. Sur la Figure 1.9, nous montrons un exemple de 2-G-Carte, représenté en utilisant pour les brins le même symbolisme que pour les tuples de cellules. Nous avons introduit ici les G-Cartes d'un point de vue intuitif, en utilisant la notion de tuple de cellules pour les définir. Il convient de préciser que les G-Cartes sont en réalité des structures combinatoires purement abstraites, définies par l'Équation 1.1, sans nécessiter une quelconque référence aux propriétés des partitions cellulaires pour leur définition. Ainsi, tandis que Brisson [Bri89] a limité arbitrairement le domaine de représentation des tuples de cellules à des variétés composées de cellules homéomorphes à des disques, Lienhardt a démontré que les G-Cartes sont en correspondance bi-univoque avec une classe d'objets plus grande, appelée *quasi-variétés cellulaires*. Son étude fournit une caractérisation complète pour ce type de représentations. Les détails concernant la preuve ainsi que la définition des quasi-variétés cellulaires sont donnés dans [Lie94].

À première vue, la structure de G-Carte apparaît très différente de notre objectif initial, à savoir la représentation des partitions cellulaires. La première question qui peut se poser est comment retrouver les cellules de l'objet. Autrement dit, en partant d'un brin d donné, comment retrouver tous les brins correspondant à la même i-cellule ? Ceci peut être réalisé facilement en considérant une G-Carte comme un graphe valué, où les noeuds du graphe correspondent aux brins de la G-Carte, et où chaque couple de brins connectés par une involution α_i engendre un arc i-valué dans le graphe entre les deux noeuds concernés. Dans ce contexte, $< \alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_k} > (d)$ dénote la composante connexe du graphe obtenue en partant du brin d et en ne traversant que les arcs valués par α_{i_1}, α_{i_2}, ..., ou α_{i_k}.

Il nous parait utile de rappeler maintenant ce que signifie traverser une involution α_i à partir d'un brin d. Supposons que le brin d est tel que $\alpha_i(d) \neq d$ (autrement dit, que d n'est pas un brin du bord). Le brin d correspond à une 0-cellule c_0 *vue depuis* une 1-cellule c_1 ... *vue depuis* une i-cellule c_i ... *vue depuis* une n-cellule c_n. Le brin $\alpha_i(d)$ correspond à la 0-cellule c_0 *vue depuis* la 1-cellule c_1 ... *vue depuis* une i-cellule $c'_i \neq c_i$ ... *vue depuis* la n-cellule c_n. En d'autres termes, $\alpha_i(d)$ correspond aux mêmes cellules que d, sauf au niveau i où elles peuvent différer. Ainsi, comme le montre la Figure 1.10, nous pouvons retrouver tous

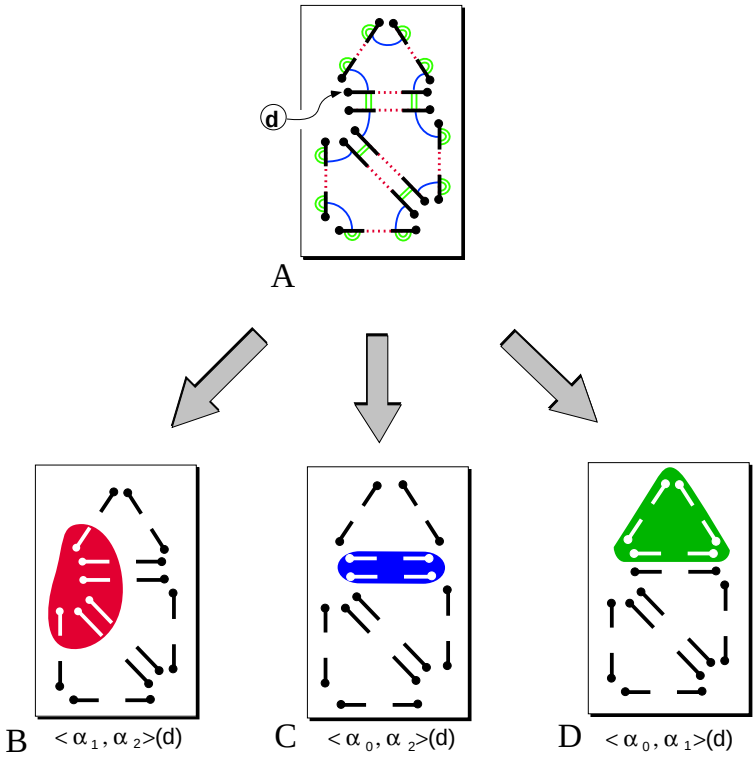


Figure 1.10: *La notion d'orbite permet de retrouver les cellules à l'intérieur d'une G-Carte. En partant d'un brin d donné (voir A), l'objectif est de retrouver tous les brins correspondant respectivement à la même* 0*-cellule (sommet),* 1*-cellule (arête) et* 2*-cellule (facette) que d. Ces ensembles de brins sont donnés par les orbites* $<\not\alpha_0>(d) = <\alpha_1,\alpha_2>(d)$, $<\not\alpha_1>(d) = <\alpha_0,\alpha_2>(d)$ *et* $<\not\alpha_2>(d) = <\alpha_0,\alpha_1>(d)$ *que l'on peut voir sur B,C,D, respectivement. Ces orbites seront utilisées plus loin comme "bloc de base" pour définir des algorithmes.*

les brins qui correspondent à la même i-cellule qu'un brin d en parcourant toutes les involutions α_i qui permettent de rester dans la même i-cellule que d, i.e. tous les α_j pour $j \neq i$ (puisque α_i change de i-cellule). Nous noterons $<\not\alpha_i>(d)$ l'orbite $<\alpha_0,\ldots\alpha_{i-1},\alpha_{i+1},\ldots\alpha_n>(d)$, qui permet de retrouver tous les brins correspondant à la même i-cellule qu'un brin d donné. Cette notation signifie que l'on parcourt récursivement depuis d toutes les involutions α_j **sauf** α_i.

La prochaine section montre comment les G-Cartes peuvent être implantées, et comment leur structure hautement générique peut bénéficier de langages supportant la programmation générique, tels que ANSI C++ [Str97, SL95]. Nous présenterons alors quelques opérations permettant de construire et de modifier ces structures.

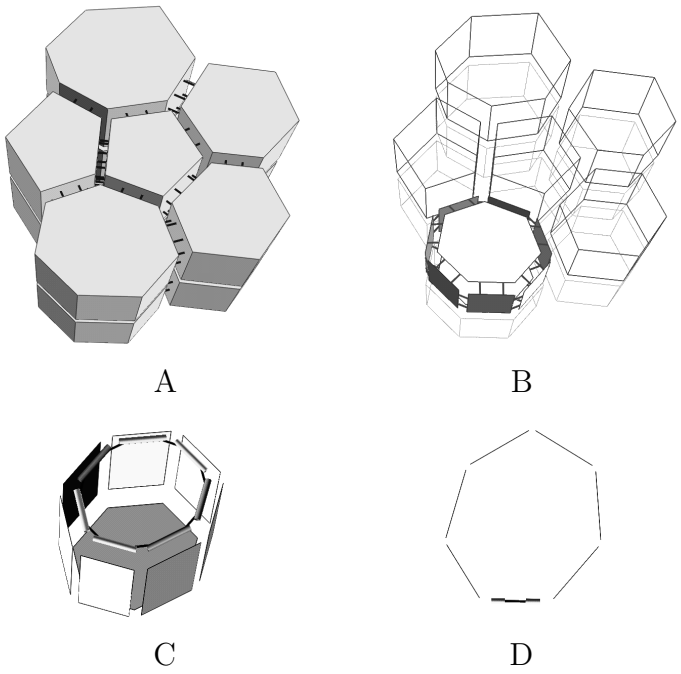


Figure 1.11: *La représentation par G-Cartes peut être considérée comme une généralisation de la représentation par frontières. A: Un objet volumique est défini comme l'assemblage de polyèdres élémentaires; B: Chaque polyèdre est défini par sa frontière, représentée par des faces connectées entre elles; C: Chaque face est représentée par sa frontière, consistant en un ensemble de segments inter-connectés; D: Chaque segment correspond à une paire de brins, correspondant à ses deux extrémités.*

1.2.3 Implantation des G-Cartes

Avant d'aller plus loin, et afin de faciliter la compréhension des algorithmes, il nous parait nécessaire d'évoquer rapidement quelques aspects liés à l'implantation des G-Cartes. Les algorithmes associés aux G-Cartes peuvent être facilement exprimés sous la forme de spécifications formelles, du fait de l'origine mathématique des G-Cartes. C'est cette présentation qui est la plus élégante, et qui permet d'appliquer des méthodes de preuves (éventuellement automatiques) pour tester ces algorithmes. Dans notre cas, nous avons fait le choix de présenter les algorithmes dans un style de programmation impérative. Aborder correctement les aspects liés à la spécification formelle des algorithmes aurait nécessité un développement trop important qui dépasse le cadre de ce mémoire.

Plusieurs implantations des G-Cartes ont déjà été réalisées dans des langages de programmation divers, tels que C, Smalltalk ou ML (voir par exemple [BDFL93, Fuc97, EL92]). Ainsi, en langage C, nous pouvons représenter une G-Carte sous la forme d'une liste chaînée de structures `Brin` détaillées ci-dessous, où le terme *plongement* désigne toute structure que l'utilisateur souhaiterait at-

```
struct Brin {
    Brin* alpha[N+1] ;
    Plongement* plong[N+1] ;
    bool marque ;
    bool clef_de_cell[N+1] ;
} ;
```

Figure 1.12: *Structure associée aux brins pour implanter les G-Cartes. Le membre* `alpha[]` *représente les involutions* α_i*. Les membres* `plong[]` *et* `clef_de_cell[]` *permettent d'attacher des informations aux cellules. Le booléen* `marque` *est utilisé par certaines opérations de parcours.*

tacher aux i-cellules. Par exemple, nous associerons à chaque 0-cellule (sommet) un enregistrement contenant ses coordonnées dans l'espace.

Dans chaque structure de brin, nous stockons $N+1$ pointeurs qui correspondent aux involutions α_i. Les informations de plongement, à savoir les informations attachées aux i-cellules, sont aussi référencées par les pointeurs `plong` de la structure. Tous les brins correspondant à une i-cellule donnée (c'est-à-dire les brins d'une orbite $<\not\alpha_i>$, voir Figure 1.10), pointent vers les données qui correspondent à la cellule. Pour certaines applications, il est probable que l'utilisateur ne souhaite associer des informations qu'aux sommets (par exemple leurs coordonnées x,y,z, des vecteurs normaux, des coordonnées de texture u,v ...). Dans ce cas, le tableau de pointeurs `plong[]` peut être remplacé par un seul pointeur. Toutefois, dans le cas général, il peut être utile de pouvoir attacher des informations à des cellules arbitraires (non seulement aux sommets, mais aussi aux arêtes, côtés, polygones, polyèdres ...), tels que des couleurs, ou encore des coefficients d'éléments finis.

Le bloc de base pour les algorithmes associés aux G-Cartes est le parcours d'une orbite donnée à partir d'un brin d donné. Autrement dit, nous souhaitons obtenir à partir d'un brin d donné l'ensemble des brins que l'on peut atteindre en ne traversant qu'un ensemble donné d'involutions α_i. Ce parcours correspond à un algorithme de graphes classique. Il peut être facilement implanté en utilisant une pile et en marquant les brins au fur et à mesure du parcours (pour ne pas passer au même endroit plusieurs fois). Le marquage est réalisé en positionnant le membre `marque` des structures `Brin`. L'algorithme correspondant peut s'implanter comme une fonction prenant en argument la liste des involutions alpha à parcourir, et un terme ACTION. Ce terme ACTION dénote l'action à effectuer pour chaque brin de l'orbite (on peut le considérer comme un pointeur de fonctions passé en argument). En pratique, il est plus efficace de rendre cet algorithme compatible avec les itérateurs STL [SL95, MS96, Str97], ce qui per-

met d'éviter le coût supplémentaire lié au pointeur de fonction. Par exemple, en utilisant la librairie STL, une liste se parcourt de la manière suivante :

```
list<int> liste ;
...
...
for(list<int>::iterator i = liste.begin(); i != liste.end(); i++) {
    int courant = *i ;
    ...
}
```

Une orbite est alors implantée comme une classe patron[9] `Orbit_n < int ALPHA1, int ALPHA2, ...int ALPHAn >`, compatible avec cette notion de conteneur et d'itérateur STL. Par exemple, le parcours de orbite $< \alpha_0, \alpha_1, \alpha_2 >$ à partir d'un brin d s'écrira alors de la manière suivante :

```
Orbit_3<0,1,2> orbit(d) ;
for(Orbit_3<0,1,2>::iterator i = orbit.begin(); i != orbit.end(); i++) {
    Brin courant = *i ;
    ...
}
```

Ceci permet d'utiliser tels quels les patrons d'algorithmes de la librairie STL. Cette écriture peut également faciliter un interfaçage avec la librairie CGAL [CGA], implantant un certain nombre d'algorithme géométriques. En effet, cette librairie est écrite d'une manière générique, dans le même style de programmation que la STL.

Plutôt que de donner ici l'expression en C++ de l'itérateur STL, nous donnons ici la forme plus lisible, prenant en argument un pointeur ACTION. Il est relativement facile de traduire cet algorithme en un itérateur STL, en surchargeant l'opérateur ++.

[9] template class

```
parcourir(debut: Brin, α_{i_1}, α_{i_2}, ..., α_{i_k} : entier)
   S : Pile ;
   marquer(debut) ;
   empiler(S,debut) ;
   tant que non vide(S)
      Brin d = depiler(S) ;
      ACTION(d) ;
      pour j de 1 à k
         si non marqué(α_{i_j}(d))
            marquer(α_{i_j}(d))
            empiler(S, α_{i_j}(d))
         fin si
      fin pour
   fin tant que
fin parcourir
```

Cet algorithme permet de parcourir efficacement toute orbite d'une G-Carte. Toutefois, suivant les relations entre les involutions α_i à parcourir, des algorithmes plus simples qu'un parcours général de graphe avec marquage peuvent être utilisés. La Figure 1.13 résume les différents cas possibles pour les itérateurs dans des G-Cartes de dimension inférieure ou égale à 3.

- Les itérateurs les plus simples correspondent à des orbites dont le nombre de brins est borné. Ces orbites bornées correspondent aux cas triviaux, et au cas $< \alpha_i, \alpha_{j\geq i+2} >$ qui parcourt 4 brins au maximum. Ceci peut se démontrer aisément, en utilisant la définition des G-Cartes (Équation 1.1). Pour l'exemple des surfaces, une telle orbite correspond aux brins représentant le même côté.

- Les orbites de la forme $< \alpha_i, \alpha_{i+1} >$ sont ordonnées, et peuvent être parcourues à l'aide d'une simple boucle (elles correspondent à la structure d'ordre mentionnée dans [Bri89] et illustrée sur la Figure 1.5). Dans le cas d'une surface (voir Figure 1.10), les brins définissant un polygone correspondent à une orbite $< \alpha_0, \alpha_1 >$, et les brins correspondant à un sommet (formant des "éventails") correspondent aux orbites $< \alpha_1, \alpha_2 >$.

- En dimension 3, les orbites $< \alpha_0, \alpha_1, \alpha_3 >$ et $< \alpha_0, \alpha_2, \alpha_3 >$, qui correspondent respectivement aux faces et aux arêtes, peuvent s'exprimer par un double parcours de boucle, sans nécessiter de marquage.

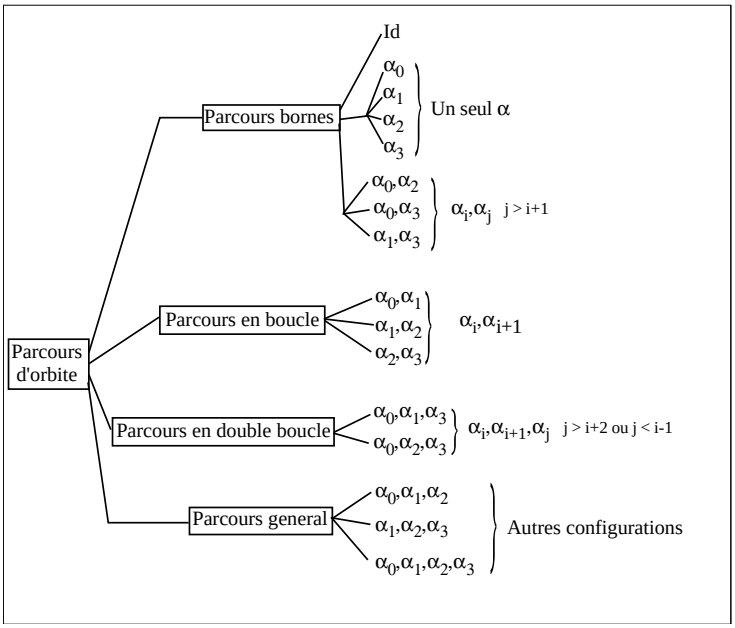


Figure 1.13: *Suivant les relations entre les involutions α_i à parcourir, des algorithmes plus simple qu'un parcours général de graphe avec marquage peuvent être utilisés.*

Dans le contexte d'une implantation en ANSI C++, il est possible d'implanter ces parcours spécifiques et de les rendre transparents du point de vue du code client, en utilisant le mécanisme de *spécialisation partielle* [Str97].

Cette classification reste valable en dimension supérieure, mais elle omet certains cas intéressants. Par exemple, en dimension 4, les orbites de la forme $< \alpha_0, \alpha_2, \alpha_4 >$ contiennent 8 brins au maximum. Dans notre cas, l'étude en dimension 3 reste suffisante, pour représenter des subdivisions de $\mathbb{R}^3$. Il est à noter que le noyau du modeleur fonctionne tout de même en dimension supérieure, même si l'implantation optimum des parcours n'est pas toujours utilisée. Ces parcours optimum pourront être ajoutés par la suite, sous forme de spécialisations partielles, si le besoin s'en fait ressentir.

Ces parcours nous permettent donc de retrouver facilement tous les brins correspondant à une cellule donnée. Nous allons à présent montrer comment ces parcours permettent de gérer les informations associées aux cellules (les *plongements*). En ce qui concerne ces plongements, nous devons gérer correctement la mémoire associée, en donnant la même durée de vie à un plongement qu'à la cellule qui lui correspond. La première idée pourrait être d'utiliser une méthode de comptage de références, ce qui revient à maintenir dans chaque plongement un entier indiquant le nombre de brins qui référencent le plongement considéré. Quand ce nombre atteint zéro, ceci indique que le plongement en question doit être

désalloué. En pratique, non seulement cette méthode augmente la consommation mémoire, mais aussi ralentit-elle beaucoup les algorithmes. Il est beaucoup plus efficace de marquer un brin de chaque cellule comme sa *clef*, définie comme le brin "responsable" de la gestion du plongement associé à la cellule [BDFL93]. D'un point de vue pratique, nous pourrons facilement compresser les différents booléens d'un brin dans un seul mot, chacun d'entre eux étant représenté par un bit de ce mot. Les algorithmes présentés dans la section suivante utilisent cette notion pour gérer efficacement les plongements.

1.3 Opérations définies sur les G-Cartes

Les G-Cartes peuvent être construites à partir d'un jeu d'opérations très réduit, comme nous allons le voir dans cette section. Nous utiliserons ensuite ces notions pour définir notre modèle hiérarchique, permettant de représenter efficacement des subdivisions de $\mathbb{R}^3$. Nous montrerons en exemple d'application comment cette structure permet de représenter un modèle géologique.

1.3.1 Opérations de base

Maintenant que nous avons introduit les structures de données et les algorithmes de parcours de base, il est facile de définir un ensemble de primitives pour créer et modifier des G-Cartes. Les opérations fondamentales, indépendantes de la dimension, sont définies à partir de la remarque suivante : comme nous le montrons Figure 1.11, un objet de dimension N représenté par une G-Carte peut être vu comme une collection d'objets de dimension $N-1$ "cousus" le long de leurs cellules de dimension $N-2$. Cette analyse peut être poursuivie récursivement, ces $(N-2)$-cellules étant elles-mêmes représentées par des $(N-3)$-cellules cousues ensemble. Cette analyse s'arrête quand on atteint les 0-cellules, à savoir les sommets. Par exemple, il est possible de considérer ainsi un volume comme un ensemble de polyèdres cousus le long de leurs faces polygonales (Figure 1.11-A), où chaque polyèdre (Figure 1.11-B) est représenté par son bord, composé de polygones cousus le long de leurs arêtes (Figure 1.11-C). Finalement, chaque arête est représentée par ses deux extrémités, sous la forme d'une paire de sommets (Figure 1.11-D), qui correspondent aux brins. Ceci suggère de définir une opération **coudre** ainsi que l'opération inverse **découdre**, permettant de modifier des G-Cartes en assemblant ou en désassemblant des cellules de dimension arbitraire. Nous détaillons plus loin les algorithmes correspondant à ces deux opérations. Nous supposons que le code client fournit les fonctions **copier_plongement**() et **détruire_plongement()**, non détaillées ici, supposées copier et détruire les informations attachées aux i-cellules. Les fonctions **répartir_plongement()**, **trouver_clef_de_cell()** et **partager_ou_copier_plong()** sont des fonctions auxiliaires, détaillées dans l'Annexe 1.7.

L'algorithme correspondant à l'opération **coudre**(d_1, d_2, dim), qui permet d'assembler deux dim-cellules le long d'une $dim-1$-cellule, consiste en deux parcours en parallèle des $(dim-1)$-cellules $<\not\alpha_{dim-1}>(d_1)$ et $<\not\alpha_{dim-1}>(d_2)$ le long desquelles les dim-cellules doivent être assemblées. L'involution α_{dim} est positionnée entre chaque paire de brins (d'_1, d'_2) ainsi parcourue. En même temps, l'algorithme vérifie pour chaque dimension $i < dim$ si des i-cellules ont été fusionnées durant le processus. Dans ce cas, le i-plongement attaché à l'une des deux i-cellules concernées sera détruit.

```
coudre (d1,d2: Brin, dim: entier)
    pour { d'1 dans <¬α_dim-1 > (d1)
         { d'2 dans <¬α_dim-1 > (d2)
        pour i de 0 a dim - 1
            Brin k1 = trouver_clef_de_cell(d'1, i) ;
            Brin k2 = trouver_clef_de_cell(d'2, i) ;
            si k1 ≠ k2
                k2->clef_de_cell[i] = faux ;
                detruire_plongement(k2, i) ;
                repartir_plongement(k2, i, k1->plong[i]) ;
            fin si
        fin pour
        d'1->alpha[dim] = d'2 ;
        d'2->alpha[dim] = d'1 ;
    fin pour
fin coudre
```

Il est clair que pour pouvoir appliquer **coudre** (d_1,d_2,dim) aux deux dim-cellules désignées par d_1 et d_2, ces deux cellules ne doivent pas être déja cousues. De plus, elles doivent être isomorphes entre elles, ce qui est toujours le cas pour des cellules de dimension inférieure à 2. À partir de la dimension 2, deux cellules quelconques ne sont pas toujours isomorphes. Par exemple, il n'est possible de coudre deux polyèdres le long de deux faces que si les faces en question ont le même nombre de sommets. La situation devient encore plus complexe en dimension supérieure, quand des hyper-polyèdres doivent être assemblés entre eux le long de polyèdres. Le test d'isomorphisme entre deux polyèdres devient alors un test général d'isomorphisme de graphe.

Nous détaillons également ci-dessous l'algorithme implantant l'opération **découdre**(d,dim), l'inverse de **coudre**. Dans cet algorithme, la boucle principale consiste en la dissociation des deux brins d_1 et $d_2 = \alpha_{dim}(d_1)$, où d_1 parcourt la $(dim-1)$-cellule $<\not\alpha_{dim}>(d)$. Une fois que d_1 et d_2 ont été dissociés, l'algorithme teste pour chaque $i < dim$ si une i-cellule a été transformée en deux i-cellules. Dans ce cas, l'information attachée aux i-cellules coupées en deux est dupliquée, si nécessaire, grâce à la fonction **partager_ou_copier_plong**, détaillée dans l'Annexe 1.7.

```
découdre (d: Dart, dim: int)
   pour d1 dans < α0, α1, ..., αdim-1 >(d)
      d2 = d1->alpha[dim] ;
      d1->alpha[dim] = d1 ;
      d2->alpha[dim] = d2 ;
      pour i de 0 a dim-1
         partager_ou_copier_plong(d1, d2, i) ;
      fin pour
   fin pour
fin découdre
```

Il est facile de compléter l'ensemble des deux opérations **coudre** et **découdre** avec les opérations simples **créer_sommet** et **détruire_sommet**, permettant respectivement de créer et de détruire un sommet **isolé**. Il a été démontré dans [Lie94] que toute G-Carte peut être créée en utilisant uniquement ces quatre opérations. Toutefois, afin de rendre plus facile la conception de certains algorithmes, il est parfois utile d'étendre cet ensemble d'opérations avec des opérations de plus haut niveau. Par exemple, il est possible d'implanter une opération **détruire_cellule**(d,dim), qui encapsule les appels à **découdre** et à **détruire_sommet** correspondants. Inversement, plusieurs fonctions **créer_cellule** peuvent être implantées, pour créer directement des triangles, des polygones, des tétraèdres ... Il convient de préciser que les complexités en espace et en temps caractérisant la structure de G-Carte ainsi définie sont similaires à celles correspondant aux implantations classiques type demi-arêtes (voir [Lie94]).

1.3.2 Dualité

La notion de dualité est importante en géométrie algorithmique. Comme nous le montrons Figure 1.14, le dual d'une triangulation de Delaunay est un diagramme de Voronoï : une structure qui caractérise les proximités d'un ensemble de points. La définition complète de ces structures dépasse le cadre de ce mémoire,

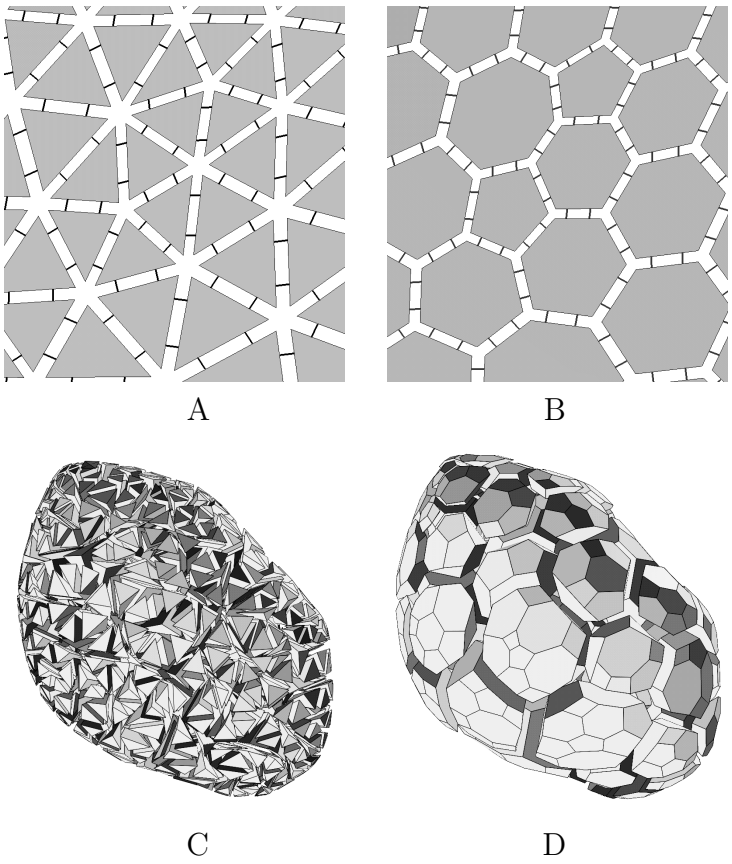


Figure 1.14: *Notion de dualité. Le dual d'une triangulation de Delaunay est obtenu en "basculant" toutes les arêtes, de manière à les rendre orthogonales à leur position initiale. Les sommets du dual sont les centres des cercles circonscrits aux triangles. A: Triangulation de Delaunay; B: Diagramme de Voronoï associé; C: Solide tétraédrisé; D: Diagramme de Voronoï 3D associé (les cellules non bornées du bord ont été supprimées).*

et le lecteur se réfèrera par exemple à [BY95] pour plus de détails. Dans cette section, nous allons voir comment construire à l'aide de G-Cartes le dual d'une triangulation en dimension arbitraire.

La notion de dualité peut être généralisée en dimension arbitraire. Ainsi, le dual d'une tétraédrisation de Delaunay est un diagramme de Voronoï 3D. Comme un diagramme de Voronoï peut contenir des polygones et des polyèdres arbitraires, il est souvent représenté indirectement par la triangulation/tétraédrisation de Delaunay sous-jacente [GS85]. Certaines applications spécifiques, telles que la résolution d'équations aux dérivées partielles, peuvent bénificier des propriétés géométriques des diagrammes de Voronoï. Par exemple, dans [Ver96], une telle structure est utilisée pour simuler l'écoulement du pétrole dans les couches du sous-sol. Les relations de perpendicularité entre le primal et le dual permettent de simplifier l'évaluation des transmitivités entre éléments finis, et d'avoir ainsi un système plus simple à résoudre. Pour de telles applications, il est parfois

nécessaire de modifier localement le maillage dans des zones riches en détails, pour pouvoir y définir un plus grand nombre d'éléments finis. Dans ce cas, les coefficients attachés aux sommets, arêtes, polygones et polyèdres composant l'objet doivent être correctement gérés. Pour cette raison, une représentation directe du diagramme de Voronoï peut être nécessaire. Ceci est possible en utilisant des G-Cartes, puisqu'aucune supposition n'a été faite concernant les polygones/polyèdres à représenter.

```
construire_dual(N-G-Carte M = {D, α0, ..., αN})
  pour d dans D
    si d->clef_de_cell[N]
      detruire_plongement(d, N) ;
      nouv_plong = geometrie_duale(d, N) ;
      repartir_plongement(d, N, nouv_plong) ;
    fin si
  fin pour
  pour d dans D
    pour i de 0 a N
      echanger(d->alpha[i], d->alpha[N-i]) ;
      echanger(d->clef_de_cell[i],
        d->clef_de_cell[N-i]) ;
      echanger(d->plong[i], d->plong[N-i]) ;
    fin pour
  fin pour
fin cconstruire_dual
```

Le dual d'un maillage peut être construit en place par l'algorithme ci-dessus, pour un maillage de dimension arbitraire. L'algorithme consiste en deux étapes. La première étape concerne la géométrie, et associe à chaque N-cellule le centre de la N-sphère circonscrite, calculé à partir de la fonction **géometrie_duale**(d, N) (non détaillée ici). Par exemple, en dimension 2, cette fonction retourne le centre du cercle circonscrit du triangle référencé par d. La seconde étape met à jour la structure combinatoire de la G-Carte, en "renversant" les involutions α_i et les plongements. À noter la simplicité de cet algorithme, bien qu'il travaille en dimension arbitraire. Avec des structures de données plus classiques [Wei85, Wei86], cet algorithme devient beaucoup plus complexe, et ne se généralise pas à des dimensions arbitraires.

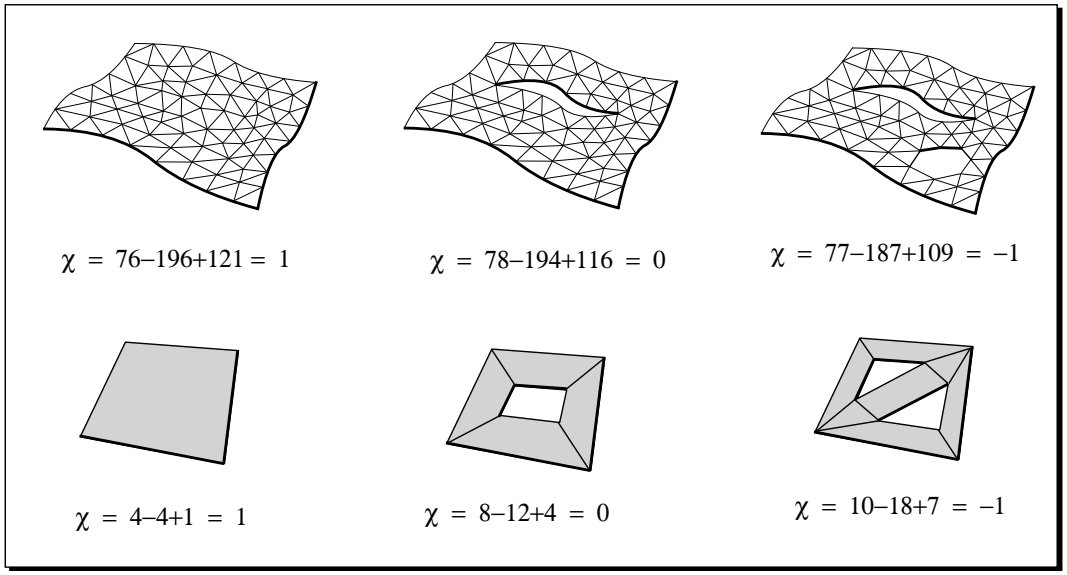


Figure 1.15: *La caractéristique d'Euler $\mathcal{X}$ est un nombre indépendant de la subdivision de l'objet en cellules. Pour des subdivisions de surfaces, elle s'exprime par $\mathcal{X} = S - C + F$, où S,C et F dénotent respectivement le nombre de sommets, de côtés et de faces.*

1.4 Caractérisation des objets cellulaires

En C.A.O., il est très important de pouvoir contrôler la construction des objets géométriques, et d'éviter ainsi de construire des objets invalides. La caractéristique d'Euler-Poincaré évoquée sur la Figure 1.15 est souvent utilisée pour classifier les objets ainsi que leurs opérations associées (voir [Pet85b, Mas67, Mau96] pour plus de détails). Plutôt que de nous intéresser à cette caractéristique, très facilement définissable pour des G-Cartes [Lie94, Ber97], nous montrons dans cette section comment des tests de cohérence plus spécifiques peuvent être définis. Ainsi, les relations combinatoires caractérisant les G-Cartes peuvent être vérifiées pour un objet donné, ce qui donne un moyen facile de vérifier qu'un objet a été correctement construit. De plus, cet ensemble de tests de base peut être étendu, en définissant des conditions plus restrictives [BDFL93]. Ces conditions permettent de discriminer les objets ayant des cellules repliées sur elle-mêmes et/ou les objets ayant des bords hétérogènes et/ou les objets non-orientables. Comme ceci a été montré dans [Lie94], la rigueur du modèle est assurée par construction, puisque l'ensemble des objets valides, muni des opérations de base (**coudre**, **découdre**, **créer_sommet**, **détruire_sommet**), est fermé.

1.4.1 Tests de cohérence de la base de données

Une N-G-Carte $\mathcal{G} = \{\mathcal{D}, \alpha_0, \ldots, \alpha_N\}$ est définie par les conditions 1 et 2, rappelées ci-dessous [Lie94]. La condition 1 vient du fait que les fonctions α_i sont

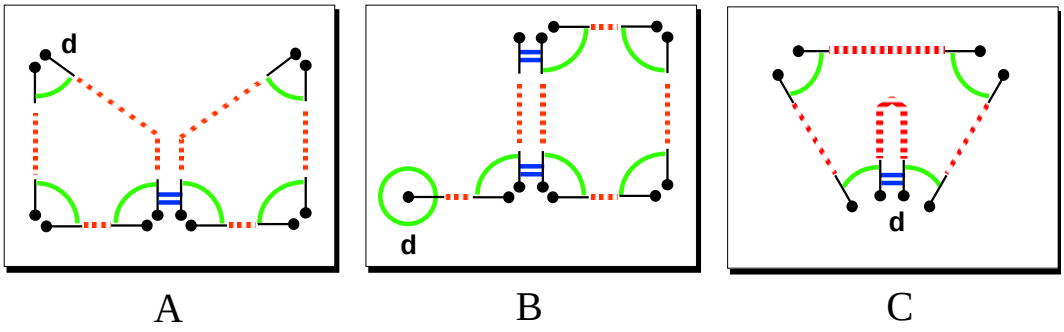


Figure 1.16: *Configurations interdites. A: Couture incomplète; B: Bords à dimensions hétérogènes; C: Arête repliée sur elle-même.*

définies à partir de relations d'adjacence *symétriques* $\mathcal{A}_i$, comme nous l'avons montré précédemment. La condition 2 caractérise l'opération **coudre**, en empêchant la réalisation de coutures incomplètes. Par exemble, le brin d mis en évidence sur la Figure 1.16-A ne vérifie pas la condition $\{\alpha_0 \; o \; \alpha_2\}^2(d) = d$, qui assure des coutures complètes. Ces deux tests sont facile à implanter sous forme de programme, et permettent de rejeter les objets invalides. Dans certaines circonstances, il peut être souhaitable de réduire la classe des objets représentables, en interdisant certaines configurations [BDFL93]. Ainsi, les bords de dimensions hétérogènes peuvent être évités en imposant la condition 3. Sur la Figure 1.16-B, l'arête pendante[10] peut être détectée, puisque $\alpha_1(d) = d$. L'arête repliée sur elle-même de la Figure 1.16-C est détectée par la condition 4 ($\alpha_0 \; o \; \alpha_2(d) = d$).

1. $\forall 0 \leq i \leq N$, α_i est une involution ;
2. $\forall 0 \leq i < i+2 \leq j \leq N$, la fonction $\alpha_i \; o \; \alpha_j$ est une involution ;
3. $\forall 0 \leq i < N$, la fonction α_i n'a pas de point fixe ;
4. $\forall 0 \leq i < i+2 \leq j \leq N$, la fonction $\alpha_i \; o \; \alpha_j$ n'a pas de point fixe.

Ces différentes conditions sont faciles à tester, et peuvent être vérifiées pour un objet donné en un temps proportionnel au nombre de brins composant cet objet. En d'autres termes, vérifier la validité d'un objet ne coûte pas plus cher que de l'afficher.

[10] ce terme est utilisé dans le domaine de la C.A.O.

1.4.2 Orientabilité

La notion d'orientabilité fournit une autre manière de caractériser les objets modélisés. En modélisation 3D, la construction de surfaces non-orientables n'est pas souhaitée en général. Toutefois, de telles surfaces peuvent être construites dans un modeleur, suite à des erreurs de manipulation, ou encore à des problèmes de précision numériques. Pour cette raison, il est nécessaire de fournir un moyen pour détecter de telles surfaces. De plus, la notion de G-Carte hiérarchiques que nous verrons dans la Section 1.5 se définit à partir de surfaces orientables.

Une surface est dite *orientable* si elle possède deux faces différentes, ce qui est le cas de la plupart des surfaces rencontrées en modélisation 3D. Le ruban de Moebius que nous montrons sur la Figure 1.17 est un exemple souvent cité pour expliquer cette notion. Cette surface est non-orientable, car elle n'a qu'une seule face. Dans le cas où des surfaces non-orientables ne devraient pas être construites, comme dans la plupart des applications industrielles, vérifier si un objet est orientable ou non permet de détecter facilement certains objets incorrects. D'un point de vue pédagogique, la méthode pour détecter des objets non-orientables, décrite plus loin, peut être visualisée graphiquement, ce qui fournit une manière intuitive pour expliquer cette notion. L'enseignement des mathématiques et de l'analyse différentielle, ou encore d'autres domaines mathématiques abstraits, pourrait bénéficier de telles représentations graphiques (se référer par exemple à [LMM95, Gun93, Pet85a]).

Comme nous le montrons Figure 1.17, l'orientabilité d'un objet représenté par une G-Carte peut également être définie à partir de l'"aimantation" de chaque paire de brins appartenant à cet objet. L'"aimantation" d'une G-Carte revient à considérer chaque paire de brin $\{(d, \alpha_0(d))\}$ comme un aimant, ayant un pôle nord et un pôle sud. Si, pour un objet donné, il est possible de choisir de manière consistante pour chacun des brins s'il correspond à un pôle Nord ou à un pôle Sud, alors l'objet est orientable. Choisir des pôles Nord et des pôle Sud revient à partitionner l'ensemble de brins $\mathcal{D}$ en deux sous-ensembles $\mathcal{D}^+$ et $\mathcal{D}^-$. Il est alors possible de formaliser comme suit la loi magnétique, signifiant qu'un brin d'une polarité donnée ne peut être connecté qu'à des brins de polarités opposées :

$$\begin{cases} \forall d \in \mathcal{D}^+,\ \forall 0 \leq i \leq N, \quad \alpha_i(d) \in \mathcal{D}^- \quad \text{ou} \quad \alpha_i(d) = d \\ \\ \text{et reciproquement:} \\ \\ \forall d \in \mathcal{D}^-,\ \forall 0 \leq i \leq N, \quad \alpha_i(d) \in \mathcal{D}^+ \quad \text{ou} \quad \alpha_i(d) = d \end{cases} \tag{1.2}$$

Il est difficile de traduire directement cette condition en un algorithme, puisque la polarité des brins n'est pas donnée *à priori*. Comme cela a été suggéré dans

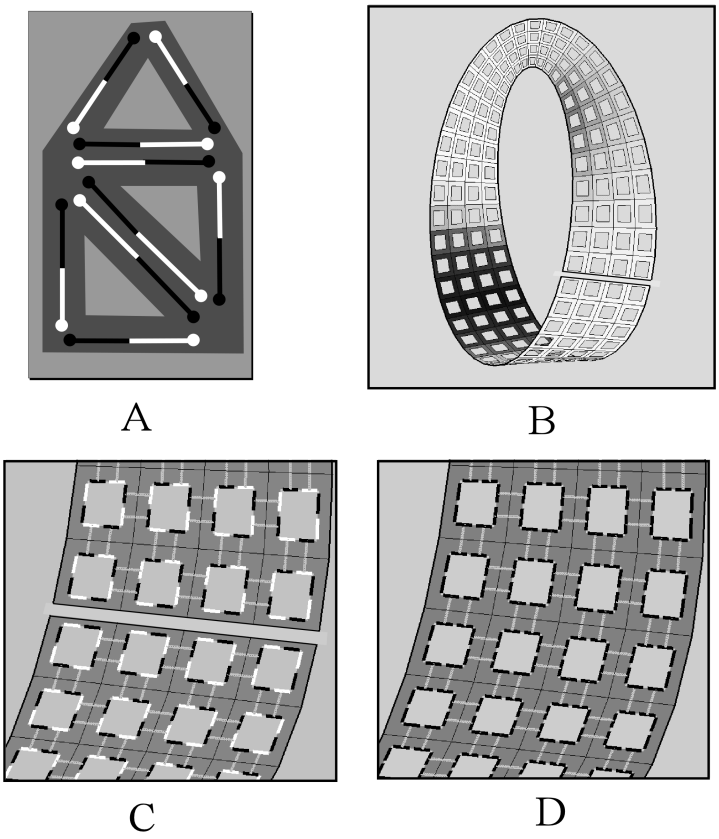


Figure 1.17: *Notion d'orientabilité. A: Dans le cas d'une surface orientable, chaque paire de brins* $\{d, \alpha_0(d)\}$ *peut être considérée comme un "aimant". Si les aimants peuvent être combinés tout en respectant les "lois magnétiques", alors l'objet est orientable. Le ruban de Moebius est un exemple bien connu de surface non-orientable. B: Cette surface a la* **géométrie** *d'un ruban de Moebius, mais pas sa* **topologie**, *puisqu'elle est coupée, donc orientable (C). Nous pouvons voir toutefois que les lois magnétiques ne pourront pas être respectées lorsque nous recoudrons la coupure ; D: Une fois que la coupure est cousue, la surface n'est plus orientable.*

[Lie94], une solution pratique se fonde sur le fait que la polarité d'une composante connexe orientable est totalement déterminée par un seul brin dont la polarité est fixée. Étant donné un brin d, l'orbite $< \alpha_0 \; o \; \alpha_1, \alpha_0 \; o \; \alpha_2, \ldots, \alpha_0 \; o \; \alpha_N > (d)$ permet de retrouver tous les brins ayant la même polarité que d. Si cette orbite parcourt la moitié des brins de la composante connexe de d, alors cette composante connexe est orientable. Dans le cas contraire, cette orbite parcourt la totalité des brins de la composante connexe.

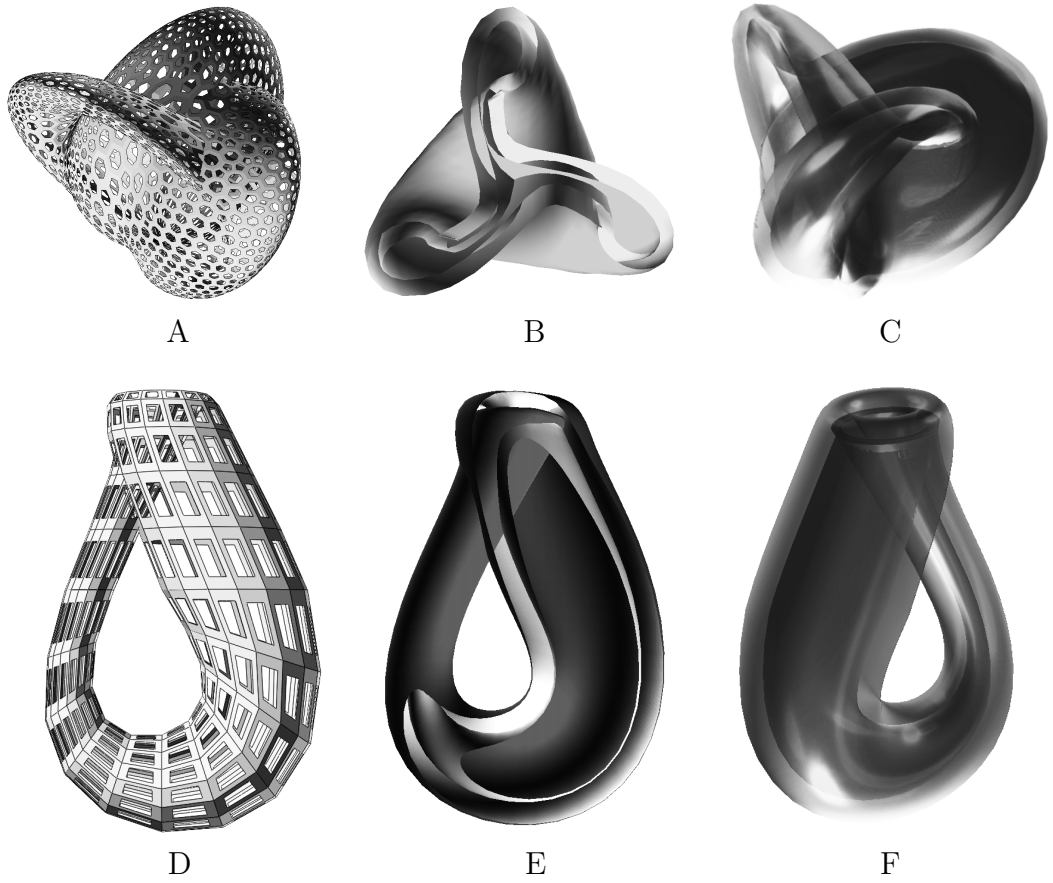


Figure 1.18: *La surface de Boy (A), tout comme la bouteille de Klein (D), est une surface non orientable fermée. B: Les revêtements à double feuillet de ces surfaces (B,C et E,F), affichés ici en coupe, sont des surfaces orientables. Il est possible de montrer que le revêtement de la surface de Boy est homéomorphe à une sphère, et celui de la bouteille de Klein est homéomorphe à un tore. Ils peuvent donc servir de point central au retournement de la sphère et du tore.*

Cette orbite peut être parcourue facilement en utilisant l'algorithme de parcours général donné en Section 1.2.3.

$$\left|\begin{array}{l} \text{soit } \mathcal{G} = \{\mathcal{D}, (\alpha_0, \ldots, \alpha_N)\} \quad \text{une N-G-Carte connexe} \\ \mathcal{G} \text{ est orientable} \\ \qquad \Longleftrightarrow \\ < \alpha_0 \; o \; \alpha_1, \alpha_0 \; o \; \alpha_2, \ldots, \alpha_0 \; o \; \alpha_N > (d) \neq \mathcal{D} \end{array}\right. \tag{1.3}$$

Si nous considérons le graphe dont les sommets sont les brins, et dont les arcs correspondent aux fonctions $\alpha_0 o \alpha_i$, le critère d'orientabilité revient à tester si ce graphe est biparti. Ce test peut facilement être effectué en assignant des "polarités" aux brins. Il est alors possible d'ajouter comme pré-condition à

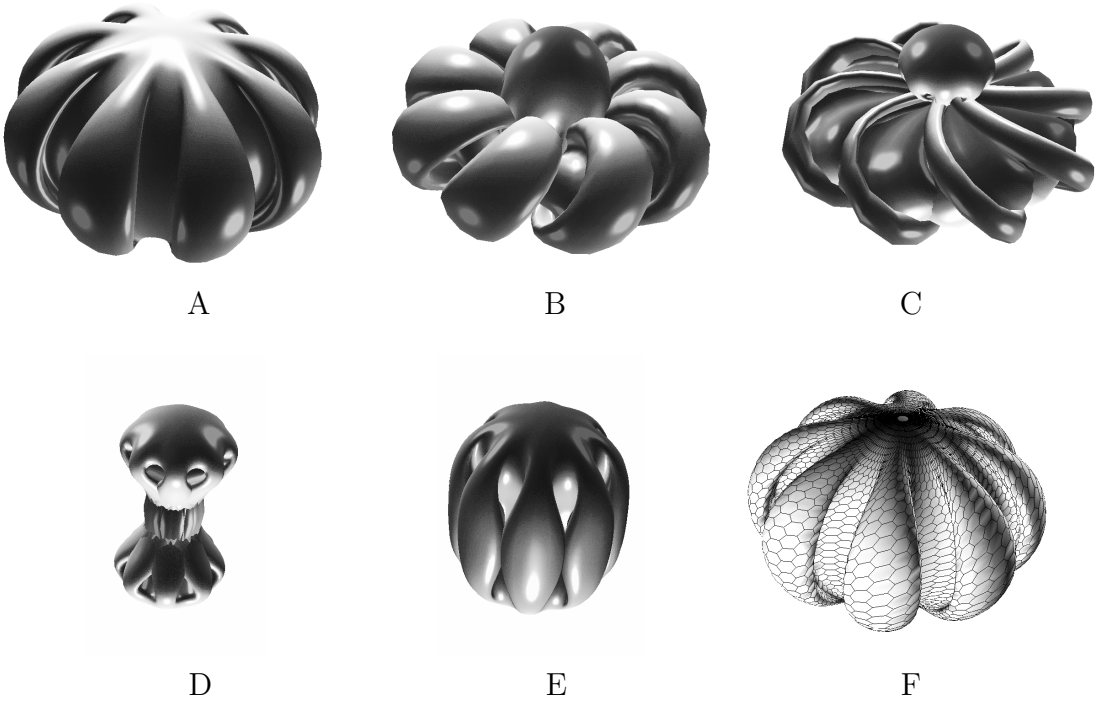


Figure 1.19: *Plusieurs étapes de l'éversion de sphère proposée par Sylvio Lévy et. al. (A-E), et la discrétisation de l'une des surfaces (F). Cette suite d'opérations permet de retourner une sphère, tout en préservant la continuité de la normale pendant l'opération.*

l'opération **coudre** un test qui vérifie que l'on ne coud pas des brins de même polarités.

La notion d'orientabilité est utile en modélisation 3D, car elle va nous permettre d'identifier rapidement certains objets invalides, construits lors d'erreurs de manipulation, ou encore à la suite de problèmes de précision numérique. Cette notion est également liée à certaines curiosités mathématiques, que nous évoquons ici brièvement. Les surfaces en question nous ont également servi à valider certains algorithmes du modeleur. Les mathématiciens se sont intéressés à ce type de surfaces dans le cadre du problème dit du "retournement" de la sphère (également appelé *éversion*). Ce problème est expliqué de manière didactique dans les ouvrages pédagogiques de Jean-Pierre Petit [Pet85a, Pet85b, Pet85c], et un historique de ce problème est donné dans le chapitre 6 de [Fra87]. En partant d'une sphère peinte en rouge à l'intérieur et en bleu à l'extérieur, le problème consiste à obtenir une sphère peinte en bleu à l'intérieur et en rouge à l'extérieur. Les règles du jeu autorisent les surfaces à se traverser, mais interdisent l'apparition d'angles (la normale à la surface devant rester définie en tous points pendant le processus). La possibilité d'une telle opération a été démontrée par Smale [Sma58], en

1958. Depuis, les mathématiciens ont cherché des manières d'atteindre cet objectif, comme Phillips qui a proposé une première "recette" dans [Phi66]. Shapiro a été l'un des premiers à définir une méthode totalement formalisée, qui a été décrite par Francis et Morin dans [FM79, Fra87]. L'opération étant relativement complexe, Max a fait appel au graphisme 3D pour visualiser le processus [Max77].

Ces méthodes ont également été décrites dans [Pet85a, Pet85b] : ainsi, si nous peignons une surface de Boy (en utilisant une seule couleur, puisqu'elle est non-orientable), et que nous retirons ensuite la surface, il ne reste plus que la peinture, qui forme ce qui s'appelle un *revêtement à double feuillet* de la surface de Boy (Figure 1.18-B,C). Cette dernière surface a deux faces : l'une était en contact avec la surface de Boy, et l'autre était en contact avec l'extérieur. De plus, elle est homéomorphe à une sphère, ce qui suggère de l'utiliser comme point central dans le processus de retournement de la sphère. En effet, dans un premier temps, la sphère est transformée en revêtement à double feuillet de la surface de Boy, puis les deux feuillets sont échangés (en se passant au travers). L'opération inverse est ensuite effectuée, transformant la surface en une sphère. D'une manière similaire, Petit [Pet78, Pet79] a également montré que le revêtement à double feuillet (Figure 1.18-E,F) de la bouteille de Klein (Figure 1.18-D) pouvait servir de point central au retournement du tore.

Une autre méthode de retournement de la sphère a été proposé dans [LMM95, LP95], fondé sur une idée différente (voir Figure 1.19). Des plis sont tout d'abord créés sur la sphère (Figure 1.19-A), puis les deux pôles sont "poussés" l'un vers l'autre jusqu'à ce qu'ils se traversent (Figure 1.19-B). Ensuite, les plis subissent une rotation (Figure 1.19-C), et il ne reste plus qu'à "déplier" la surface (Figure 1.19-D,E). Ceci peut paraître bien compliqué, mais la nécessité de préserver la continuité de la normale lors de l'opération impose tous ces "détours". La Figure 1.19-F montre la discrétisation de l'une de ces surfaces, représentée ici à l'aide d'une G-Carte. Ces objets ont été construits à partir du programme décrit dans [LP95] et disponible dans [LT95], adapté de manière à produire des G-Cartes pour notre modeleur.

Sullivan *et. al.* ont proposé dans [SFL98] une approche totalement différente, fondée sur une minimisation de l'énergie de la surface. L'évolution de la surface est alors obtenue automatiquement, par minimisation d'une fonctionnelle d'énergie, à l'aide du programme *Evolver* de Brakke [Bra92b, Bra92a]. Afin de générer des surfaces possédant des propriétés géométriques utiles pour nos problèmes de modélisation 3D, nous proposerons une approche de lissage discret dans le Chapitre 2, similaire à l'approche de Brakke, et offrant la possibilité de prendre en compte des contraintes linéaires.

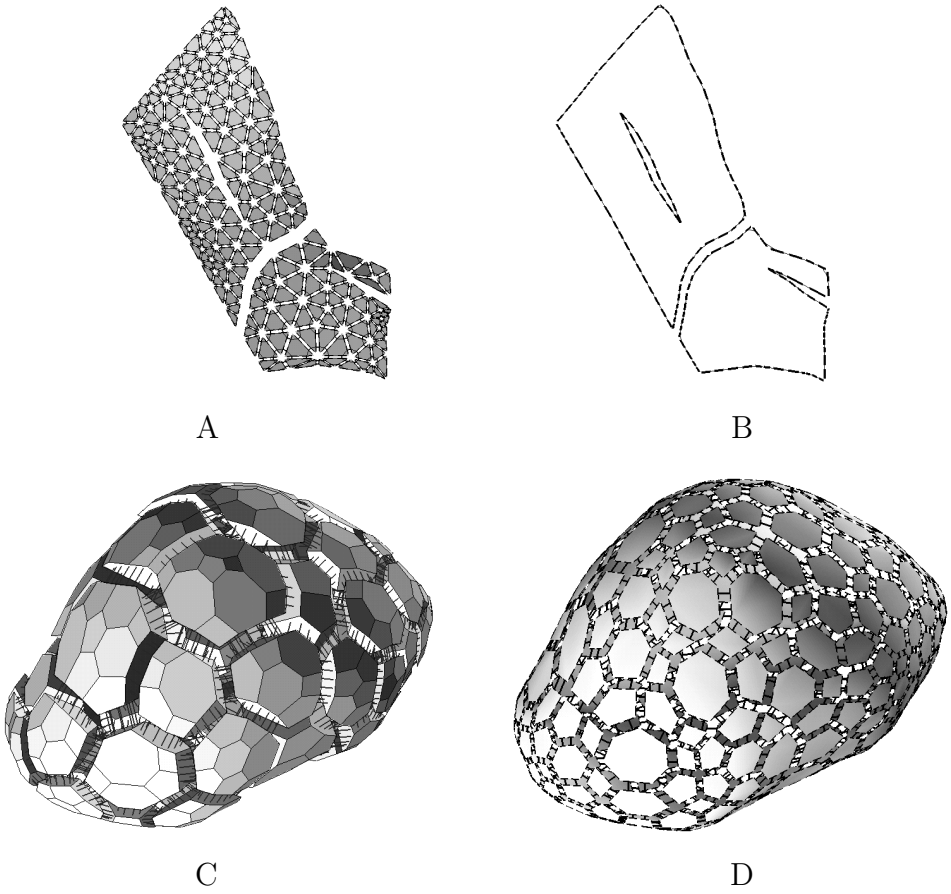


Figure 1.20: *A: Une surface représentée par une 2-G-Carte; B: La carte des bords correspondante; C: un volume discrétisé en polyèdres, les liens volumiques sont affichés; D: le bord du volume.*

1.4.3 Carte des bords

La notion d'orientabilité introduite dans la section précédente offre une manière simple de rejeter une classe d'objets invalides (les objets non-orientables). Les surfaces liées au problème du retournement de la sphère nous ont permis de tester ces méthodes de validation.

Dans certaines circonstances, ces tests ne sont pas suffisant, et le domaine d'application décessite d'autres manières de caractériser les objets construits.

Ainsi, le nombre et la nature des bords d'une surface offrent une manière différente de les classifier, et de détecter certains problèmes liés à la construction des modèles. Par exemple, si certains polygones ne sont pas connectés de manière combinatoire bien que partageant géométriquement un côté, ceci pourra être détecté par la présence d'un bord le long du côté en question.

Comme cela a été montré par Lienhardt dans [Lie94], il est facile d'extraire le bord d'un objet représenté par une N-G-Carte $\mathcal{G} = \{\mathcal{D}, (\alpha_i)_{0 \leq i \leq N}\}$, sous la forme d'une $N-1$-G-Carte $\partial\mathcal{G} = \{\mathcal{D}', (\alpha'_i)_{0 \leq i \leq N-1}\}$, définie par:

$$\begin{cases} & \mathcal{D}' & = & \{d \in \mathcal{D} / \alpha_N(d) = d\} \\ \forall 0 \leq i \leq N-2, \forall d \in \mathcal{D}', & \alpha'_i(d) & = & \alpha_i(d) \\ \forall d \in \mathcal{D}', & \alpha'_{N-1}(d) & = & d'/d' \in < \alpha_N, \alpha_{N-1} > (d) \text{ et } d' \neq d \end{cases} \tag{1.4}$$

Cette notion est indépendante de la dimension des objets considérés et peut donc s'appliquer à des lignes, à des surfaces, à des volumes ... (voir Figure 1.20). En utilisant cette définition, il est possible d'utiliser le même code pour extraire le bord d'une surface (Figure 1.20-A,B) et le bord d'un volume (Figure 1.20-C,D). Cette notion va également nous servir de base pour définir la notion de G-Carte hiérarchique, introduite dans la section suivante.

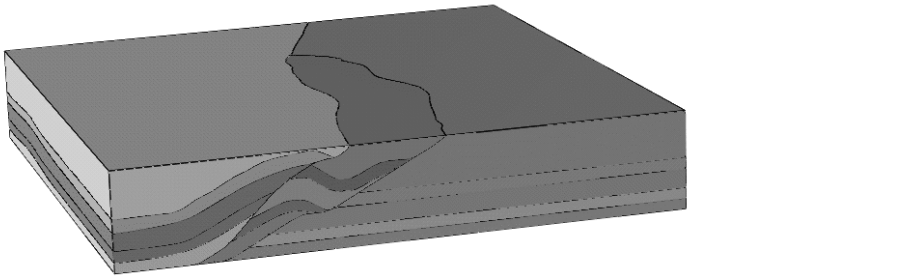
1.5 Cartes Généralisées Hiérarchiques

Notre objectif est de représenter de manière efficace des modèles géologiques du sous-sol. De tels modèles pouvant être considérés comme une partition de l'espace 3D, ils peuvent donc être représentés par des 3-G-Cartes. En utilisant les opérations précédentes ainsi que des algorithmes de plus haut niveau qui les combinent, il est possible de construire des modèles du sous-sol. Néanmoins, les 3-G-Cartes sous leur forme originelle ne s'avèrent pas totalement adaptées à ce type de modélisation, qui présente certaines spécificités :

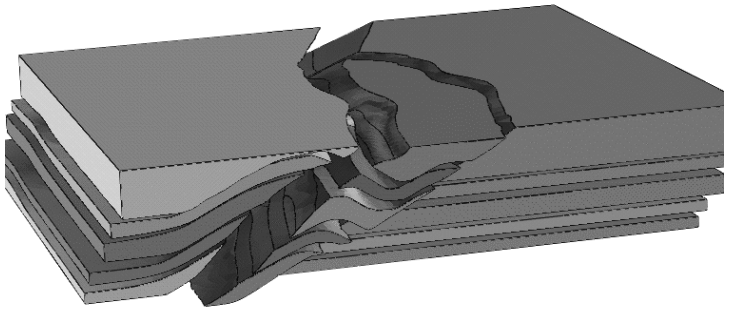
1. Les couches géologiques sont souvent représentées par des polyèdres comportant un très grand nombre de faces, et liés les uns aux autres d'une manière qui présente une forte cohérence spatiale (voir Figure 1.21). Autrement dit, si une face d'un polyèdre P_1 est connectée à un polyèdre P_2, il est probable que les faces voisines soient également connectées à P_2;

2. L'utilisateur va souvent devoir manipuler de grands ensembles de cellules, considérés comme une primitive de plus haut niveau, comme les couches géologiques;

3. Les *interfaces*, surfaces limites séparant deux couches géologiques, jouent un rôle prépondérant dans le processus de modélisation. En effet, à cause de la nature des données, les utilisateurs commencent souvent par construire des surfaces qui représentent ces interfaces, pour ensuite les assembler entre elles afin de partitionner l'espace 3D en couches géologiques. Une approche



A



B

Figure 1.21: *A: Modèle 3D du sous-sol (extrait du modèle EAEG-SEG), décomposé en cellules volumiques correspondant aux couches géologiques; B: Le même modèle, en vue "éclatée". Ces cellules volumiques ont toutes un très grand nombre de faces (de l'ordre de la dizaine de milliers), ce qui va nécessiter un modèle hiérarchique, afin de ne pas dupliquer inutilement les liens volumiques.*

différente à été proposée dans [Nll98], où des volumes sont directement reconstruits à partir du diagramme de Voronoï des points de données.

Les points 1 et 2 suggèrent de regrouper les cellules élémentaires (sommets, segments, polygones ...) en primitives de plus haut niveau, afin de "factoriser" les liens volumiques, et de pouvoir ainsi désigner un grand ensemble d'éléments par une seule primitive de plus haut niveau. Dans ce qui suit, nous allons montrer comment définir un modèle hiérarchique adapté à ce type de besoins, permettant facilement de manipuler à la fois des surfaces et des volumes (point 3).

Dans la littérature, nous avons pu remarquer que les points 1 et 2 ont souvent été présents dans le "cahier des charges" devant être respecté par les modèles topologiques. Ainsi, Ansaldi *et. al.* ont proposé dans [AFF85b, AFF85a] un modèle topologique fondé sur un graphe d'adjacence des faces, et l'ont ensuite enrichi d'une structure hiérarchique dans [FF88, AF86], ceci pour permettre une représentation compatible avec les processus mis en jeu dans le domaine de la

CFAO[11]. Des modèles numériques de terrains ont également été représentés par une structure hiérarchique dans [FP95], à l'aide de triangulations dans lesquelles certains triangles peuvent être munis récursivement d'une sous-triangulation. Nous pouvons également citer les modèles topologiques pour la multi-résolution [ZSS97, KCVS98], fondés sur l'application itérative d'une opération de simplification sur un maillage de départ. Ceci permet d'obtenir une hiérarchie de maillages plus simples, en supprimant itérativement des ensembles de sommets et de triangles. En ce qui concerne les G-Cartes, Bertrand décrit un modèle hiérarchique dans [Ber97], permettant de "cloner" des parties d'objets, sans dupliquer leur représentation en mémoire.

Ces différents modèles sont adaptés aux besoins spécifiques de leur domaine d'application, mais n'offrent pas le type de fonctionalités souhaitées pour nos applications. Pour la plupart, ces modèles hiérarchiques présentent un couplage trop fort entre leur différents niveaux. Avant de présenter notre modèle hiérarchique, nous allons tout d'abord présenter les différentes approches que nous avons considérées, fondées sur la notion de *sous-partition cellulaire*. Nous verrons ensuite comment le modèle de la *délégation de plongement* apporte une plus grande flexibilité.

Sous-partition cellulaire

Une première idée pour définir un modèle hiérarchique fondé sur les G-Cartes pourrait être de considérer la notion de sous-partition cellulaire. Étant donnée une partition cellulaire $\mathcal{P} = \mathcal{C}_0 \cup \mathcal{C}_1 \ldots \cup \mathcal{C}_N$ où les $\mathcal{C}_i$ dénotent les ensembles des cellules de dimension i de la partition, une sous-partition cellulaire $\mathcal{P}' = \mathcal{C}'_0 \cup \mathcal{C}'_1 \ldots \cup \mathcal{C}'_N$ est définie comme une partition cellulaire telle que toute cellule de $\mathcal{C}'_N$ est égale à l'union d'un ensemble de cellules de $\mathcal{P}$:

$$\forall c \in \mathcal{C}'_N, \quad \exists c_1, c_2, \ldots c_k \in \mathcal{P} \quad / \quad c = \bigcup_{i=1}^{k} c_i \tag{1.5}$$

Par exemple, sur la Figure 1.22-A, les traits fins représentent une partition cellulaire $\mathcal{P}$ d'une surface, et les traits gras correspondent à une sous-partition cellulaire $\mathcal{P}'$ de $\mathcal{P}$.

Si la partition $\mathcal{P}$ est représentée par une G-Carte $\mathcal{G} = \{\mathcal{D}, \alpha_0, \alpha_1, \ldots \alpha_N\}$, la sous-partition $\mathcal{P}'$ revient à partitionner l'ensemble $\mathcal{D}$ en k sous-ensembles $\mathcal{D}_1$, $\mathcal{D}_2$, $\ldots \mathcal{D}_k$, chacun d'entre eux correspondant à une N-cellule de $\mathcal{P}'$ (autrement dit à un ensemble de triangles délimité par un trait gras sur la Figure 1.22-A). Il est alors possible de représenter $\mathcal{P}'$ par une G-Carte $\mathcal{G}' = \{\mathcal{D}', \alpha'_0, \alpha'_1, \ldots \alpha'_N\}$, entièrement déterminée à partir de $\mathcal{G}$. L'idée consiste à considérer les G-Cartes des bords des sous-ensembles $\mathcal{D}_1$, $\mathcal{D}_2$, $\ldots \mathcal{D}_N$, et à les relier par une fonction α'_N:

[11] conception et fabrication assistées par ordinateur

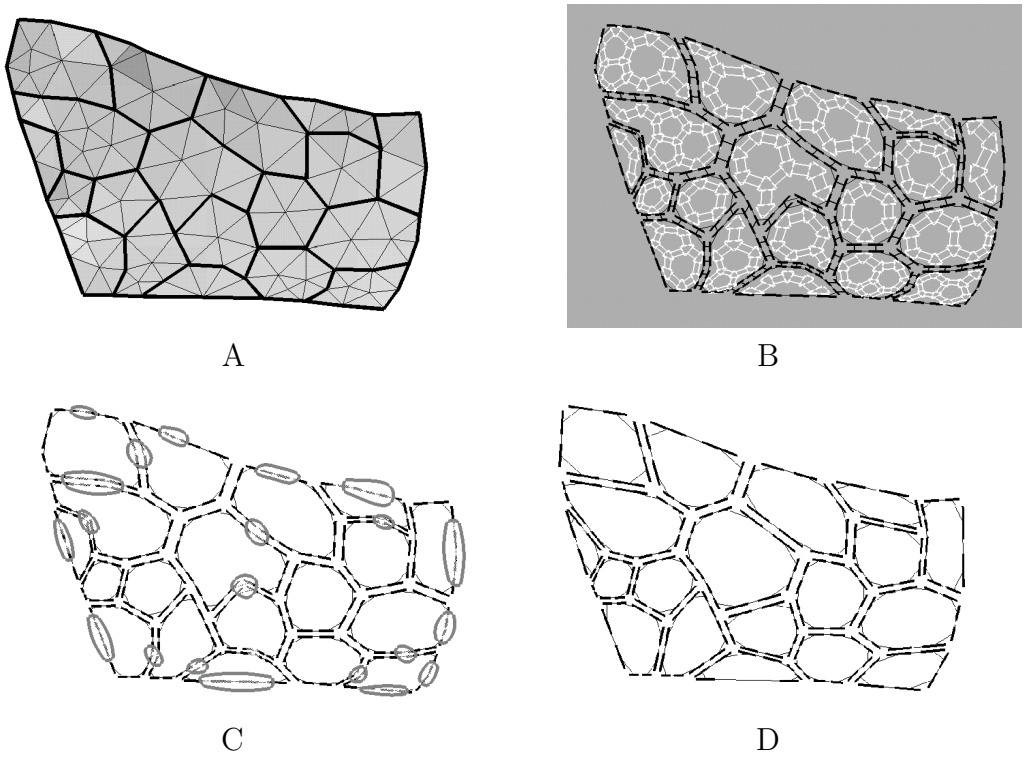


Figure 1.22: *A: Partition cellulaire et sous-partition d'une surface; B: représentation des deux niveaux par deux G-Cartes imbriquées, la G-Carte correspondant à la partition cellulaire étant représentée en blanc, et celle correspondant à la sous-partition étant représentée en noir; C: la G-Carte de plus haut niveau comporte des brins qui pourraient être supprimés (zones grisées); D: simplification de la G-Carte de plus haut niveau.*

$$\left\{\begin{array}{l} \mathcal{D}' = \{d \in \mathcal{D} \quad / \quad \exists i \neq j \quad / \quad d \in \mathcal{D}_i \text{ et } \alpha_N(d) \in \mathcal{D}_j\} \\ \\ \qquad \forall 1 \leq i \leq k, \quad \forall d \in \mathcal{D}_i, \quad \alpha'_{N-1}(d) \;=\; d' \in < \alpha_{N-1}, \alpha_N > (d) \cap \mathcal{D}_i \quad / \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \alpha_N(d') \notin \mathcal{D}_i \text{ et } d' \neq d \\ \\ \forall 0 \leq i \neq N-1 \leq N, \quad \forall d \in \mathcal{D}', \quad \alpha'_i(d) \;=\; \alpha_i(d) \end{array}\right. \tag{1.6}$$

Il convient à présent de vérifier que la structure combinatoire définie par $\mathcal{G}'$ est bien une G-Carte, à savoir qu'elle vérifie bien la relation suivante (voir Section 1.2.2):

$$\left\{\begin{array}{lll} \forall 0 \leq i \leq N, & \alpha_i & \text{est une involution} \\ \\ \forall 0 \leq i < i+2 \leq j \leq N, & \alpha_i \; o \; \alpha_j & \text{est une involution} \end{array}\right. \tag{1.7}$$

Si nous nous intéressons à un des sous-ensembles $\mathcal{D}_i$, muni des fonctions $\tilde{\alpha_0}, \tilde{\alpha_1} \ldots, \tilde{\alpha_N}$:

$$\begin{cases} \forall d \in \mathcal{D}_i, \forall 0 \leq i < N, & \tilde{\alpha_i}(d) = \alpha_i(d) & \\ \forall d \in \mathcal{D}_i, & \tilde{\alpha_N}(d) = \alpha_N(d) & \text{si } \alpha_N(d) \in \mathcal{D}_i \\ & \tilde{\alpha_N}(d) = d & \text{sinon} \end{cases} \tag{1.8}$$

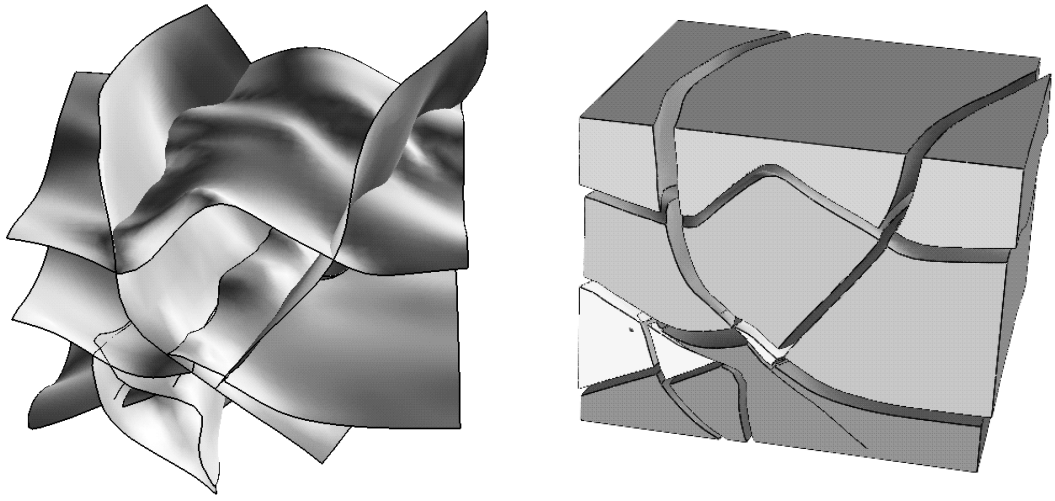
alors il est clair que $\{\mathcal{D}_i, \tilde{\alpha_0}, \tilde{\alpha_1}, \ldots \tilde{\alpha_N}\}$ est une G-Carte, et que $\{\mathcal{D}_i \cap \mathcal{D}', \alpha'_0, \alpha'_1, \ldots \alpha'_{N-1}\}$ est la G-Carte de ses bords (voir la section précédente). La condition est donc vérifiée pour les α'_i avec $0 \leq i \leq N-1$. Il reste alors à vérifier que α_N et $\alpha_i \; o \; \alpha_N$ pour $0 \leq i \leq N-2$ sont bien des involutions. C'est effectivement le cas, puisque α'_i pour $0 \leq i < N-1$ est la restriction de α_i à $\mathcal{D}'$.

La Figure 1.22-B montre la G-Carte $\mathcal{G}$ correspondant à la discrétisation la plus fine (en blanc), et $\mathcal{G}'$, définie par une sous-partition de la surface (en noir).

Cette première approche définit bien un modèle hiérarchique, permettant de désigner des ensembles de cellules par des primitives de plus haut niveau, mais elle impose que la discrétisation des bords des cellules soit la même pour deux niveaux consécutifs (donc pour tous les niveaux). Ceci limite fortement l'intérêt de l'approche, en imposant la représentation de liens volumiques pour tous les polygones présent à la jonction de deux couches géologiques dans un modèle tel que celui de la Figure 1.21.

Bien sûr, il est possible de simplifier la représentation des bords des ensembles $\mathcal{D}_i$, en appliquant itérativement le même processus de simplification à différents ensembles de brins. À chaque étape, ces ensembles de brins comportent tous les brins ayant des orbites isomorphes du point de vue de la partition $(\mathcal{D}_i)$. Nous montrons sur la Figure 1.22 le résultat de cette simplification. Les brins mis en évidence sur la Figure 1.22-A ont été supprimés de la discrétisation (Figure 1.22-B).

Dans [Nll98], Nullans a décrit une méthode pour construire une partition de l'espace à partir de points munis d'un vecteur normal et d'un numéro de région à laquelle ils sont censés appartenir (une telle information est souvent appelée une "couleur", et les points sont alors dits "coloriés"). Notre méthode de représentation d'une sous-partition permet d'extraire d'un tel modèle une représentation des interfaces entre les différentes régions. De plus, il serait également possible de mettre en place des méthodes de reconnaissance d'images dans une telle structure, comme cela a été réalisé par Fiorio dans [Fio95, FG96] dans le cas de grilles régulières (ou plus précisément, d'images). Ce type d'approches



A B

Figure 1.23: *A: horizons et failles, correspondant aux frontières entre couches géologiques. La surface représentée est non-variété dans les zones où plusieurs surfaces se rencontrent à la manière des pages d'un livre. Une représentation directe de cette surface nécéssite des structures de données spécialisées. B: le volume défini comme la partition de l'espace engendrée par cette surface est une variété, donc il peut être représenté par des structures plus simples, par exemple une 3-G-Carte.*

permet bien de traiter les deux premiers points de notre cahier des charges ("factoriser" les liens volumiques, et permettre la désignation de grand ensembles de primitives). Le dernier point n'a toujours pas été traité, à savoir la possibilité de représenter tout d'abord des surfaces, pour ensuite les assembler afin de définir une partition de l'espace 3D. Dans la section suivante, nous allons voir une structure permettant ce type d'interactions avec l'utilisateur.

Délégation de plongement

En C.A.O. comme en géologie numérique, les utilisateurs souhaitent pouvoir définir facilement des partitions de l'espace 3D, correspondant aux couches géologiques. Au cours du processus de modélisation, des surfaces sont tout d'abord construites (Figure 1.23). Les intersections entre ces surfaces sont ensuite calculées, ce qui génère des configurations non-variété dans les zones où plusieurs surfaces se rencontrent, à la manière des pages d'un livre. Les zones non-variété correspondent aux points dont les voisinages ne sont pas homéomorphes à des disques (voir la Figure 1.3 de la Section 1.1). Une représentation directe de cette surface nécessite des structures de données spécialisées, telles que la représentation *arête radiale* de Weiler [Wei86], ou des chaînes de G-Cartes [EL92]. Nous pro-

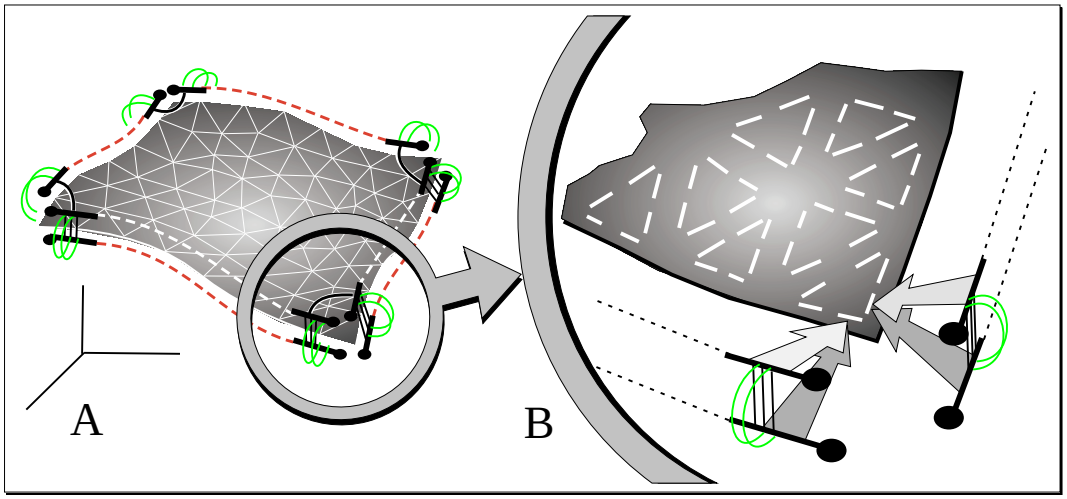


Figure 1.24: *A: Dans une H-G-Carte, chaque objet est muni d'un "cadre", représentant le complémentaire de l'espace dans lequel l'objet est plongé. Ainsi, une surface* $\mathbf{S}$ *représentée par une G-Carte* $\mathcal{G}$ *est munie d'une 3-G-Carte* $\hat{G}$*, représentant* $\mathbb{R}^3 - \mathbf{S}$*. Cette représentation permet de combiner la surface* $\mathbf{S}$ *avec d'autres surfaces, afin de subdiviser l'espace; B: détail des relations mises en jeu au niveau d'un coin. Chaque brin de* $\hat{\mathcal{G}}$ *(en noir) référence celui de* $\mathcal{G}$ *(en blanc) qui lui correspond (flèches).*

posons ici une autre approche, fondée sur la représentation de surfaces non-variété par des relations variété entre des volumes.

Cependant, ces surfaces correspondent en réalité à des frontières entre des volumes, qui représentent des couches géologiques (voir Figure 1.23). Les volumes ainsi délimités sont des variétés, dont la représentation peut être définie à l'aide de structures plus simples que les arêtes radiales. Cette constatation va nous permettre de représenter des relations non-variété entre des surfaces en nous fondant sur le fait que ces surfaces correspondent aux frontières de volumes variété. Par exemple, il est ainsi possible de représenter ce type de configurations par une 3-G-Carte. Si les surfaces sont orientables (ce qui est toujours le cas dans les applications pratiques), chaque surface peut alors être considérée comme une "cloison", pouvant partitionner l'espace. Ainsi, comme le suggère la Figure 1.24-A, nous allons munir chaque surface $\mathbf{S}$ d'une structure, que nous appellerons un *cadre*, représentant l'espace $\mathbb{R}^3 - \mathbf{S}$ entourant la surface. Nous appellerons *treillage* la G-Carte qui représente la subdivision de $\mathbf{S}$. Dans la suite, le terme *cloison* désignera l'ensemble formé par un cadre et par son treillage.

De même, en 2D, une courbe $\mathbf{C}$ peut être munie d'un cadre, correspondant à l'espace $\mathbb{R}^2 - \mathbf{C}$ qui l'entoure. Nous définissons un cadre d'une N-G-Carte $\mathcal{G}$ comme une $N+1$-G-Carte $\hat{\mathcal{G}} = \{\mathcal{D}^+ \cup \mathcal{D}^-, (\hat{\alpha}_i)_{0 \leq i \leq N+1}\}$. Les fonctions $\hat{\alpha}_i$ pour $0 \leq i \leq N$ sont définies de sortes que les deux N-G-Cartes

$\hat{\mathcal{G}}^+ = \{\mathcal{D}^+, (\hat{\alpha}_i)_{0\leq i\leq N-1}\}$ et $\hat{\mathcal{G}}^- = \{\mathcal{D}^-, (\hat{\alpha}_i)_{0\leq i\leq N-1}\}$ soient isomorphes à une même sous-partition de la G-Cartes des bords de $\mathcal{G}$. Les fonctions $\hat{\alpha}_N$ et $\hat{\alpha}_{N-1}$ lient les paires de brins qui se correspondent dans les deux G-Cartes isomorphes $\hat{\mathcal{G}}^+$ et $\hat{\mathcal{G}}^-$. Comme nous le montrons sur la Figure 1.24-B, les brins d'un cadre référencent ceux de son treillage. En quelque sorte, nous pouvons dire que la géométrie d'un cadre est "déléguée" à son treillage, c'est pourquoi nous désignons par *délégation de plongement* ce type de modèles hiérarchiques.

Certains algorithmes vont nécessiter la définition de parcours pour cette structure, permettant de naviguer entre les cadres et les treillages associés. Par exemple, l'algorithme consistant à construire une tétraédrisation d'un espace délimité par un ensemble de cloisons nécessite ce type d'accès. En pratique, nous stockerons dans les structures `Brin` correspondant aux cadres un pointeur vers les `Brins` du treillage, en utilisant par exemple leur champ `plong`. Dans certains cas, il peut être également utile de retrouver les brins du cadre correspondant à un brin du treillis donné. Dans ce cas, nous ajoutons un pointeur supplémentaire dans les brins du treillage concernés, désignant l'un des brins du cadre.

Comme nous le montrons sur la Figure 1.25, une fois qu'une telle structure est définie, il est facile de définir une opération créant des relations non-variété entre des surfaces. En utilisant les cadres, représentant des objets de dimension supérieure, ces relations non-variété entre des surfaces sont remplacées par des relations variété entre des volumes. Nous définissons une nouvelle opération, appelée **assembler**, permettant de joindre des cloisons, et définie par l'algorithme que nous donnons ci-dessous :

assembler (d_1,d'_1: Brin, dim: **entier**)

Brin $d_2 = \alpha_{dim-1}(d_1)$
Brin $d'_2 = \alpha_{dim-1}(d'_1)$

pour $\begin{cases} b_1 \textbf{ dans } < \alpha_0, \ldots \alpha_{dim-3}(d_1) > \\ b'_1 \textbf{ dans } < \alpha_0, \ldots \alpha_{dim-3}(d_2) > \\ b_2 \textbf{ dans } < \alpha_0, \ldots \alpha_{dim-3}(d'_1) > \\ b'_2 \textbf{ dans } < \alpha_0, \ldots \alpha_{dim-3}(d'_2) > \end{cases}$

b_1->alpha[dim - 1] $= b'_1$; b'_1->alpha[dim - 1] $= b_1$
b_2->alpha[dim - 1] $= b'_2$; b'_2->alpha[dim - 1] $= b_2$

fin *pour*
fin *assembler*

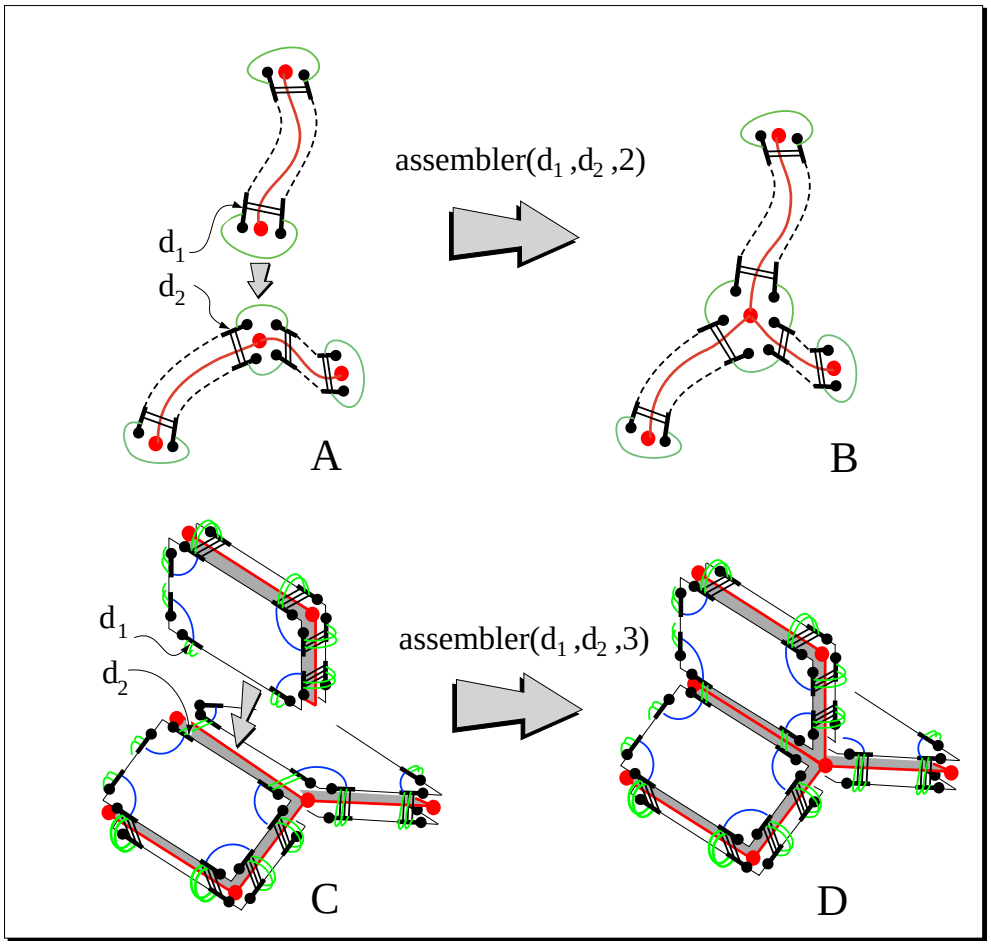


Figure 1.25: *A,B: Création d'une connection non-variété entre trois courbes, représentée en termes de relations variété entre des* **surfaces***). Cette configuration peut alors être représentée par une 2-G-Carte. C,D: cette opération s'applique également à une connection non-variété entre des surfaces, qui peut être considérée comme des relations variété entre des* **volumes***, représentables par une 3-G-Carte.*

La Figure 1.25 montre deux exemples pour la fonction **assembler**, appliquée à des cadres de dimension 2 (Figure 1.25-A,B) et 3 (Figure 1.25-C,D). Lorsque la dimension *dim* des cadres est égale à 2, les orbites de type $< \alpha_0, \ldots \alpha_{dim-3} > (d)$ ne comportent que le brin d, puisque l'ensemble des involutions à parcourir est vide.

Cette opération **assembler** nous permet de construire des partitions de l'espace en assemblant des cloisons de dimension inférieure. Nous appellerons *H-G-Carte* , ou G-Carte hiérarchique, un ensemble de cloisons ainsi combinées par l'opération **assembler**.

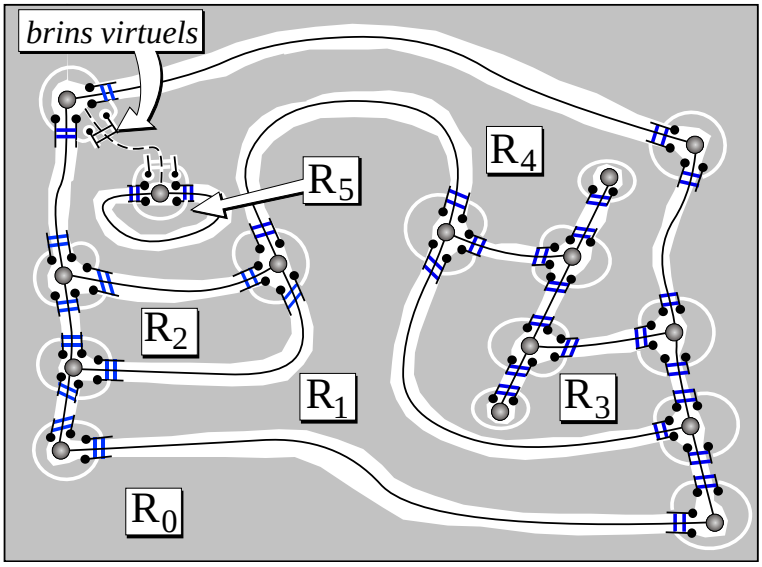


Figure 1.26: *Exemple de H-G-Carte de dimension 2. Les cadres peuvent être combinés, de manière à définir une subdivision de l'espace. Les différentes régions de l'espace ainsi subdivisé peuvent alors être retrouvées en parcourant les orbites* $< \alpha_0, \alpha_1, \ldots \alpha_{dim-1} >$*, où dim désigne la dimension de l'espace. Les régions* $\mathbf{R}_0, \ldots \mathbf{R}_5$ *ont été ainsi identifiées. La région* $\mathbf{R}_0$ *correspond à "l'espace extérieur" du modèle. Dans le cas où des régions présentent des bords internes, comme* $\mathbf{R}_4$ *qui contient* $\mathbf{R}_5$*, des "brins virtuels" sont ajoutés, afin de définir un bord connexe pour la région concernée* $(\mathbf{R}_4)p$*.*

Sur la Figure 1.26, nous montrons un exemple de H-G-Carte, représentant une partition de $\mathbb{R}^2$, déterminée par un ensemble de lignes (les liens α_0 ne sont pas affichés par souci de lisibilité). Nous pouvons alors retrouver les régions 2D de cette partition en parcourant les orbites $< \alpha_0, \alpha_1 >$ de la G-Carte formée par l'assemblage des cadres. Toutefois, les régions présentant des bords internes, comme la région $\mathbf{R}_4$, ne peuvent pas être représentées directement. L'information correspondant à ce qui s'appelle l'*arbre d'inclusion* des différents objets doit être représentée dans la structure. Cet arbre d'inclusion indique pour chaque bord la liste des bords qu'il contient. Plutôt que de représenter directement cet arbre, nous proposons d'utiliser des brins "virtuels", qui vont lier le bord interne de $\mathbf{R}_4$ à son bord externe, ceci permettant d'utiliser les parcours d'orbites standard pour retrouver tous les brins du bord d'une région. Ces brins jouent un rôle purement combinatoire, et ne sont donc pas munis de plongements géométriques. Ils seront identifiés comme des brins "virtuels" par une variable booléenne associée, indiquant qu'ils ne doivent pas être pris en considération lors d'opérations géométriques (tels que des calculs d'intersection).

1.6 Interêt pratique de ce type de représentation

Nous allons discuter ici de l'interêt pratique de ce modèle. Nous verrons dans un premier temps les performances en espace et en temps des G-Cartes, qui ne sont pas leur caracteristique la plus interessante, puisque des représentations comme DCEL [Ket98] affichent un meilleur score en ce qui concerne l'occupation mémoire. Comme nous le verrons plus loin, l'interêt pratique des G-Cartes se situe plutot d'un point de vue généricité, qui permet facilement de gérer ce que nous nommerons l'"explosion combinatoire" induite par l'ecriture de certaines operations. Nous pensons que ce dernier point peut grandement faciliter le processus de développement des grands projets de modeleur 3D. Nous avons implanté les différentes structures et algorithmes décrits dans ce chapitre, ce qui nous a permis décrire un noyau de modeleur. Comme nous le verrons a la fin de la section, la généricité des G-Cartes a rendu ce noyau particulièrement compact et facile à développer.

Afin d'étudier la complexité en espace de cette représentation, nous nous plaçons dans le cas d'une surface triangulée "moyenne", c'est à dire ayant pour chaque sommet six triangles qui s'y rejoignent. Nous considérons que les elements des bords, ayant des voisinnages différents, ont un effet négligeable par rapport aux autres elements (nous pouvons supposer que pour une surface ayant n sommet, il y a de l'ordre de $\sqrt{n}$ sommets sur le bord).

Soient S, A, T les nombres de sommets, aretes, triangles respectivement.
Si A représente le nombre d'aretes, alors:

- Chaque arete est partagee par deux triangles, et chaque triangle compte trois aretes, donc $T = 2/3 \times A$.
- Chaque sommet est partage par six aretes, et chaque arete compte deux sommets, donc $S = 1/3 \times A$.

Ces nombres ($S = 1/3 \times A, A, T = 2/3 \times A$) sont communément admis pour évaluer les performances des modèles. Ainsi, l'occupation en mémoire d'une représentation est évaluer en nombres de mots par arête.

Performances en espace des G-Cartes comparées a DCEL

Nous supposons dans un premier temps que les objets sont munis de plongements sur les sommets uniquement. Dans un deuxieme temps, nous considèrerons également les plongements sur les arêtes et sur les polygones.

Dans la plupart des etudes d'occupation mémoire de structures topologiques, les bits de marquage associes aux élements ne sont pas pris en compte (comme dans [Ber96, Ber97]) (ces bits de marquage sont nécessaire pour les algorithmes de

parcours). Dans le cas des G-Cartes, il nous parait judicieux de les comptabiliser, car les brins des G-Cartes étant des elements "atomiques" plus petis que les `Face`, `HalfEdge` et `Vertex` des autres structures, ils sont présents en nombre beaucoup plus important, ce qui fait qu'un mot ajouté à la structure aura une influence non negligeable sur l'occupation mémoire d'un objet.

Nous considérons donc ici que les brins ont un mot utilisé pour le marquage, et que les `HalfEdge` du DCEL en ont un egalement.

Un brin est alors representé par la structure suivante (en langage C):

```
struct Brin {
  Brin* alpha[3] ;
  Plongement_de_sommet* plongement ;
  int marques_booleennes ;
} ;
```

Et une structure de donnée DCEL par les structures suivantes :

```
struct Vertex {
  Plongement_de_sommet* plongement ;
  HalfEdge* half_edge ;
} ;

struct HalfEdge {
   HalfEdge* next ;
   HalfEdge* opposite ;
   Vertex*   vertex ;
   Face*     face ;
   bool marquage ;
} ;

struct Face {
   HaldEdge* half_edge ;
} ;
```

Pour ces différentes structures, le coût mémoire est (en mots):

G-Cartes:	cout(Brin) = 5
DCEL:	cout(Vertex) = 2 cout(HalfEdge) = 5 cout(Face) = 1

Donc pour une surface triangulée comportant A arêtes, si nous admettons que le nombre de triangles T est egal à $2A/3$ et que le nombre de sommets S est de

$A/3$, alors le coût d'une G-Carte ayant A arêtes est de $5 * 4 * A = 20A$ (il y a quatre brins par arête).

Pour DCEL, le cout est de (il y a 2 `HalfEdges` par arête): $2 \times S + 5 \times 2 \times A + 1 \times T = 2 \times A/3 + 10 \times A + 1 \times 2 \times A/3 = 11A$

Supposons à présent que nous souhaitons représenter des plongements sur tous les éléments (sommets, faces, arêtes). Un brin est alors representé par la structure suivante:

```
struct Brin {
  Brin* alpha[3] ;
  Plongement* plongement[3] ;
  int marques_booleennes ;
} ;
```

Et une structure de donnée DCEL par les structures suivantes:

```
struct Vertex {
  Plongement_de_sommet* plongement ;
  HalfEdge* half_edge ;
} ;

struct HalfEdge {
   HalfEdge* next ;
   HalfEdge* opposite ;
   Vertex*   vertex ;
   Face*     face ;
   Plongement_de_cote* plongement ;
   bool      marquage ;
} ;

struct Face {
   HaldEdge* half_edge ;
   Plongement_de_face* plongement ;
}
```

Pour ces différentes structures, le coût mémoire est:

G-Cartes:	coût(Brin) = 7
DCEL:	coût(Vertex) = 2 coût(HalfEdge) = 6 coût(Face) = 2

- Pour une G-Carte, le coût en mots d'une surface triangulée moyenne comportant A arêtes est de $4 \times A \times 7 = 28A$ (chaque arete correspond à quatre brins).

- Pour DCEL, il est de: $2 \times A/3 + 2 \times 6 \times A + 2 \times 2 \times A/3 = 14A$.

Ces résultats sont résumés dans les tableaux suivant:

Coûts comparés des G-Cartes et de DCEL (avec bits de marquage)

	G-Carte plongements aux sommets	DCEL plongements aux sommets	G-Cartes plongements partout	DCEL plongements partout
coût	$20A$	$11A$	$28A$	$14A$

Les chiffres correspondant à d'autres structures de données, telles que celle de Weiler [Wei84], sont comparables a ceux obtenus pour DCEL. Le lecteur se rapportera à [Ber96, Ber97] pour plus de details.

Nous donnons également les coûts des différentes structures, calculés sans tenir compte des bits de marquage (comme dans [Ber96, Ber97]).

Coûts comparés des G-Cartes et de DCEL (sans bits de marquage)

	G-Carte plongements aux sommets	DCEL plongements aux sommets	G-Cartes plongements partout	DCEL plongements partout
coût	$16A$	$8.7A$	$24A$	$12A$

En pratique, puisque les G-Cartes n'utilisent qu'un seul type d'élément, de taille constante, il est facile d'écrire un allocateur memoire spécialisé, qui fournira des meilleures performances que l'operateur `new` par défaut du C++. L'allocateur écrit par T. Valentin permet de gagner 25 % d'occupation mémoire par rapport à l'allocateur par défaut, ce qui réduit l'écart entre les deux types de structures.

Performances en temps des G-Cartes comparées a DCEL

Avant de nous interesser aux aspects liés au développement logiciel autour des G-Cartes, nous abordons rapidement le problème des performances en temps. Ce problème peut être traité en mettant en évidence un isomorphisme entre les deux structures, si nous nous limitons à des subdivisions de variétés orientables. Nous nous intéressons ici plus particulièrement à des surfaces (la même étude peut être réalisée pour des volumes).

Si nous considérons les subdivisions de variétes 2D orientables, domaine de représentation de DCEL, alors nous pouvons mettre en correspondance toute 2-G-Carte avec une structure DCEL et récriproquement. Pour une G-Carte d'une surface orientable, l'ensemble $\mathcal{D}$ de ses brins peut être partitionné en deux sous ensemble $\mathcal{D}^+$ et $\mathcal{D}^-$ comportant le même nombre de brins (voir la Section 1.4.2). Un `HalfEdge` DCEL correspond alors à un couple de brins $(d+, d-)$, avec :

$$\begin{cases} d+ & \in & \mathcal{D}^+ \\ d- & \in & \mathcal{D}^- \\ \alpha_0(d+) & = & d- \\ \alpha_0(d-) & = & d+ \end{cases}$$

Dans cette optique, il est possible de décrire une structure DCEL en remplaçant les mots du vocabulaire DCEL par des mots de celui des G-Cartes. Ceci permet d'implanter les operations de DCEL en termes de G-Cartes. Comme le même élément peut être désigné par plusieurs brins différents, le test d'égalité entre deux elements peut être exprimé en comparant les plongements.

`HalfEdge e`	$\leftrightarrow$	$(d+, d-)$
`next`	$\leftrightarrow$	$\alpha_1(d+)$
`opposite`	$\leftrightarrow$	$\alpha_2(d-)$
`vertex`	$\leftrightarrow$	$d+$
`face`	$\leftrightarrow$	$d+$
`plongement`	$\leftrightarrow$	$d + .plong[1]$
`e == e'` (test d'égalité)	$\leftrightarrow$	$d == d'$
`Face f`	$\leftrightarrow$	d (Brin)
`half_edge`	$\leftrightarrow$	d
`plongement`	$\leftrightarrow$	$d.plong[2]$
`f == f'` (test d'egalite)	$\leftrightarrow$	$d.plong[2] == d'.plong[2]$
`Vertex v`	$\leftrightarrow$	d (Brin)
`half_edge`	$\leftrightarrow$	d
`plongement`	$\leftrightarrow$	$d.plong[0]$
`v == v'` (test d'egalite)	$\leftrightarrow$	$d.plong[0] == d'.plong[0]$

Comme il est possible de ré-exprimer les opérations de base de DCEL en termes d'operations de G-Cartes ayant une complexite constante, ceci nous donne la meme complexite en temps pour les algorithmes agissant sur les deux structures (Néanmoins, des précautions doivent etre prises: en effet, avec cette structure, le parcours des sommets d'une face ne se fera pas toujours dans le meme ordre suivant le brin utilisé pour référencer la face). En termes d'implantation, dans CGAL [CGA], nous pensons qu'il est possible d'écrire des classes dites

"Adaptateur"[12] pour décrire une classe pouvant servir d'argument formel de type `DCEL_data_structure` aux classes generiques de CGAL.

En pratique, des tests de performances ont montré que le temps de parcours des sommets connectés a un sommet donné (temps mesures sur une machine SGI, processeur R4000 a 250 MHZ):

- dans le cas des surface, coûte plus cher de 10% que l'implantation actuelle de gOcad (qui stocke un tableau de pointeurs d'`HalfEdge` dans chaque `Vertex`).
- dans le cas des volumes, coûte plus cher de 120% que l'implantation actuelle de gOcad (comme nous utilisons un marquage, il faut démarquer après le parcours, ce qui explique que cette operation coûte deux fois plus cher qu'avec un stockage direct des `HalfEdge` dans les `Vertex`).
- Interpolation D.S.I. sur une surface comportant 20000 triangles (utilisation intensive des itérateurs retournant l'ensemble des sommets connectés par une arête à un sommet donné):
  - implantation actuelle de Gocad: 18 secondes
  - G-Cartes: 21 secondes

Nous avons observé des meilleurs temps de réponse pour les opérations de modification de maillage:

- Subdivision de chaque triangles en 4, pour une surface en comportant 20000:
  - implantation actuelle de gOcad: 25 secondes
  - G-Cartes: 15 secondes

Toutefois, comme nous l'avons indiqué, les performances en temps (tout comme l'occupation mémoire) ne sont pas le point le plus intéressant de cette structure. Sa puissance se manifeste plutôt lorsque nous considérons globalement le processus de développement d'un modeleur 3D.

Les G-Cartes : Cartes Généralisés ou Cartes Génériques

Lorsque certaines familles d'opérations sont considérées, on obtient rapidement ce que nous appelons une "explosion combinatoire" du code à écrire. Nous pouvons citer par exemple le problème des itérateurs locaux. Ainsi, dans le cas des subdivisions de surfaces, nous souhaitons être capables d'itérer:

[12] Adapter class

- sur les sommets d'un polygone
- sur les segments d'un polygone
- sur les sommets d'un segment

Dans le cas des subdivisions de volumes, les itérateurs doivent permettre de parcourir:

- les sommets d'un polyedre
- les segments d'un polyedre
- les faces d'un polyedre
- les sommets d'un polygone
- les segments d'un polygone
- les sommets d'un segment

Dans le cas général, pour un objet de dimension n, il y a $1 + 2 + ... + n = n(n + 1)/2$ iterateurs différents. Si nous utilisons des structures classiques, une fonction différente devra etre écrite pour chacun de ces iterateurs, car les pointeurs permettant de naviguer entre des elements de dimensions différentes ont des noms differents. En utilisant les G-Cartes, il est possible d'exprimer ces iterateurs par des classes patron[13], dont les arguments formels sont les parcours d'orbites des differentes cellules mises en jeu. Ceci permet de réduire la taille du code de manière importante (voir le tableau a la fin du chapitre), et de diminuer les risques d'erreurs de programmation. Ceci s'applique également aux operations d'edition de maillage, comme:

- creation/destruction d'une n-cellule
- insertion/supression d'une n-cellule (par exemple, si un segment est supprime, les deux polygones qui le partagent sont fusionnés).
- identification de tous les sommets d'une cellule ("écroulement" d'une cellule sur elle-même).
- opérations *coudre*/*découdre* (voir au début du chapitre)

[13] template

Pour toutes ces operations, la cellule à manipuler est désignée par un brin incident à la cellule et par un entier correspondant à la dimension de la cellule. L'explosion combinatoire de ces operations (chacune d'entre elle compte $n + 1$ versions différentes pour un object de dimension n) est gérée par des classes patrons, parametrées par les parcours des cellules mises en jeu.

Ces deux exemples expliquent la reduction du volume de code mise en evidence dans le tableau de la fin du chapitre. Nous pouvons également citer les opérations de raffinement de maillages, l'affichage graphique, ou encore les opérations d'accès aux voisinnages (utilisées intensivement par le lissage discret et par la méthode de paramétrisation des deux autres chapitres). Ces differentes fonctions ont elles aussi pu etre parametrées par des parcours de cellules.

Nous pensons que cet aspect générique (qui facilite l'implantation d'opération *générales*) ainsi que l'aspect formel (qui facilite la spécification et la mise en place de tests de non-régression) compensent la consommation mémoire légèrement supérieure de ce type de représentations. En ce qui concerne les performances en temps, les complexités mises en jeu sont les mêmes que celles des modèles classiques (DCEL), ce qui a été confirmé en pratique (temps d'execution des algorithmes). Pour conclure sur l'occupation mémoire légèrement supérieure, il nous paraît important de rappeler que le choix de coder l'année sur deux chiffres, effectué au début des années 80 pour un grand nombre de logiciels, a coûté récemment sans doute plus cher que la mémoire nécessaire pour ajouter les deux chiffres manquants. Ceci est d'autant plus vrai à notre époque, où le prix de la mémoire a sensiblement diminué.

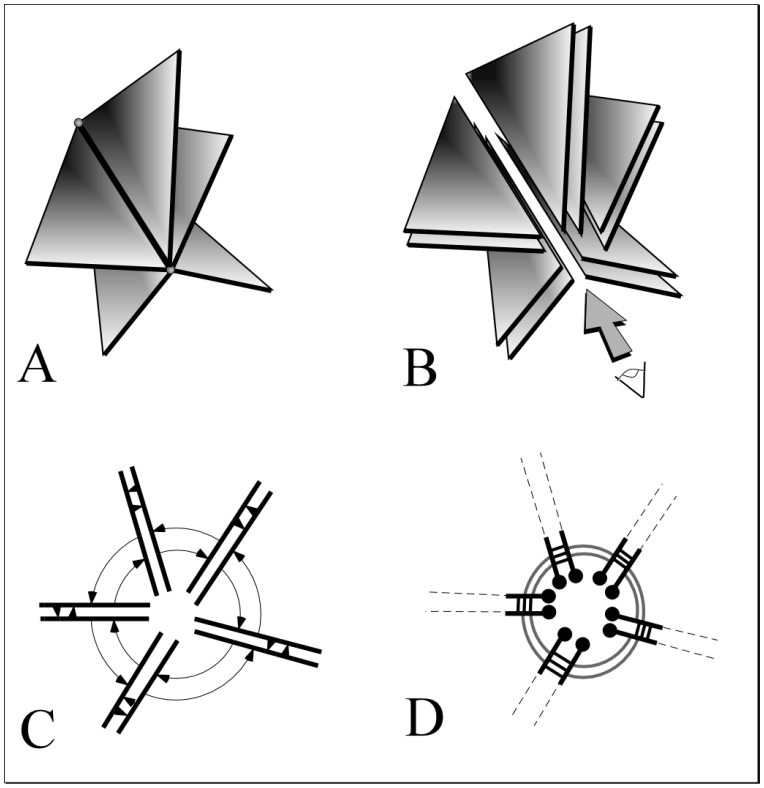


Figure 1.27: *Pour représenter une arête radiale (A), la représentation de Weiler consiste à dédoubler les surfaces (B) et à positionner un ensemble de pointeurs entre ces surfaces (C). Notre structure de G-Cartes hiérarchique se présente d'une manière très similaire (D), à la distinction près que des relations entre des* **volumes** *sont représentées. De plus, les relations combinatoires entre les involutions α_i mises en jeu sont mieux formalisées que les pointeurs intervenant dans la représentation de Weiler.*

1.7 Bilan et perspectives

Dans ce chapitre, nous avons introduit la notion mathématique de carte généralisée [Lie94]. En nous basant sur cette notion, nous avons défini une représentation hiérarchique, appelée H-G-Carte, et nous avons montré comment la modélisation géométrique pourrait bénéficier de ce type d'approches. Nous avons implanté un noyau topologique utilisant ces notions et fonctionnant en dimension arbitraire. Ce noyau a en même temps fourni un domaine de représentation plus vaste que les approches classiques, tout en réduisant fortement la taille du code (voir le tableau ci-dessous). Cette réduction de la taille du code provient de la structure hautement générique des G-Cartes, qui permet une utilisation intensive de constructions C++ telles que les *classes patron* et la *spécialisation partielle* [Str97, SL95]. Les performances en espace et en temps n'ont pas changé, alors que beaucoup de fonctionnalités ont été ajoutées.

Représentation	**Taille du code** (lines)	**Objets représentables**
Demi-arête	$\simeq$ 10000	Lignes polygonales, Surfaces triangulées, Volumes tétraédrisés
H-G-Cartes	$\simeq$ 3000	Lignes polygonales, Surfaces ayant des polygones arbitraires, Volumes ayant des polyèdres arbitraires, objets 4D (et nD)

Si nous comparons notre structure avec la représentation par arêtes radiales [Wei86] (voir Figure 1.27), nous pouvons voir que les deux structures se présentent d'une manière très similaire. La différence majeure entre les deux est que notre représentation modélise des relations variété entre des volumes, alors que la représentation par arêtes radiales modélise des relations non-variété entre des surfaces. Notre structure est ainsi définie d'une manière plus formelle, en termes de relations entre cellules qui se généralisent facilement à des dimensions arbitraires. La fonction **assembler** que nous avons présentée pourrait alors être utilisée pour construire des volumes non-variété, considérés comme une partition variété d'un hyper-volume (de dimension 4).

La principale limitation de ce type d'approches est qu'elle ne permet pas la représentation directe de configurations non-variété. Nous pensons que ceci ne constitue pas une réelle limitation de notre représentation. En effet, nous avons montré que dans notre cas, les configurations de surfaces non-variété correspondaient en réalité à des volumes variété, représentables par notre structure. En définissant l'opération **assembler**, nous avons également préservé le modèle d'interaction avec l'utilisateur, qui sera toujours persuadé de manipuler des surfaces.

Un autre cas de configuration non-variété se rencontre parfois en C.A.O., quand par exemple la connection entre une antenne de radio et la carrosserie d'une automobile doit être modélisée. Dans la plupart des cas, l'antenne sera représentée par une courbe, connectée de manière non variété avec la carrosserie de la voiture. Un problème similaire se pose en géologie, lorsque nous devons modéliser l'intersection entre un puits et une surface représentant la limite entre deux couches géologiques (un horizon). Dans de telles configurations, nous pensons que l'antenne et le puits devraient être représentés par des cylindres volumiques, car les objets qu'ils représentent sont des volumes. Toutefois, il convient de prendre en compte les besoins de l'utilisateur, qui ne souhaite pas effectuer le

travail supplémentaire lié à la modélisation de ces volumes. Dans cette optique, notre structure hiérarchique pourrait être utilisée, en représentant l'antenne ou le puits par une ligne polygonale. Cette ligne polygonale serait alors entourée par un cadre volumique, comme nous l'avons défini dans la Section 1.5. Ainsi, du point de vue combinatoire, des relations entre des **volumes** seraient représentées, et du point de vue de la géométrie, l'antenne et le puits pourraient être modélisés par des courbes, comme le souhaitait l'utilisateur.

D'un point de vue algébrique, comme les G-Cartes constituent une représentation totalement spécifiée, une opération permettant de les manipuler doit plutôt être considérée comme un opérateur que comme un algorithme. Il est alors possible de prouver qu'une opération réalise bien ce qu'elle est censée faire [BDFL93]. Ceci permet de limiter une certaine classe d'erreurs de programmation, et résulte ainsi en un logiciel très robuste. Pour ces raisons, le noyau à base de G-Cartes a été utilisé pour remplacer le noyau du logiciel Gocad, dédié aux géosciences. Dans ce contexte, comme nous le montrons sur la Figure 1.28, les structures du sous-sol à modéliser peuvent être vues comme une partition de l'espace 3D en régions 3D. Chaque région est représentée par sa frontière, définie comme un ensemble de surfaces inter-connectées, chacune de ces surfaces étant elle-même discrétisée en un ensemble de polygones. Notre modèle hiérarchique peut alors être utilisé pour représenter cette structure, comportant deux (ou plusieurs) G-Cartes emboîtées. Au niveau supérieur, une 3-G-Carte représente les relations volumiques entre les régions 3D, et au niveau inférieur, une 2-G-Carte correspond à la discrétisation des surfaces en polygones.

Des progrès récents dans le domaine de l'analyse par éléments finis et la résolution d'équations aux dérivées partielles (voir par exemple [Ver96]) requièrent la création de maillages 3D complexes, tels ceux que nous montrons sur la Figure 1.28. La possibilité de représenter des polyèdres arbitrairement connectés permet de discrétiser adaptativement les problèmes à modéliser, ce qui réduira les artefacts dus à l'anisotropie de maillage, que nous rencontrons lorsque des grilles héxaédrales sont utilisées. Les opérations de création et d'édition de ce type de maillages sont actuellement étudiées et réalisées par S. Conreaux, dans le cadre de sa thèse [Cnr00], qui traite également de certains aspects des G-Cartes hiérarchiques. La Figure 1.28 montre des maillages obtenus grâce aux premiers résultats de S. Conreaux. Afin de permettre une utilisation de ce nouveau type de support par des applications industrielles, les points suivants vont également être étudiés:

- Chargement/sauvegarde de fichiers.
- Affichage graphique, rendu volumique.

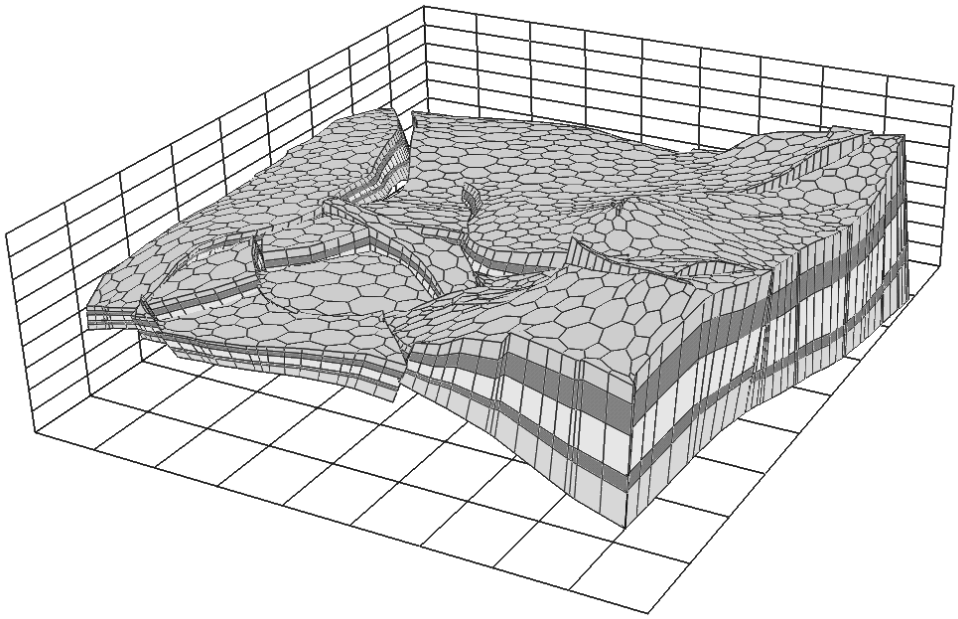


Figure 1.28: *Maillage pour solveur éléments finis, utilisé pour simuler le flux de pétrole dans une couche géologique (premiers résultats de S. Conreaux).*

- Manipulation des propriétés attachées à ces objets.

Dans l'objectif de munir le modeleur avec un jeu d'opérations complet, permettant une grande puissance de modélisation, les opérations booléennes représentent une extension possible de ce modèle. Ces opérations, fondées sur un calcul d'intersection entre les objets, permettent de considérer les objets comme des ensembles de points de l'espace, et de les munir des opérations ensemblistes (union, intersection, négation). Les calculs d'intersections liés à ce type d'opérations sont très sensibles aux problèmes de précision numérique, et mettent souvent en jeu un ensemble de structures de données complexes. C'est pourquoi nous avons jeté les bases d'une structure simpliciale hiérarchique, qui sera améliorée et développée par M. Dazy dans le cadre de son travail de thèse [Daz01]. Bien que les G-Cartes permettent de représenter des cellules ayant un nombre de faces arbitraire, plusieurs facteurs suggèrent fortement d'utiliser des simplexes pour effectuer les calculs d'intersection (voir Figure 1.29). Dans son mémoire de thèse [Cnr97], J. Conraud a également étudié ces problèmes. Nous les mentionnons ici rapidement, afin de justifier le choix d'utiliser des simplexes dans ce cas précis.

Considérons tout d'abord le choix le plus général, consistant à utiliser des polygones ayant un nombre quelconque de côtés, et dont les sommets peuvent ne pas être coplanaires. Comme nous le montrons Figure 1.29-A,B,C,D, ces polygones non plans peuvent parfois causer certaines ambiguités. En effet, la ligne

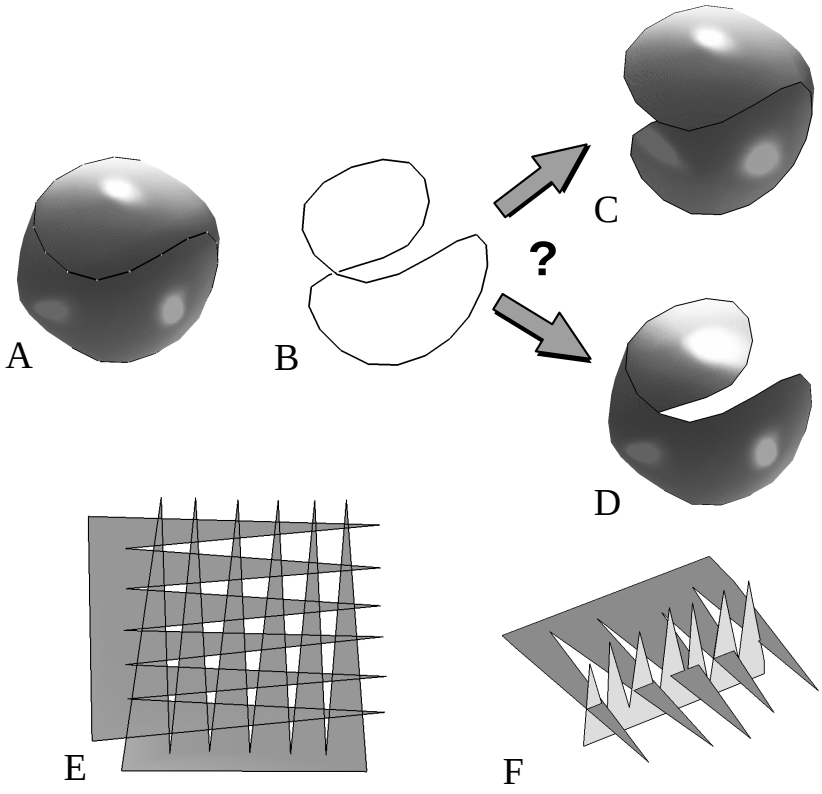


Figure 1.29: *Problèmes liés à la modélisation avec des polygones arbitraires; un polygone non-plan défini comme la couture d'une balle de tennis (A,B) est ambigü (C,D); des polygones concaves peuvent générer un grand nombre d'intersections (E), pouvant avoir des relations topologiques complexes (F)*

polygonale correspondant à la couture d'une balle de tennis (Figure 1.29-A,B) peut correspondre aux deux moitiés de la balle de tennis (Figure 1.29-C,D). Pour cette raison, nous limiterons donc les éléments à des polygones plans.

Comme nous le montrons Figure 1.29-E,F, cette restriction n'est pas suffisante. En effet, des polygones concaves peuvent générer un grand nombre d'intersections, qui peut atteindre l'ordre du carré du nombre de côtés des polygones mis en jeu (Figure 1.29-E). De plus, les relations topologiques entre les segments résultant de l'intersection et les polygones concernés peuvent être relativement complexes (Figure 1.29-F). Ce problème a été étudié par Cazier dans [Caz97], mais l'algorithme qu'il a décrit et mis en oeuvre reste difficile à spécifier totalement. Afin de simplifier les algorithmes et structures de données, nous éliminerons donc également les polygones concaves.

A cette étape du raisonnement, les éléments que nous considérons sont des polygones plans et convexes. Il s'avère que pour des raisons de précision numérique,

nous devons également éliminer les polygones de plus de trois côtés. En effet, à cause de la représentation des nombres en mémoire, un ensemble de points supposés coplanaires ne le restera pas obligatoirement lorsqu'il sera représenté en machine. Lorsque des jeux de données complexes doivent être traités, comme ceux rencontrés en géologie numérique, ces problèmes de précision numérique sont suffisants pour rendre les algorithmes instables. Ces différents aspects justifient notre choix des simplexes comme éléments de base pour la structure. Dans le cas où des intersections entre des objets non-simpliciaux doivent être calculées, il est toujours possible de décomposer "virtuellement" les objets considérés en simplexes. La thèse de M. Dazy [Daz01] traitera donc d'une structure de données appelée un *ensemble simplicial hiérarchiquement plongé*, décrivant les relations d'inclusions entre plusieurs ensembles simpliciaux [LL95, Lan96], générés par les intersections entre objets de dimensions diverses.

Annexe: Fonctions auxiliaires

Dans cette annexe, nous détaillons les différentes fonctions auxiliaires référencées dans le texte. Ces fonction sont utilisées comme "briques de base" pour définir les opérations plus évoluées que nous avons vues dans les sections précédentes.

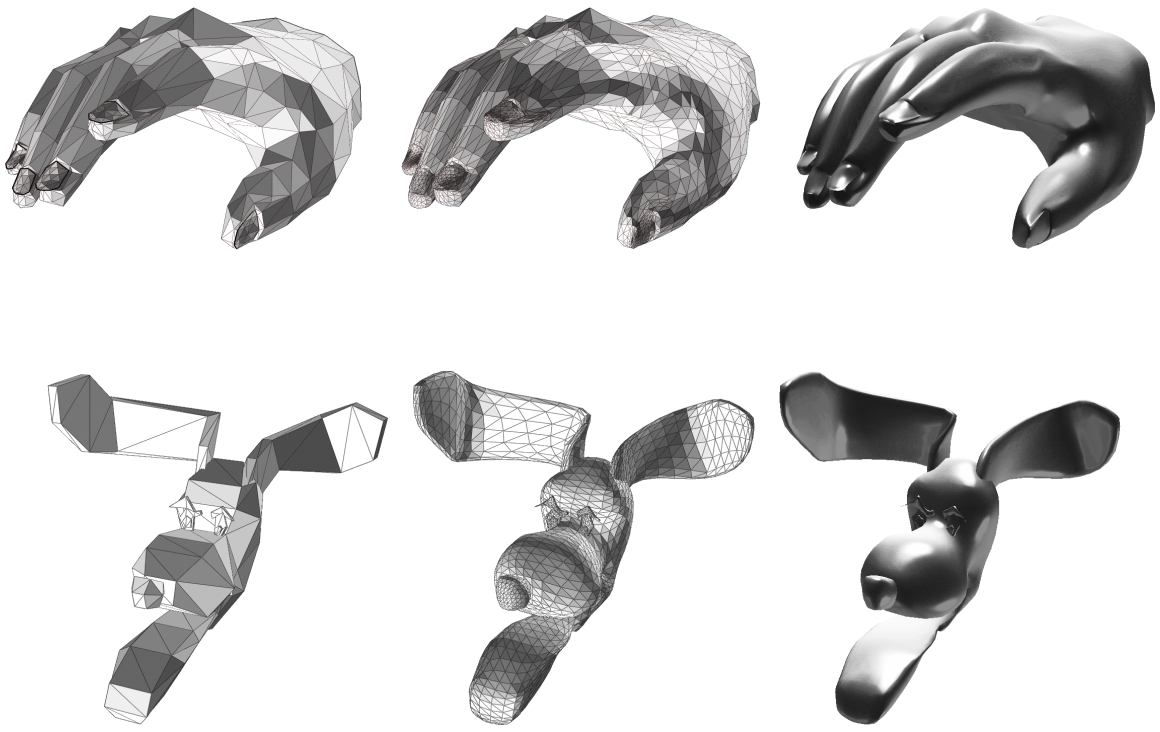
```
partager_ou_copier_plong (d1,d2: Dart, dim: int)
   k1 = trouver_clef_de_cell(d1, dim) ;
   k2 = trouver_clef_de_cell(d2, dim) ;
   si k1 ≠ k2
      si k1 = NIL
         nouv_plong : pointeur de Plongement =
           copier_plongement(d2, dim);
         d1->clef_de_cell[dim] = vrai ;
         repartir_plongement(d1, dim, nouv_plong) ;
      sinon
         nouv_plong : pointeur de Plongement =
           copier_plongement(d1, dim);
         d2->clef_de_cell[dim] = vrai ;
         repartir_plongement(d2, dim, nouv_plong) ;
      fin si
   fin si
fin partager_ou_copier_plong
```

```
trouver_clef_de_cell (debut: Brin, dim: entier)
   pour d dans <α_dim >(debut)
      si d->clef_de_cell[dim] alors retourner d
   fin pour
   retourner NIL
fin trouver_clef_de_cell
```

```
repartir_plongement (debut: Brin, dim: entier,
         plong: pointeur de Plongement )
   pour d dans <α_dim >(debut)
      d->plong[dim] = plong
   fin pour
fin repartir_plongement
```

Chapitre 2

Lissage variationnel discret et contraint



2.1 Introduction

Le chapitre précédent a introduit un modèle combinatoire, permettant de représenter un ensemble de discrétisations emboitées, correspondant aux différents niveaux de représentations souhaitées pour un modèle géologique. Cette représentation ne prend en compte que les relations **combinatoires** entre les éléments composant le modèle. Autrement dit, au niveau d'une G-Carte, la manière dont les éléments sont connectés entre eux est représentée. Afin de modéliser des objets géologiques, il est nécessaire de munir ce modèle combinatoire d'une **géométrie** (nous utiliserons également le terme de *plongement*), à savoir mettre en correspondance les éléments du modèle combinatoire avec des ensembles de points de $\mathbb{R}^3$. Dans ce chapitre, nous étudierons une représentation **discrète** des objets, où seuls les sommets sont munis de coordonnées dans $\mathbb{R}^3$ (par opposition aux approches de type "courbes et surfaces", que nous évoquerons dans la suite du chapitre).

Dans ce contexte, nous présentons une nouvelle méthode d'interpolation, de lissage et d'édition de maillages arbitraires. Notre approche est fondée sur la méthode D.S.I.[1][Mal89, Mal92, Mal99], considérée ici dans un contexte de modélisation variationnelle. Ceci va nous permettre de mieux situer ce type de méthodes par rapport aux autres approches existantes, de caractériser la surface limite obtenue après une suite de subdivision d'une surface donnée, et enfin d'éliminer certains effets d'oscillations qui apparaissent près des bords des surfaces.

Étant donné un maillage dont certains nœuds sont fixés, la méthode que nous allons présenter dans cette partie permet de choisir pour les autres nœuds une position qui minimise une fonction objective. Le critère que nous allons minimiser, le carré du *Laplacien discret*, est une fonction objective similaire à l'énergie de flexion d'une plaque mince, bien connue dans le domaine de la C.A.O. [Gre94]. Comme nous le verrons, notre approche fournit le même domaine de modélisation qu'une approche récente, appelée *surfaces de subdivision* [ZS98, Zor98]. En utilisant ce type de méthodes, des objets de topologie arbitraire peuvent être traités; il est possible de modéliser des arrondis et des angles plus ou moins marqués; et des subdivisions récursives de la surface initiale convergent vers une surface lisse. Par ailleurs, notre méthode offre plus de flexibilité que les surfaces de subdivision, puisqu'elle ne se limite pas aux maillages présentant une connectivité de subdivision, et que des contraintes linéaires peuvent être prises en compte au sens des moindres carrés. Par exemple, nous verrons comment ajuster une surface à un ensemble de points arbitraires, ayant éventuellement des normales spécifiées. Notre méthode pourrait également avoir d'importantes répercussions dans les domaines de l'édition multi-résolution et de la compression de maillages.

[1] Discrete Smooth Interpolation

Ces quelques dernières années, plusieurs classes de méthodes de modélisation, agissant directement sur des maillages polygonaux, ont soulevé beaucoup d'intérêt de la part de la communauté scientifique. Comme nous le verrons plus loin, la famille de méthodes connue sous le nom de *subdivision* permet de considérer une surface comme le résultat de raffinements successifs, appliqués à une surface initiale appelée *maillage de contrôle* [DS78, Cat74]. Des approches plus conventionnelles, désignées par le terme de *modélisation variationnelle* [Gre94, MS92, Sei98], consistent à représenter les surfaces sous forme paramétrique, par des fonctions (en général des polynômes) et à satisfaire certains critères. Récemment, une classe de méthodes hybrides appelée *lissage discret* [Kob97] a fait son apparition, permettant d'appliquer une approche de type variationnel à des maillages discrets. Tandis que l'approche de type subdivision consiste à raffiner itérativement un maillage donné en utilisant une règle systématique pour insérer de nouveaux sommets, le lissage discret permet de manipuler directement des maillages arbitraires. Un maillage pour lissage discret possède un ensemble de sommets fixés par l'utilisateur, appelés *noeuds de contrôle*, et la position des autres sommets est calculée de manière à interpoler les noeuds de contrôle. Toutefois, cette dernière famille de méthodes n'offrait pas jusqu'à maintenant l'ensemble des fonctionnalités spécifiques aux autres domaines que nous avons mentionnés. Par exemple, les méthodes de lissage variationnel permettent la modélisation d'arrondis et d'angles plus ou moins marqués, ou encore l'ajustement d'une surface à un ensemble de points de données.

La méthode de lissage discrète D.S.I. [Mal89, Mal92, Mal99], combinée avec les développements présentés dans cette partie, permet de satisfaire ces contraintes. Du point de vue du lissage variationnel, nous minimisons une fonctionnelle correspondant au carré du *Laplacien discret* de la surface. Nous étudions ici ce critère en utilisant une paramétrisation arbitraire de la surface, locale ou globale, alors que les travaux précédents [Kob97, KCVS98, WW94] s'appliquent à quelques cas particuliers. De plus, l'algorithme itératif proposé ici permet d'obtenir une réponse du modeleur en temps réel dans un environnement interactif. Nous montrerons également que, comme toute autre technique de lissage discret, notre méthode peut être utilisée de la même manière qu'un schéma de subdivision, permettant de raffiner itérativement les maillages.

Dans la section 2.2 de ce chapitre, nous introduisons les différents champs théoriques connus en modélisation, tels que les méthodes de subdivision, la modélisation variationnelle et le lissage discret. Dans la section 2.3, nous présentons notre méthode d'interpolation discrète comme une façon de minimiser le carré du Laplacien discret d'un maillage arbitraire. Nous aborderons également les problèmes posés par la modélisation d'arrondis et d'angles plus ou moins marqués, ainsi que ceux liés en C.A.O. au raccordement entre plusieurs surfaces. La méthode est généralisée en section 2.4, permettant de prendre en compte un ensemble de contraintes linéaires au sens des moindres carrés. Par exemple,

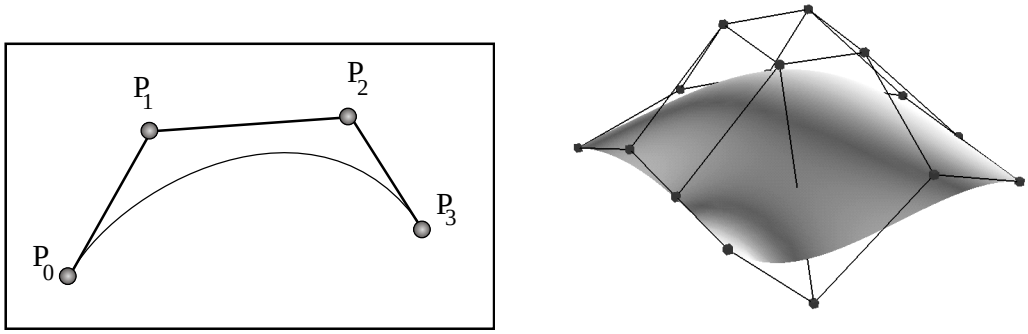


Figure 2.1: *A: Courbe de Bézier de degré 3; B: Carreau de Bézier obtenu par produit tensoriel de deux courbes de degré 3.*

nous montrons comment ajuster une surface à un ensemble de points de données répartis de manière arbitraire, ayant éventuellement des normales définies. La dernière section conclut sur des améliorations éventuelles.

2.2 Fondements théoriques

Dans le domaine de la modélisation 3D, l'objectif principal est de définir des modèles de courbes et de surfaces permettant un maximum de flexibilité pour l'utilisateur, ainsi que certaines propriétés géométriques, différentes suivant les domaines d'application. Afin d'atteindre ces objectifs, plusieurs outils mathématiques différents ont été définis dans la littérature, comme nous allons le voir dans cette partie. La méthode que nous proposons partage certaines caractéristiques avec les quatre domaines les plus connus, comme les méthodes "courbes et surfaces", les surfaces de subdivision, la modélisation variationnelle ainsi que le lissage discret. Pour cette raison, il nous semble intéressant de rappeler ici brièvement quelques aspects de ces différents domaines, qui vont nous permettre ensuite de définir et de caractériser notre méthode.

2.2.1 Courbes et surfaces

Cette sous-section ne prétend pas donner une description exhaustive des méthodes de type courbes et surfaces polynomiales, car notre problématique concerne la manipulation directe de surfaces triangulées. Toutefois, comme les méthodes que nous allons introduire plus loin partagent certains points communs avec ces ensembles de techniques, il nous a paru utile d'en rappeler en quelques pages l'idée générale. Les approches classiques consistent à définir une surface par une fonction polynomiale, ou encore par une fraction rationnelle. Une vue d'ensemble de ce type d'approches est donnée dans l'ouvrage de Farin [Far92] (voir également

[Sei98, Fuc97]). Nous allons ici évoquer brièvement ces approches classiques, de type **continu**, avant de nous pencher sur les différentes familles de méthodes **discrètes**. Nous montrerons en conclusion du chapitre le lien qui lie notre approche avec les méthodes classiques d'ajustement aux données.

Courbes et carreaux de Bézier

Les courbes et carreaux de Bézier représentent l'un des modèles les plus connus et les plus anciens en matière de modélisation de courbes et de surfaces. Ils ont été mis au point à la fin des années cinquante par Bézier [Bez70, Bez86] et De Casteljau [dC59]. Une courbe de Bézier de degré n est déterminée par un ensemble de $n+1$ points $\mathbf{p}_i$, appelé *maillage de contrôle* (voir Figure 2.1-A). L'équation d'une courbe de Bézier est définie sous forme paramétrique, par une fonction φ d'un paramètre u variant sur l'intervalle $[0,1]$:

$$\varphi(u) = \sum_{i=0}^{n} B_i^n(u).\mathbf{p}_i$$

avec: (2.1)

$$B_i^n(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$$

Dans cette équation, les fonctions B_i^n (appelées les polynômes de Bernstein) vérifient certaines propriétés utiles. En effet, comme les coefficients mis en jeu sont ceux du binôme de Newton $(a+b)^n$, il est facile de vérifier les équations suivantes (en posant $a=u$ et $b=1-u$):

$$\left|\begin{array}{lll} \forall 0 \leq u \leq 1, & \sum_{i=0}^{n} B_i^n(u) & = (u+1-u)^n = 1 \\ \forall 0 \leq u \leq 1, & B_i^n(u) & \geq 0 \end{array}\right. \quad (2.2)$$

Autrement dit, pour toute valeur $0 \leq u \leq 1$, $\varphi(u)$ est une *combinaison convexe* des points de contrôle $\mathbf{p}_i$. Ceci signifie qu'une courbe de Bézier est entièrement contenue dans l'enveloppe convexe de ses points de contrôle. De plus, il est facile de vérifier que la courbe passe par le premier et le dernier point du maillage de contrôle ($\varphi(0)=\mathbf{p}_0$ et $\varphi(1)=\mathbf{p}_n$). La courbe aura donc une forme relativement similaire à celle du maillage de contrôle. Ce dernier offre alors un moyen intuitif (bien qu'indirect) pour modifier cette courbe.

Les équations définissant les courbes de Bézier peuvent être également utilisées pour définir une surface. Une telle surface est alors représentée par une fonction de deux paramètres, appelée un *carreau* (le terme Anglais *patch* est également utilisé dans la littérature). Par exemple, le produit tensoriel entre deux courbes est une manière simple d'obtenir ce résultat:

$$\varphi(u,v) = \sum_{i=0}^{n} . \sum_{j=0}^{m} B_i^n(u).B_j^m(v).\mathbf{p}_{i,j} \tag{2.3}$$

Le maillage de contrôle est alors une grille $n \times m$ de points de contrôle (voir Figure 2.1-B) et la propriété de l'enveloppe convexe est toujours vérifiée.

Pour définir des surfaces de Bézier, une autre possibilité est donnée par les carreaux triangulaires [Far86, Far87, Far92] (qui peuvent également se généraliser à des simplexes de dimension supérieure, voir par exemple [Fuc97]):

$$\varphi(u,v,w) = \sum_{i+j+k=n} B_{i,j,k}^n(u,v,w).\mathbf{p}_{i,j,k}$$

avec:

$$\left| \begin{array}{lcl} w & = & 1-u-v \\ B_{i,j,k}^n(u,v,w) & = & \frac{n!}{i!j!k!}.u^i.v^j.w^k \end{array} \right. \tag{2.4}$$

Plusieurs autres familles de surfaces, appelées des *Splines* [dB78, dB87, Sch81, BBB87], ont ensuite été définies pour contourner certaines limitations liées aux carreaux de Bézier. Par exemple, pour les surfaces de Bézier, le contrôle est global, ce qui signifie que la modification d'un point de contrôle va se répercuter sur la totalité de la surface. Au contraire, les surfaces dites β-Splines [Bar81, Bar88] fournissent un contrôle local, mieux adapté à un environnement interactif.

Continuité géométrique G^k

Si des modèles complexes doivent être construits, deux manières différentes permettent *a priori* de prendre en compte cette complexité :

- *Utiliser des carreaux de haut degré :* Malheureusement, ceci conduit à la fois à des phénomènes d'oscillations et à des instabilités numériques, ce qui rend ce type d'approches inutilisable dans la pratique.
- *Utiliser plusieurs carreaux, assemblés entre eux :* (voir Figure 2.2). Le problème du raccord entre carreaux est loin d'être trivial, si certains ordres de continuité doivent être respectés. Nous y reviendrons par la suite.

Lorsque plusieurs surfaces sont raccordées entre elles, l'aspect visuel du résultat dépendra beaucoup de l'ordre de continuité au niveau du raccord. Ceci va également avoir une grande importance si la surface en question doit être utilisée comme support pour des calculs numériques. La définition classique de la continuité d'une fonction φ en un point x_0 est donnée par l'Équation ci-dessous :

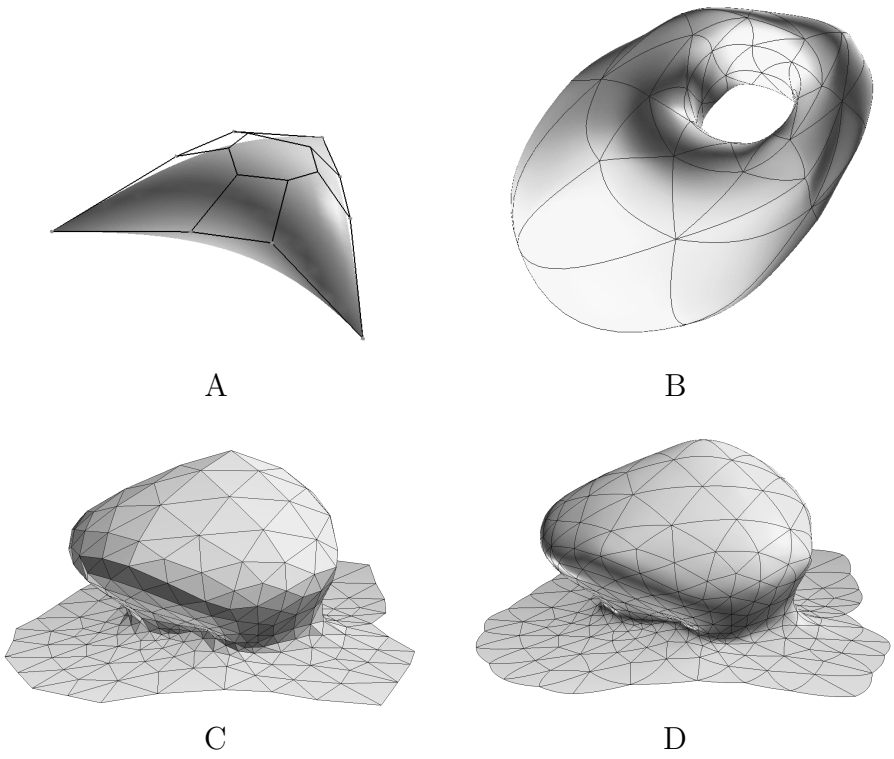


Figure 2.2: *A: Carreau triangulaire de Gregory; B: Surface G^1 obtenue par assemblage de triangles de Gregory; C: Surface géologique triangulée; D: La même surface, convertie en surface G^1.*

$$\forall \epsilon > 0, \exists a / \quad \forall x, |x - x_0| < a \Rightarrow |\varphi(x) - \varphi(x_0)| < \epsilon$$

En d'autres termes, une fonction φ est continue (nous dirons aussi C^0) en un point x_0 si sa valeur "ne varie pas trop" par rapport à $\varphi(x_0)$ lorsque son argument reste dans le voisinage de x_0. Une fonction est dite de continuité C^k si elle est k fois dérivable et si sa dérivée d'ordre k est continue. Cette définition est parfois trop restrictive dans le cadre de la modélisation 3D, car elle caractérise une **fonction**, qui n'est pas toujours explicitement définie pour les objets représentés. Par exemple, une surface correspondant à l'assemblage de plusieurs carreaux n'est pas définie explicitement par une seule fonction. Il n'est alors pas possible d'utiliser cette définition de la continuité pour caractériser ce type de surfaces (notamment au niveau des raccords entre les carreaux). Pour cette raison, une autre définition de la continuité a été donnée, appelée *continuité géométrique*, et caractérisant des **ensembles de points** plutôt que des **fonctions**. La continuité G^k ainsi définie donne une caractérisation des surfaces plus utile pour la modélisation 3D.

Nous en donnons une idée intuitive, le lecteur se réferrera à [Her87] pour plus de détails :

- G^0 caractérise des surfaces sans trous.
- G^1 correspond a des surfaces pour lesquelles la direction du vecteur normal est continue.
- G^2 dénote la continuité du paraboloïde osculateur (qui caractérise les courbures).

La continuité G^1 est également appelée *continuité visuelle*, car les modèles d'éclairement utilisés en image de synthèse sont calculés à partir du vecteur normal (comme les modèles de Lambert, de Phong, de Whitted ... voir [FvFH90]). Ainsi, une surface G^1 génèrera une image où la couleur et l'ombrage sont continus sur la surface. Une idée faussement répandue consiste à croire que *"les surfaces G^1 sont moins lisses que les surfaces C^1"*. Au contraire, un résultat important est que toute surface de type G^k admet une paramétrisation d'ordre C^k [Her87]. Autrement dit, pour toute surface G^k, il est possible de trouver une fonction C^k qui représente la surface. Ces deux notions caractérisent des objets mathématiques différents: C^k caractérise des **fonctions** alors que G^k caractérise des **ensembles de points**, le terme "surface C^k" est en réalité un abus de langage.

L'abondante littérature traitant du sujet a proposé différentes manières pour assembler des carreaux tout en préservant une continuité géométrique (G^1 ou G^2 le plus souvent) [Alf87, BBG74, Far87, Jen87, Prz95, Sch93]. Pour atteindre la continuité G^1, utiliser des carreaux de Bézier nécessite des polynômes de degré au moins égal à 7, ce qui a été démontré dans [Pip87]. Nous en revenons donc aux problèmes de stabilité numérique et d'oscillations déjà mentionnés. Afin de limiter le degré des polynômes mis en jeu, plusieurs approches ont été proposées. Clough et Tocher [Alf84, Pip87] ont proposé de subdiviser les carreaux en trois sous-carreaux, ce qui a permis de réduire le degré à 4 pour une surface G^1, et même à 3 dans le cas d'une surface représentable par une fonction de type $z = f(x, y)$. D'autres approches ont consisté à ajouter des termes en fraction rationnelle, permettant d'obtenir les degrés de liberté nécessaires à la continuité G^1, sans augmenter le degré des termes. Par exemples, les carreaux triangulaires de Gregory [Gre80, GLH93, SBD86, Du88] et les NURBS (Non Uniform Rational B-Spline) [Far91] ont été beaucoup étudiés dans la littérature et utilisés dans l'industrie. Les carreaux de Gregory ont été généralisés en dimension arbitraire pour des cellules ayant un nombre arbitraire de côtés par Bechmann *et. al.* [BBT97].

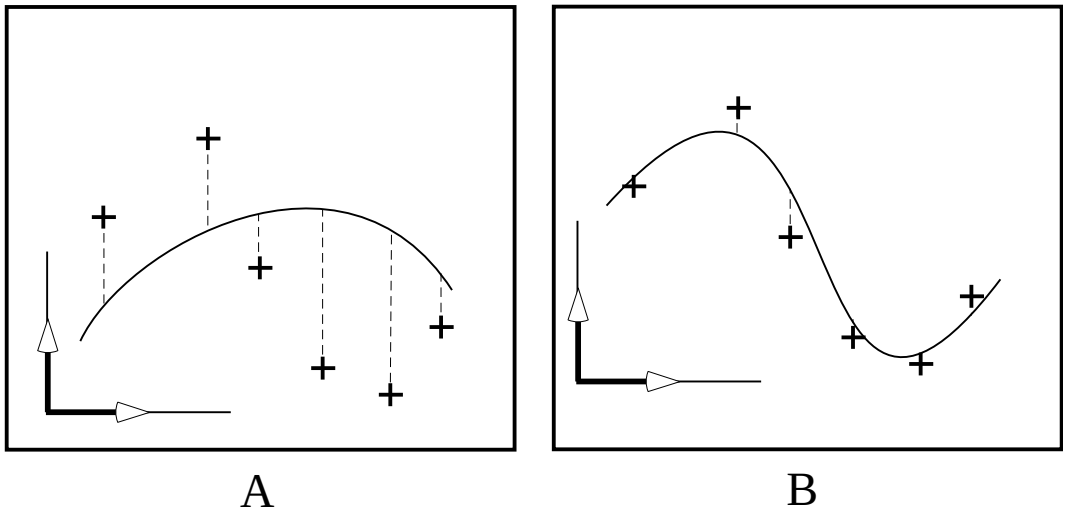


Figure 2.3: *A: Ajustement d'un polynôme de degré 3 à des points de données* (x_i, y_i)*; B: Solution "moindres carrés".*

La Figure 2.2-A montre un carreau de Gregory ainsi que son maillage de contrôle (qui compte plus de noeuds qu'un maillage de contrôle de Bézier de degré trois). La surface de la Figure 2.2-B a été générée en assemblant plusieurs carreaux de Gregory. Cette approche a été étudiée et implantée dans le cadre du projet Gocad dans [Lev95a, SML98, Seg98]. En géologie, de telles surfaces peuvent être utiles pour certains calculs numériques. Par exemple, le tracé de rayons sismiques nécessite une continuité au moins G^1 (voir G^2 dans certains cas) pour les surfaces qui représentent les interfaces entre les couches géologiques. La surface de la Figure 2.2-D a été ainsi générée à partir de la triangulation montrée Figure 2.2-C.

Ajustement aux données

Un autre problème classique de modélisation 3D, appelé *ajustement aux données*, consiste à faire respecter un nuage de points par une surface (voir par exemple [LS86, QS90, SBD86, FLN+90, PS95]). Dans [Toc79], des applications à la géologie (notamment à la cartographie) sont proposées, tirant parti des ordres de continuité fournis par la fonction ajustée aux données. Hoppe *et. al.* ont proposé dans [HDD+92] une méthode de reconstruction de surfaces à partir de nuages de points non structurés. Les surfaces B-Splines ont été utilisées dans [EH96], pour créer des surfaces de topologies arbitraires à partir de nuages de points. Forsey *et. al.* proposent de tirer parti d'une représentation hiérarchique pour avoir une représentation mieux adaptée à la spécificité des données [FB95].

Nous allons donner ici une idée de la théorie mathématique sous-jacente, puis après avoir présenté notre méthode, nous montrerons à la fin du chapitre comment celle-ci fournit une solution discrète de ce problème.

Pour simplifier, nous allons étudier ici le problème pour un polynôme de degré n, dans le cas $y = \varphi(x)$. Ceci peut facilement être adapté au cas paramétrique $x = \varphi^x(u); y = \varphi^y(u)$, qui se résoud en appliquant la méthode à chaque composante. Il est également possible d'adapter la méthode à des fonctions exprimées dans d'autres bases polynomiales (courbes de Bézier, courbes Splines, ...). Cette méthode reste également valable dans le cas des surfaces. Néanmoins, le simple cas d'une fonction polynomiale nous permet d'introduire les notions dont nous aurons besoin par la suite. Étant donné un ensemble de m points de données (x_i, y_i), nous souhaitons déterminer les coefficients a_i du polynôme permettant de passer au plus près de ces points :

$$\varphi(x) = \sum_{i=0}^{n} a_i.x^i$$

Cette condition s'écrit simplement :

$$\forall 0 \leq i \leq m, \quad \varphi(x_i) = y_i$$

En remplaçant $\varphi(x)$ par son expression, nous obtenons le système linéaire suivant :

$$\begin{bmatrix} x_0^0 & x_0^1 & \dots & x_0^n \\ x_1^0 & x_1^1 & \dots & x_1^n \\ \vdots & \ddots & & \\ \vdots & & \ddots & \\ \vdots & & & \\ x_m^0 & x_m^1 & \dots & x_m^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ \vdots \\ y_m \end{bmatrix} \tag{2.5}$$

Comme le nombre de points de données peut être totalement arbitraire, ce système n'est pas inversible en général. La méthode d'ajustement aux données consiste alors à le résoudre *au sens des moindres carrés*. Si nous notons A la matrice des x_i^j, X le vecteur des a_i et B le vecteur des y_i, alors ce système se note $AX = B$. La solution X au sens des moindres carrés est l'ensemble des valeurs a_i qui permettent de minimiser la somme $\sum_i^m (y_i - f(x_i))^2$ (autrement dit, la somme des carrés des longueurs des segments en pointillé de la Figure 2.3). Le vecteur X qui correspond aux moindres carrés est solution du système $A^tAX = A^tB$, où A^t dénote la transposée de la matrice A. Nous vérifierons facilement que la matrice A^tA est une matrice carrée $n \times n$. Ceci n'assure pas pour autant que le système soit inversible. En effet, ce système peut être sous-déterminé et comporter une infinité de solutions. Par exemple, dans le cas où le nombre de points de données est inférieur au degré du polynôme, le système est

sous-contraint et la matrice A^tA n'est pas inversible. Dans le cas où le système est inversible, l'inverse généralisée de Moore-Penrose $A^\dagger$ correspond à sa solution (voir Équation 2.6), et dans le cas contraire, elle permet de sélectionner l'une des solutions parmi l'infinité.

$$A^\dagger = \lim_{\epsilon \to 0} (A^tA + \epsilon I)^{-1} A^t \tag{2.6}$$

En pratique, le calcul numérique de cette limite est peu utilisé, et les numériciens préfèrent plutôt résoudre le système suivant :

$$(A^tA + \epsilon R)X = A^tB \tag{2.7}$$

Dans cette équation, la matrice R est une matrice bien conditionnée, permettant de régulariser le système. Comme le suggère l'équation de Moore-Penrose, il est possible d'utiliser pour R la matrice identité, mais les fonctionnelles de lissage que nous évoquerons dans la section sur le lissage variationnel sont également bien adaptées et couramment utilisées. Le scalaire $\epsilon \in]0, +\infty[$ est défini comme un pourcentage de la trace de la matrice A (ce pourcentage est souvent déterminé de manière empirique). Ce type d'approches peut également s'appliquer à des surfaces définies par l'assemblage de plusieurs carreaux. Nous l'avons expérimenté pour une subdivision de Clough et Tocher dans le cas $z = f(x, y)$ dans [Lev95b]. Dans ce cas, comme la continuité G^k doit être préservée au niveau des raccords entre carreaux, un système global est résolu, permettant de trouver simultanément tous les coefficients définissant chaque carreau. Pour des applications géologiques, comme le nombre de carreaux nécessaires peut être relativement grand, les sytèmes à résoudre croissent rapidement en taille, ce qui cause à la fois de longs temps de calcul et une occupation mémoire importante. Nous verrons plus loin comment notre approche de lissage variationnel discret permet de résoudre le même type d'équations de manière itérative, sur un support discret, et sans nécessiter l'inversion directe d'un grand système.

2.2.2 Surfaces de subdivision

Pour créer des formes compliquées, les carreaux polynomiaux et les NURBs (Non-Uniform Rational B-Splines) précédemment mentionnés sont souvent utilisés. Plusieurs problèmes bien connus apparaissent rapidement lorsque des modèles complexes sont créés en utilisant ce type de représentation.

- Des objets présentant une topologie complexe ne peuvent pas être représentés par un seul carreau;
- il est relativement difficile de connecter plusieurs carreaux pour former des objets de topologie arbitraire tout en maintenant une continuité géométrique acceptable (au moins G^1) au niveau des raccords. Ceci se vérifie en particulier lorsque l'on souhaite *animer* des surfaces, puisque la cohérence des

raccords doit être préservée au cours du temps. Par exemple, dans [DKT98], les problèmes liés à l'animation du personnage principal du film *Toy Story* sont évoqués;

- la modélisation d'objets ayant des bords de forme arbitraire requiert un *découpage*[2] des carreaux, ce qui cause bien souvent des problèmes de stabilité numérique;

- dès que nous utilisons de telles méthodes pour modéliser des formes naturelles, comme celles rencontrées en géologie, le manque de souplesse de ces méthodes réduit sensiblement le réalisme géologique des objets construits.

Une famille de surfaces, dites *surfaces de subdivision*, a suscité beaucoup d'intérêt ces dernières années. Elles ont fait récemment leur preuve dans le domaine de l'animation, en permettant à De Rose *et. al.* la réalisation du court-métrage *Geri's game* [DKT98]. Comme nous le montrons sur les Figures 2.4 et 2.5, en partant d'un maillage $\mathcal{M}^0$ donné, que nous appellerons le *maillage de contrôle*, des règles systématiques sont définies pour le raffiner. Ainsi, le maillage $\mathcal{M}^{i+1}$ est obtenu à partir du maillage $\mathcal{M}^i$ en insérant de nouveaux sommets. La position de chaque sommet de $\mathcal{M}^{i+1}$ est calculée à partir d'une combinaison simple (linéaire), liant localement un ensemble de sommets de $\mathcal{M}^i$. Ces raffinements définissent une suite de maillages $\mathcal{M}^0$, $\mathcal{M}^1$, $\ldots \mathcal{M}^\infty$. Les règles à appliquer pour calculer la position des sommets de $\mathcal{M}^{i+1}$ en fonction de ceux de $\mathcal{M}^i$ sont définies de telle sorte que la suite des maillages $\mathcal{M}^i$ converge vers une surface lisse (en général G^1), que nous notons $\mathcal{M}^\infty$ (voir Figure 2.5-D). Une revue complète de cette famille de surfaces est donnée par Zorin dans [Zor98].

Du point de vue de l'utilisateur de ces méthodes, les surfaces de subdivision présentent plusieurs propriétés utiles, à savoir:

- le maillage de contrôle $\mathcal{M}^0$ peut avoir une topologie arbitraire;

- les règles de raffinement permettant de passer de $\mathcal{M}^i$ à $\mathcal{M}^{i+1}$ sont simples, faciles à implanter et très peu coûteuses en temps de calcul;

- ces méthodes traitent directement des maillages polygonaux, une représentation "naturelle" pour le domaine du graphisme par ordinateur. En effet, la plupart des matériels et logiciels graphiques sont optimisés pour ce type de représentation (voir par exemple [FvFH90]). De plus, la suite de maillages générés $\mathcal{M}^i$ fournit un moyen facile d'implanter la notion de niveaux de détails dans un environnement interactif.

[2] trimming

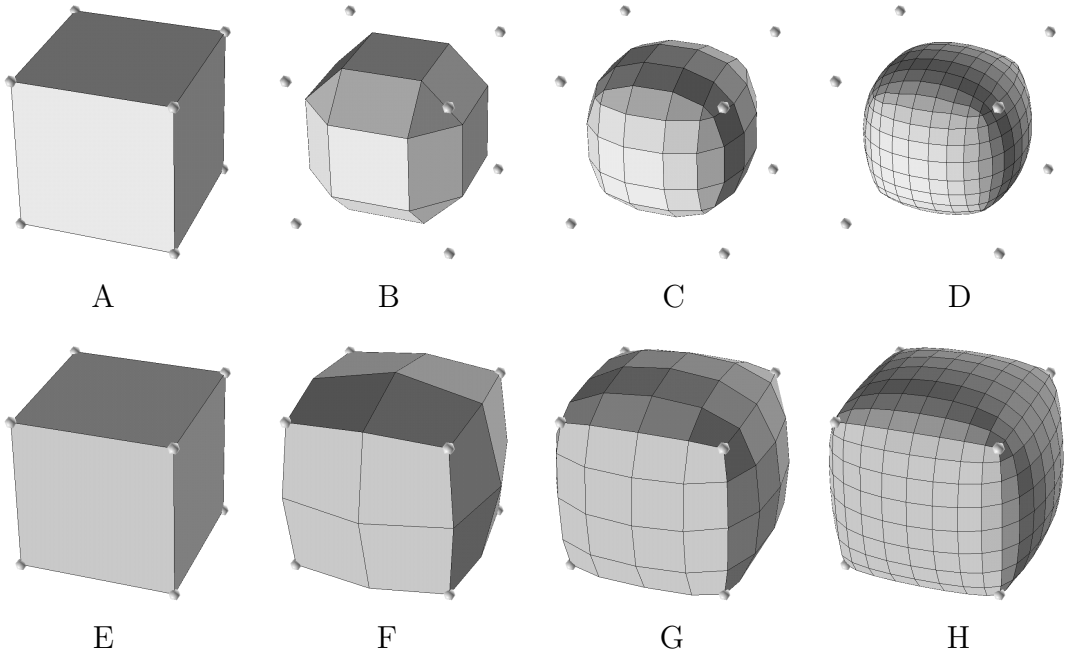


Figure 2.4: *Les schémas de subdivision dits "approximants" produisent des maillages qui ne contiennent pas les sommets initiaux (A-D), à l'inverse des schémas dits "interpolants" (E-H)*

Historiquement, les deux schémas de subdivision mis au point simultanément par Doo et Sabin [DS78], et Catmull et Clark [CC78] en 1978, convergent respectivement vers des B-Splines bi-quadratiques et bi-cubiques. Par la suite, les efforts ont porté sur la caractérisation des surfaces limites et l'analyse de continuité, réalisées pour les schémas classiques dans [Sch96, PR97a]. De nouveaux schémas ont été proposés dans [ZSS96, Kob96b, PR97b] et une théorie générale pour l'analyse de la continuité C^1 - puis C^n - a été peu à peu mise au point [Rei95, Zor97]. Un inconvénient majeur des surfaces de subdivision a longtemps été l'impossibilité d'en avoir une évaluation exacte pour un couple de paramètres (u, v) donné. Ce problème a été résolu dans le cas des surfaces de type Catmull-Clark [Sta98] et la méthode proposée par les auteurs peut facilement être généralisée à d'autres types de surfaces.

Ensuite, la communauté scientifique a orienté ses efforts vers des schémas de subdivision permettant de converger vers différentes familles de surfaces. Ainsi, la subdivision *milieu des côtés* [PR97b] et le schéma de Loop [Loo87] convergent vers des surfaces lisses appelées des *Box Splines*. Ces différentes méthodes correspondent toutes à des schémas *approximants* plutôt qu'*interpolants*, à savoir que les sommets de la surface initiale ne sont pas respectés par la surface lisse (voir Figure 2.4). Du point de vue de l'utilisateur, la manière d'interagir avec ces sur-

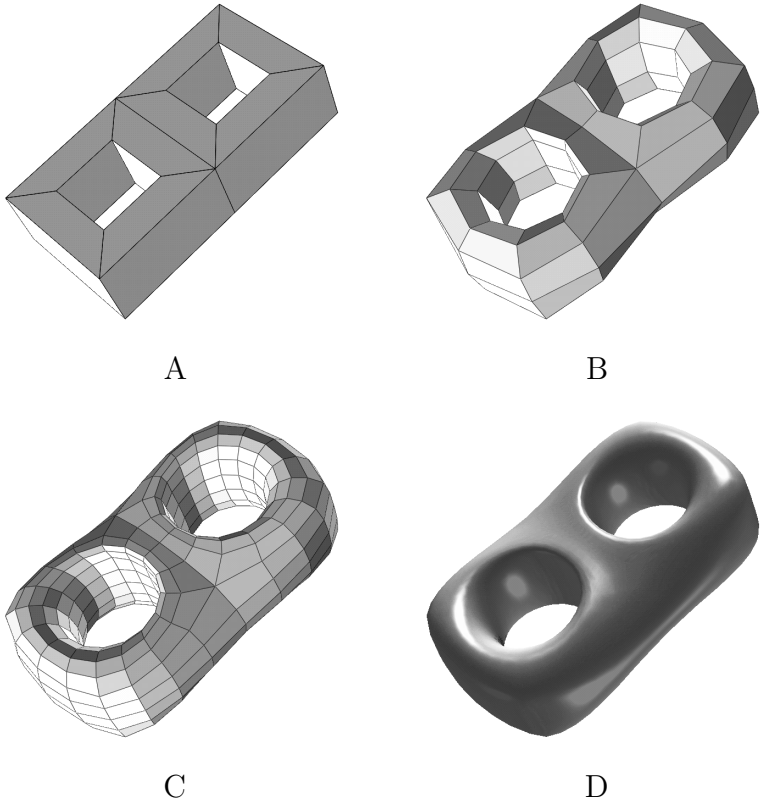


Figure 2.5: *Notre méthode utilisée ici comme un schéma de subdivision interpolante pour les maillages quadrangulaires. A: maillage de contrôle $\mathcal{M}^0$; B: après une étape de raffinement ; C: après deux étapes ; D: surface limite $\mathcal{M}^\infty$.*

faces sera alors similaire à celle obtenue avec des carreaux de Bézier, à savoir que la surface est manipulée indirectement par l'intermédiaire du maillage de contrôle. Les schémas de subdivision interpolants permettent au contraire de manipuler la surface directement, puisque les sommets du maillage de contrôle sont des sommets de la surface finale. Ceci permet une modélisation plus intuitive, mais les théories mathématiques sous-jacentes sont plus complexes. C'est pourquoi ces schémas interpolants sont apparus un peu plus tardivement. Le schéma dit *Papillon*[3] [DLG90], qui permet de lisser des surfaces triangulées, fournit un schéma de subdivision interpolant. Un autre schéma de subdivision interpolant pour des surfaces composées de quadrilatères a été étudié dans [Kob96b]. Dans [CG91], les relations entre les surfaces de subdivision et la modélisation formes-libres[4] ont été traitées.

[3]appelé ainsi à cause de la forme des voisinages mis en jeu, qui évoque un papillon
[4]Free-form design

Les schémas simples de subdivision mis en jeu dans ce type d'approches génèrent des maillages qui ont une topologie régulière. Toutefois, comme le maillage $\mathcal{M}^0$ peut être arbitraire, le nombre de voisins d'un sommet original peut être différent de celui d'un sommet généré. Pour cette raison, les sommets originaux qui ont une connectivité différente sont appelés des *points extraordinaires*. Les connectivités particulières de ces points font que le schéma *Papillon* [DLG90] ne converge pas vers une surface G^1 en ces points. Ce problème a été résolu par Zorin dans [ZSS96], en modifiant le schéma de subdivision au voisinage de ces points extraordinaires. En ce qui concerne les schémas approximants (i.e. qui ne préservent pas les sommets du maillage de contrôle $\mathcal{M}^0$), Rief *et. al.* [Rei95] ont proposé une théorie générale pour analyser la continuité au voisinage des points extraordinaires.

Plusieurs améliorations ont été réalisées afin de fournir à l'utilisateur de ces surfaces de subdivision de plus en plus de fonctionnalités. Par exemple, il est facile de définir certains côtés du maillage de contrôle $\mathcal{M}^0$ comme des angles et des sommets comme étant des coins. Autrement dit, la condition de continuité G^1 est "relâchée" au niveau de ces côtés et sommets. Ceci a été étudié dans le cas de surfaces de Catmull-Clark, afin de modéliser des surfaces de raccordement et des arrondis comme des angles plus ou moins marqués (voir par exemple [DKT98]). La possibilité de règler le niveau de discontinuité du vecteur normal au niveau d'un angle a également été traitée par Sederberg *et. al.* dans [SZSS98], où la notion de NURSS[5] est introduite, comme une généralisation non uniforme des surfaces de Doo-Sabin et Catmull-Clark.

2.2.3 Modélisation variationnelle

La motivation de générer des surfaces ayant des formes visuellement agréables a engendré une autre famille de méthodes, regoupées sous le nom de *modélisation variationnelle* [Gre94, MS92]. Ce type d'approche consiste à générer une surface *lisse* qui vérifie un certain nombre de condition de bords et de contraintes, telles que :

- interpoler des points de données, munis éventuellement de plans tangents à respecter;
- générer une surface de raccord[6] qui interpole les bords d'un ensemble de surfaces;
- lisser une zone particulière d'une surface existante.

Seidel présente dans [Sei98] une vue d'ensemble des méthodes existantes de ce domaine. Dans ce cadre, ces problèmes sont traités en choisissant une classe de

[5] Non Uniform Recursive Subdivision Surface
[6] blending surface

fonctions paramétriques $(u,v) \rightarrow \varphi(u,v) = \{\varphi^x(u,v), \varphi^y(u,v), \varphi^z(u,v)\}$, ayant un nombre de degrés de liberté qui ne correspond pas obligatoirement au nombre *à priori* nécessaire pour honorer l'ensemble des contraintes. Un *critère de lissage*[7] permet de sélectionner une fonction unique dans la classe choisie. Un tel critère de lissage $\mathcal{F}(\varphi)$ est défini comme un scalaire qui caractérise la *qualité*[8] de la surface engendrée par une fonction φ. Une des fonctionnelles les plus souvent utilisées correspond à une approximation de l'énergie de flexion d'une plaque mince (voir Équation 2.8 et par exemple [KHD93, Kal93]):

$$\mathcal{F}_{thinplate_1}(\varphi) = \sum_{\nu \in \{x,y,z\}} \int \left\{ \left(\frac{\partial^2 \varphi}{\partial u^2} \right)^2 + 2 \left(\frac{\partial^2 \varphi}{\partial u \partial v} \right)^2 + \left(\frac{\partial^2 \varphi}{\partial v^2} \right)^2 \right\} dudv \qquad (2.8)$$

Cette équation est parfois simplifiée, en négligeant le terme croisé (Équation 2.9). Suivant le type de surfaces polynomiales utilisées, cette simplification peut rendre les calculs beaucoup plus simples.

$$\mathcal{F}_{thinplate_2}(\varphi) = \sum_{\nu \in \{x,y,z\}} \int \left\{ \left(\frac{\partial^2 \varphi}{\partial u^2} \right)^2 + \left(\frac{\partial^2 \varphi}{\partial v^2} \right)^2 \right\} dudv \qquad (2.9)$$

Cette équation *plaque-mince* a été utilisée sous différentes formes dérivées. Par exemple, il est possible d'ajouter un terme permettant de minimiser l'aire de la surface: $\int \left(\frac{\partial \varphi}{\partial u}\right)^2 + \left(\frac{\partial \varphi}{\partial u}\right)^2 dudv$, ou encore un terme correspondant à la déviation par rapport à une fonction φ^0, décrivant une surface dont l'utilisateur ne souhaite pas trop s'éloigner: $\int (\varphi(u,v) - \varphi^0(u,v))^2 dudv$.

Différentes fonctionnelles de lissage du même type ont été étudiées dans [WW92, CG91]. En ce qui concerne le réalisme physique des phénomènes modélisés, les équations citées ici ne sont que des approximations de l'énergie de flexion. Il est possible d'en définir de plus précises, correspondant à une modélisation réaliste d'un phénomène physique [Gre96]. Ces dernières approches fournissent en général les meilleurs résultats, mais nécessitent des calculs coûteux, en raison de la nature non-linéaire des phénomènes à modéliser. Par exemple, le calcul de l'énergie de flexion exacte met en jeu la courbure de la surface [Gre96]. Lounsbery *et. al.* ont étudié dans [LMD92] différents exemples de ces fonctionnelles à base physique, ainsi que leurs coûts respectifs.

[7] fairness criterion
[8] fairness

Nous proposons ici d'utiliser une forme encore plus simplifiée de l'énergie d'une plaque mince $\mathcal{F}_{\Delta^2}$, mentionnée dans [Sei98], donnée Équation 2.10, où $\Delta\varphi^\nu(u,v)$ dénote le Laplacien de l'une des composantes de la fonction $\varphi(u,v) = \{\varphi^x(u,v), \varphi^y(u,v), \varphi^z(u,v)\}$. Le Laplacien est bien connu dans le domaine de la génération de maillages pour éléments finis, car minimiser ce critère permet d'obtenir des triangles approximativement équilatéraux, donc bien adaptés aux calculs numériques (voir [Han95]).

$$\begin{cases} \mathcal{F}_{\Delta^2}(\varphi) & = & \sum\limits_{\nu\in\{x,y,z\}} \int \mu(u,v).\left\{\Delta\varphi^\nu(u,v)\right\}^2 .dudv \\ \\ \text{avec:} & & \\ \\ \Delta\varphi^\nu(u,v) & = & \frac{\partial^2\varphi^\nu}{\partial u^2}(u,v) + \frac{\partial^2\varphi^\nu}{\partial v^2}(u,v) \end{cases} \tag{2.10}$$

La fonction positive $\mu(u,v)$ permet de simuler une *rigidité* qui varie sur la surface. Cette rigidité variable permet à l'utilisateur de donner plus d'importance au critère de lissage dans certaines zones. Nous montrerons plus loin dans ce chapitre comment cette dernière fonctionnelle peut être discrétisée et utilisée pour lisser des surfaces polygonales.

Le domaine de représentation des méthodes de type modélisation variationnelle est souvent défini comme une classe de surfaces polynomiales par morceaux, ce qui peut poser des problèmes de raccordements dans le cas de topologies complexes. Des résultats récents ont été obtenus, concernant l'application directe des méthodes de modélisation variationnelle à des maillages polygonaux de topologie arbitraire. Ainsi, des schémas de subdivision ont été définis, permettant de converger vers une surface lisse qui minimise une fonctionnelle de lissage. Par exemple, les Splines plaque-mince ont été ainsi traitées dans [WW98]. D'une manière plus générale, Kobbelt [Kob96a] a montré comment des fonctionnelles de lissage peuvent être discrétisées en utilisant des différences finies, et comment de nouveaux schémas de subdivision peuvent être dérivés de fonctionnelles discrétisées de cette manière.

2.2.4 Lissage discret

Malgré les progrès réalisés dans le domaine des surfaces de subdivision, les schémas stationnaires utilisés pour construire de telles surfaces ne permettent pas beaucoup d'interaction avec l'utilisateur. Par exemple, la surface limite $\mathcal{M}^\infty$ peut présenter des oscillations parasites dans les zones où le maillage de contrôle est irrégulier. Même en utilisant les améliorations permettant de modéliser des angles plus ou moins marqués [HDD+94, DKT98, SZSS98], bien souvent, le seul moyen de supprimer les oscillations consiste à modifier le maillage de contrôle. Ceci rend les maillages de contrôle relativement complexes, et nécessite d'autres

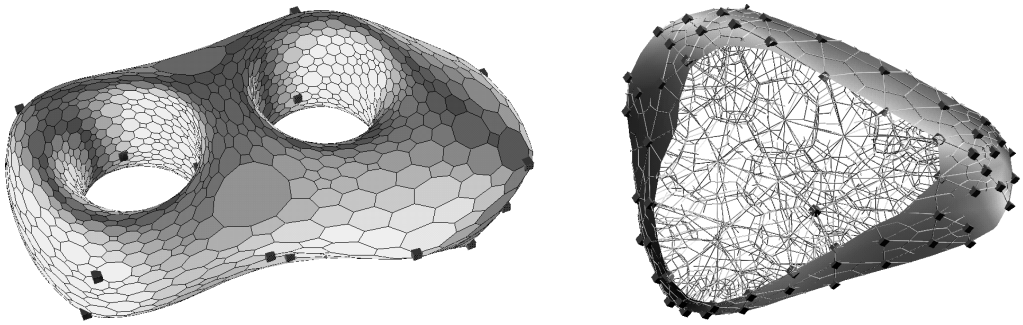
méthodes pour les créer, ce qui repousse le problème de la modélisation. Par exemple, nous pouvons à nouveau citer le personnage *Geri* du court métrage *Geri's Game*, qui a dû être modélisé en argile pour être ensuite saisi en 3D, ceci résultant en un maillage $\mathcal{M}^0$, que nous pouvons voir sur les illustrations de [DKT98], et qui semble plus complexe que nécessaire.

Une autre famille de méthodes, appellée *lissage discret*[9] [Tau95, Kob97, Kob98, Mal89, Mal92, KCVS98, WW94], offre bien plus de flexibilité. Alors que l'approche utilisant des surfaces de subdivision consiste à raffiner itérativement un maillage de contrôle $\mathcal{M}^0$ en utilisant une règle systématique, le lissage discret permet de lisser un maillage arbitraire sans insérer de nouveaux noeuds. Un maillage pour lissage discret a un ensemble de sommets fixés par l'utilisateur, appelés des *noeuds de contrôle*, et les autres sommets sont déplacés de manière à interpoler les noeuds de contrôle. Cette approche peut être considérée comme un sur-ensemble des schémas de subdivision interpolants, puisqu'un maillage de subdivision $\mathcal{M}^i$ peut être vu comme un maillage de lissage discret où les sommets du maillage de contrôle $\mathcal{M}^0$ sont marqués comme noeuds de contrôle. Il est alors facile de supprimer les oscillations parasites en *déverrouillant* les noeuds de contrôle dans les zones concernées.

Dans cette partie, nous introduisons une nouvelle méthode de lissage variationnel discret, fondé sur l'approche D.S.I.[10] [Mal89, Mal92, Mal99]. Notre méthode permet de définir un lissage discret ayant des fonctionnalités habituellement réservées à la modélisation variationnelle et aux surfaces de subdivision. Ainsi, la suite de cette partie détaille les points suivants :

- une méthode de minimisation itérative, qui permet l'édition interactive de surfaces présentant une topologie arbitraire;
- spécifier une rigidité μ variant sur la surface, ce qui permet de choisir les zones de la surface à lisser en priorité;
- modéliser des arrondis ou des angles plus ou moins marqués;
- prendre en compte un ensemble de contraintes linéaires au sens des moindres carrés, ce qui permet par exemple d'ajuster une surface à un ensemble de points de données, munis éventuellement de normales à respecter également.

[9] discrete fairing
[10] Discrete Smooth Interpolation



A B

Figure 2.6: *Lissage de maillages arbitraires. A: Les petits cubes correspondent aux noeuds de contrôle, tous les autres sommets ayant été interpolés. On peut remarquer que la topologie de ce maillage ainsi que la répartition des noeuds de contrôle sont tous deux complètement arbitraires; B: Lissage d'un maillage volumique. Les sommets du bord ont été marqués comme noeuds de contrôle et les sommets internes ont été interpolés.*

2.3 Lissage Laplacien discret

Un maillage triangulé $\mathcal{M}$ est défini comme un triplet $\{\Omega, \mathcal{E}, \mathcal{T}\}$, où $\Omega = \{\alpha_1, \ldots \alpha_N\}$ dénote l'ensemble des sommets de la triangulation, $\mathcal{E}$ dénote l'ensemble des sommets et $\mathcal{T}$ correspond à l'ensemble des triangles. L'ensemble des positions géométriques des sommets de la triangulation peut être considéré comme l'échantillonnage d'une fonction paramétrique continue φ, inconnue, mettant en correspondance l'ensemble des sommets Ω avec un ensemble de points de $\mathbb{R}^3$ $\{\varphi(\alpha) = \varphi(u_\alpha, v_\alpha), \alpha \in \Omega\}$. Étant donné un ensemble de sommets noté $L \subset \Omega$, où la valeur de φ est supposée connue, et l'ensemble de sommets $I = \Omega - L$, où la fonction φ est inconnue, le problème du lissage discret, dans sa forme la plus simple, consiste à choisir aux sommets de I les valeurs de φ qui minimisent une fonctionnelle $\mathcal{F}(\varphi)$. Une telle fonctionnelle fait correspondre à toute fonction φ un scalaire $\mathcal{F}(\varphi)$ qui donne une mesure de la *qualité* [11] de φ, en fonction de critères de lissage. Comme φ est représentée uniquement aux sommets Ω de la triangulation $\mathcal{M}$, la fonctionnelle $\mathcal{F}(\varphi)$ est *estimée* en utilisant uniquement les valeurs de φ aux sommets de $\mathcal{M}$. Dans ce qui suit, nous montrerons comment l'intégrale du carré du Laplacien peut être ainsi approximée et minimisée sur l'ensemble d'une surface triangulée. En utilisant cette approximation du Laplacien, nous définirons un algorithme de lissage itératif permettant de la minimiser. La section suivante montrera alors comment prendre en compte un ensemble de contraintes linéaires au sens des moindre carrés.

[11] fairness

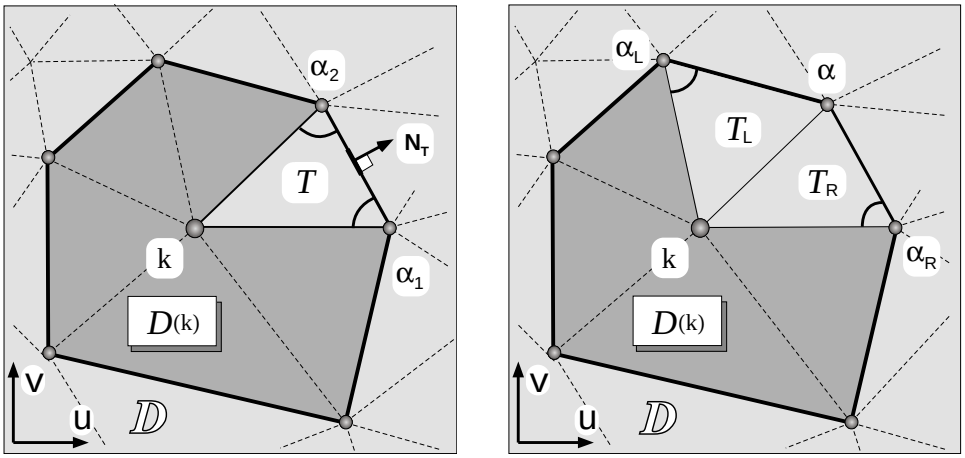


Figure 2.7: *Approximation du Laplacien sur une surface triangulée.*

2.3.1 Laplacien discret d'une surface triangulée

Étant donnée une fonction $\varphi(u,v) = (\varphi^x(u,v), \varphi^y(u,v), \varphi^z(u,v))$, définissant la géométrie d'une surface, nous nous intéressons maintenant au Laplacien de l'une des composantes $\varphi^\nu(u,v)$ de $\varphi(u,v)$, où $\nu \in \{x,y,z\}$.

L'expression du Laplacien $\Delta\varphi^\nu(u,v)$ d'une fonction φ^ν aux paramètres (u,v) est rappelée Équation 2.11 ci-dessous.

$$\Delta\varphi^\nu(u,v) \quad = \quad \left\{ \frac{\partial^2\varphi^\nu}{\partial u^2} + \frac{\partial^2\varphi^\nu}{\partial v^2} \right\}(u,v) \tag{2.11}$$

Notre objectif est maintenant d'approximer cette expression du Laplacien dans le cas d'une surface triangulée. Comme nous pouvons le voir, le Laplacien caractérise des fonctions paramétriques, et pour cette raison, nous devons choisir une paramétrisation de la surface triangulée. Il est possible de construire une paramétrisation globale d'une surface triangulée, comme nous allons le montrer dans le troisième chapitre (voir également [LM98]), mais utiliser cette méthode limiterait le domaine d'application de notre étude à des surfaces homéomorphes à un sous-ensemble d'un disque. Par exemple, il ne serait pas possible de traiter des surfaces fermées.

Une pratique communément utilisée consiste à définir une paramétrisation **locale** (voir par exemple [Tau95, KCVS98, WW92, Flo97]). Étant donné un sommet k, son voisinage $N(k)$ est défini comme l'ensemble des sommets directement connectés à k par une arête, y compris k (le "parapluie" défini par k, comme le nomment Kobbelt *et. al.* dans [KCVS98]).

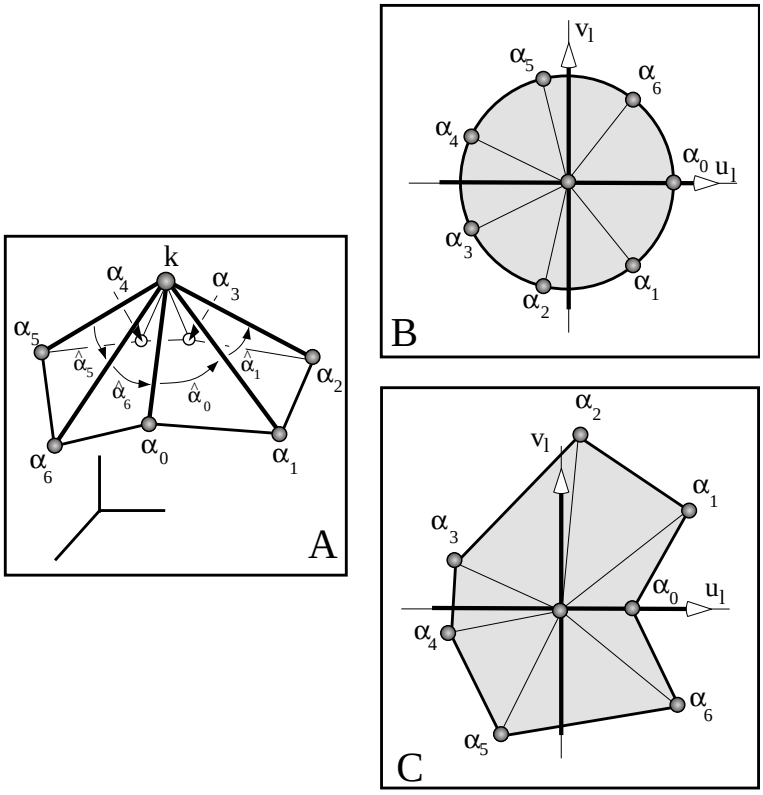


Figure 2.8: *Paramétrisations locales; A: voisinage d'un sommet k; B: paramétrisation uniforme; C: carte exponentielle.*

Une paramétrisation locale consiste à choisir des coordonnées (u_α, v_α) pour tous les voisins $\alpha \in N(k)$ du sommet k. Nous notons $\mathcal{D}(k)$ le polygone défini **dans l'espace paramétrique** par $N(k)$ (voir Figure 2.7-A). Ce polygone $\mathcal{D}(k)$ est appelé le *domaine local* du sommet k.

Une mauvaise idée pour définir une paramétrisation locale consisterait à vouloir projeter les voisins de k sur une estimation du plan tangent au sommet k. Cette approche pourrait causer des recouvrements des triangles dans l'espace (u, v), au niveau des zones irrégulières. La paramétrisation symétrique, utilisée dans [KCVS98], consiste à positionner régulièrement les voisins du sommet k considéré sur un cercle de rayon 1, k étant associé au centre du cercle (voir Figure 2.8-B). Les coordonnées (u, v) locales du sommet α_i sont alors données par l'équation suivante, où n désigne le nombre de voisins du sommet k considéré:

$$\begin{cases} u_{\alpha_i} &= \cos\left(i.\frac{2\pi}{n}\right) \\ v_{\alpha_i} &= \sin\left(i.\frac{2\pi}{n}\right) \end{cases} \tag{2.12}$$

Une autre possibilité est donnée par la *carte exponentielle*[12], aussi appelée *carte géodésique polaire*[13] [WW92, Flo97] (voir Figure 2.8-C). L'idée de la carte exponentielle consiste à préserver les longueurs des côtés, et à pondérer les angles de la surface initiale de sorte que leur somme soit égale à 2π.

$$\left\{ \begin{array}{rcl} u_{\alpha_i} & = & ||\mathbf{p}(k)\mathbf{p}(\alpha_i)||.\cos(\hat{\beta}_i) \\ \\ v_{\alpha_i} & = & ||\mathbf{p}(k)\mathbf{p}(\alpha_i)||.\sin(\hat{\beta}_i) \end{array} \right. \quad \text{avec:} \quad \hat{\beta}_i = 2\pi.\frac{\sum_{j=0}^{i}\hat{\alpha}_j}{\sum_{j=0}^{n-1}\hat{\alpha}_j} \tag{2.13}$$

Une dernière possibilité consiste à utiliser la *pondération préservatrice de formes*[14], définie dans [Flo97].

Dans ce qui suit, nous donnons une estimation du Laplacien pour une paramétrisation **arbitraire**. Entre autres, nous pourrons donc utiliser l'une des paramétrisations évoquées ci-dessus. Nous pouvons remarquer que la paramétrisation symétrique peut être calculée très efficacement, puisqu'en un sommet k donné, elle ne dépend que du nombre de voisins de k, ce qui permet de la tabuler. Au contraire, la carte exponentielle et la pondération préservatrice de formes requièrent des calculs plus coûteux, mais rendent le résultat final moins dépendant de la forme originale des triangles. Étant donné l'une de ces paramétrisations possibles (globale, uniforme, carte exponentielle ou pondération préservatrice de formes, ...), nous montrons plus loin comment estimer le Laplacien sur une surface triangulée et comment le minimiser.

Afin d'estimer le Laplacien en un sommet k donné, la surface triangulée est considérée comme un *échantillonnage* d'une fonction C^2 inconnue $\varphi(u,v) = \{\varphi^x(u,v), \varphi^y(u,v), \varphi^z(u,v)\}$, définie sur le domaine local $\mathcal{D}(k)$ du sommet k (voir Figure 2.7). Si nous considérons maintenant l'une des composantes $\nu \in \{x,y,z\}$, le Laplacien discret $D\varphi^\nu(k)$ de φ^ν au sommet k est défini comme étant une estimation du Laplacien de φ^ν, en utilisant uniquement les valeurs connues de φ^ν aux sommets $\alpha \in N(k)$.

L'approximation que nous effectuons consiste à considérer que dans l'hypothèse où les triangles sont suffisamment petits, le Laplacien est approximativement égal à sa valeur moyenne sur le domaine $\mathcal{D}(k)$, où $|\mathcal{D}(k)|$ dénote l'aire de $\mathcal{D}(k)$:

$$\Delta\varphi^\nu(u_k,v_k) \simeq \frac{1}{|\mathcal{D}(u_k,v_k)|}\int_{\mathcal{D}(k)} \Delta\varphi^\nu(u,v)dudv \tag{2.14}$$

Une expression du Laplacien équivalente à celle donnée Équation 2.11 est

[12] exponential map
[13] geodesic polar map
[14] shape preserving weighting

donnée par $\Delta\varphi^\nu = div(grad\varphi^\nu)$, qui permet de simplifier l'expression précédente en appliquant le théorème de Green Gauss (voir Équation 2.15 ci-dessous). Le vecteur $\mathbf{N}$ dénote la normale au bord $\partial\mathcal{D}(k)$ de $\mathcal{D}(k)$, pointant à l'extérieur de $\mathcal{D}(k)$. En appliquant ce théorème, nous avons ainsi remplacé l'intégrale sur le domaine $\mathcal{D}(k)$ par une intégrale plus simple le long de la courbe $\partial\mathcal{D}(k)$.

$$\begin{aligned} \Delta\varphi^\nu(u_k, v_k) &\simeq \frac{1}{|\mathcal{D}(k)|} \int_{\mathcal{D}(k)} div(grad\varphi^\nu)(u,v)dudv \\ &\simeq \frac{1}{|\mathcal{D}(k)|} \int_{\partial\mathcal{D}(k)} grad\varphi^\nu(s).\mathbf{N}(s).ds \end{aligned} \tag{2.15}$$

Comme la valeur de φ^ν n'est connue qu'aux sommets de $N(k)$, il parait naturel d'utiliser sur chaque triangle l'interpolation linéaire par morceaux, notée Φ^ν, pour approximer φ^ν sur le domaine $\mathcal{D}(k)$.

Le gradient $grad\Phi^\nu(u,v)$ est alors constant sur chaque triangle $T(k,\alpha_1,\alpha_2)$ de $\mathcal{D}(k)$, et nous le notons $grad\Phi^\nu(T)$. Le vecteur normal N est lui aussi constant pour chaque triangle T, et nous le notons N_T. L'intégrale curviligne exprimée le long du bord de $\mathcal{D}(k)$ peut alors être remplacée par la somme suivante sur les triangles $\mathcal{D}(k)$. Nous notons $|E(\alpha_1,\alpha_2)|$ la longueur du segment $E(\alpha_1,\alpha_2)$ dans l'espace paramétrique (u,v) (voir Figure 2.7-A).

$$\begin{aligned} &\Delta\varphi^\nu(k) \quad \simeq \\ &\frac{1}{|\mathcal{D}(k)|}. \sum_{T(k,\alpha_1,\alpha_2)\subset\mathcal{D}(k)} |E(\alpha_1,\alpha_2)|.grad\Phi^\nu(T).\mathbf{N}_T \end{aligned} \tag{2.16}$$

Nous pouvons montrer que l'expression $|E(\alpha_1,\alpha_2)|.grad\Phi^\nu(T).\mathbf{N}_T$ peut être réécrite comme une combinaison linéaire des valeurs de φ^ν aux trois sommets du triangle $T(k,\alpha_1,\alpha_2)$, dont les coefficients sont donnés Équation 2.17:

$$\begin{aligned} &|E(\alpha_1,\alpha_2)|.grad\Phi^\nu(T).\mathbf{N}_T = \\ &\varphi^\nu(\alpha_1).\tau(\hat{\alpha_2}|T) \; + \; \varphi(\alpha_2).\tau(\hat{\alpha_1}|T) \; + \\ &\varphi^\nu(k).\{-\tau(\hat{\alpha_2}|T) - \tau(\hat{\alpha_1}|T)\} \end{aligned} \tag{2.17}$$

Les coefficients $\tau(\hat{\alpha}|T)$ correspondent à la valeur absolue de la co-tangente de l'angle au sommet α du triangle T, **mesuré dans l'espace paramétrique**:

$$\tau(\hat{\alpha}|T(k,\alpha,\beta)) = |cotg(\hat{\alpha})| \tag{2.18}$$

Étant donnés ces coefficients, il est facile de les trier suivant le sommet qu'ils pondèrent. L'Équation 2.19 plus loin donne l'expression des coefficients $v^\alpha(k)$, qui permettent de calculer le Laplacien discret $D\varphi^\nu(k)$.

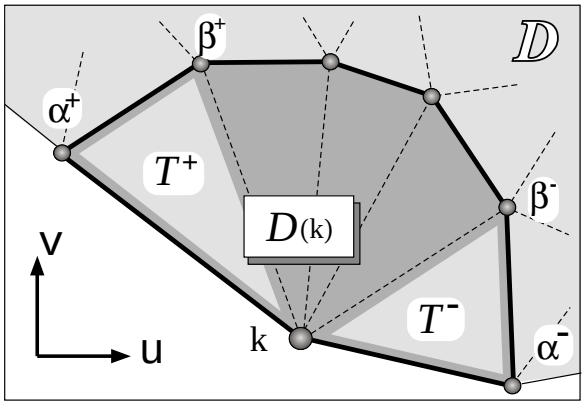


Figure 2.9: *Calcul des coefficients de correction de bord $B^\alpha(k)$ à ajouter quand le sommet k est sur le bord de la surface.*

$$D\varphi^\nu(k) = \frac{3}{\sqrt{|\mathcal{D}(k)|}} . \sum_{\alpha \in N(k)} v^\alpha(k).\varphi^\nu(\alpha)$$

avec:

$$\left\{ \begin{array}{l} \forall \alpha \in N(k) - \{k\}, \\ \\ v^\alpha(k) \;=\; \frac{1}{3}.\frac{1}{\sqrt{|\mathcal{D}(k)|}}.\{\tau(\hat{\alpha_R}|T_R) + \tau(\hat{\alpha_L}|T_L) + B^\alpha(k)\} \\ \\ v^k(k) \;=\; -\sum_{\alpha \in N(k)-\{k\}} v^\alpha(k) \end{array} \right. \tag{2.19}$$

Nous avons défini les coefficients $v^\alpha(k)$ de telle sorte que le Laplacien discret $D\varphi^\nu(k)$ soit égal à la somme $\sum v^\alpha(k).\varphi^\nu(\alpha)$ pondérée par un coefficient (pour l'instant arbitraire) $3/\sqrt{|\mathcal{D}(k)|}$. Nous verrons plus loin comment ce coefficient va se simplifier dans les calculs. Dans l'Équation 2.19, $N(k)$ dénote l'ensemble des sommets directement connectés à k, y compris k. Comme nous le montrons Figure 2.7-B, les triangles T_L et T_R sont définis comme les deux triangles qui partagent le côté $E(k, \alpha)$. Les sommets α_L et α_R sont les sommets opposés à $E(k, \alpha)$ dans T_L et T_R respectivement. Les coefficients $B^\alpha(k)$ restant sont des coefficients de *correction du bord*, dont l'expression est donnée plus loin.

Comme nous le montrons Figure 2.9, le calcul du Laplacien discret $D\varphi^\nu(k)$ en un sommet k sur le bord de la surface requiert la prise en compte de coefficients supplémentaires $B^\alpha(k)$, puisque le bord $\partial D(k)$ du domaine d'intégration comprend deux côtés supplémentaires (k, α^+) et (k, α^-). Ces deux côtés doivent être pris en compte dans le processus de sommation mis en jeu lorsque nous intègrons le long du bord de $\mathcal{D}(k)$ (voir Equation 2.15). Ceci permet de calculer les coefficients de correction de bord $B^\alpha(k)$:

$$\left\{\begin{array}{lcl} B^{\alpha^-}(k) &=& \tau(\hat{\beta^-}|T^-) \quad ; \quad B^{\alpha^+}(k) = \tau(\hat{\beta^+}|T^+) \\ B^{\beta^-}(k) &=& -\left\{\tau(\hat{\alpha^-}|T^-) + \tau(\hat{\alpha}|T^-)\right\} \\ B^{\beta^+}(k) &=& -\left\{\tau(\hat{\alpha^+}|T^+) + \tau(\hat{\alpha}|T^+)\right\} \\ B^{\alpha}(k) &=& 0 \quad \text{partout ailleurs} \end{array}\right. \tag{2.20}$$

Remarque 1: Si nous utilisons une paramétrisation locale symétrique sur un maillage régulier où tous les sommets ont un degré six (c'est à dire six voisins), tous les triangles sont équilatéraux dans le domaine paramétrique. On obtient alors le résultat connu [KCVS98] à un facteur multiplicatif près, à savoir $v^\alpha(k) = 1$ pour $k \in N(k) - \{k\}$, et $v^k(k) = degre(k)$.

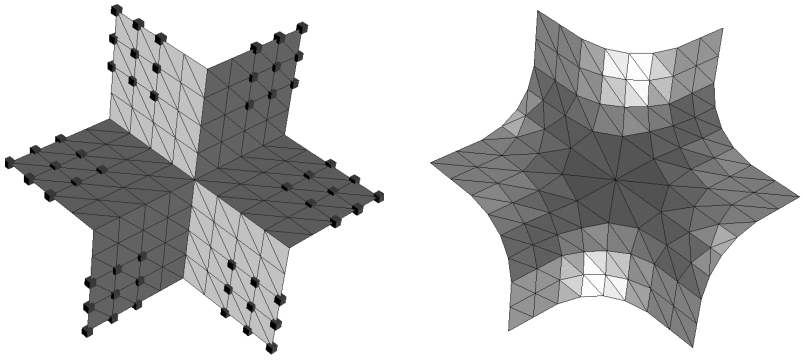
Remarque 2: Le Laplacien discret ressemble beaucoup à un critère de non-distortion, que Hormann et Greiner ont introduit dans [HG98] afin de définir une paramétrisation isométrique. Nous reviendrons plus en détail sur ces notions dans le Chapitre 3, qui traite de la paramétrisation des surfaces triangulées (voir en particulier la Section 3.5.5).

2.3.2 Algorithme itératif de minimisation du Laplacien

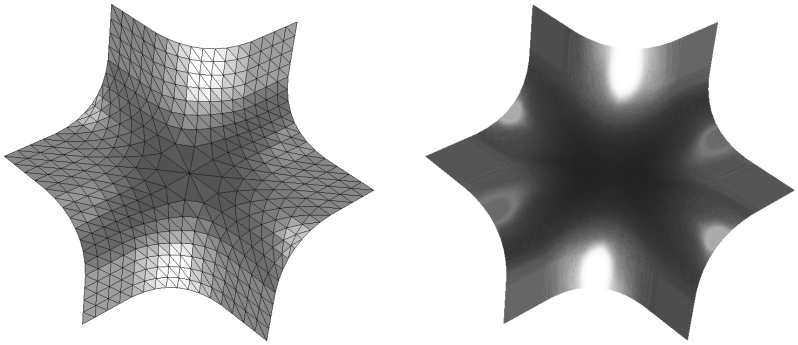
Étant donnée une surface triangulée $\{\Omega, \mathcal{E}, \mathcal{T}\}$ et un sous ensemble de ses sommets $L \subset \Omega$ marqués comme noeuds de contrôle, l'objectif à atteindre est de faire en sorte que les autres sommets de $I = \Omega - L$ interpolent les noeuds de contrôle. Ceci peut être réalisé en minimisant la fonctionnelle discrete $\mathcal{F}_{D^2}$ qui approxime la fonctionnelle $\mathcal{F}_{\Delta^2}$. Le coefficient $1/3$ vient du fait que chaque triangle est pris en compte trois fois lors de l'intégration (une fois pour chacun de ses sommets). Les coefficients positifs **donnés** $\mu(k)$ correspondent à la rigidité de la surface, et permettent à l'utilisateur de moduler le lissage partout sur la surface. La fonction φ^ν dénote une des trois composantes de $\varphi = \{\varphi^x, \varphi^y, \varphi^z\}$.

$$\left|\begin{array}{lcl} \mathcal{F}_{\Delta^2}(\varphi) &=& \sum\limits_{\nu\in\{\alpha,\beta,\gamma\}} \int \mu(u,v).\left\{\Delta\varphi^\nu(u,v)\right\}^2 .dudv \\ \simeq & & \\ \mathcal{F}_{D^2}(\varphi) &=& \sum\limits_{\nu\in\{\alpha,\beta,\gamma\}} \sum\limits_{k\in\Omega} \mu(k).\left\{D\varphi^\nu(k)\right\}^2 .\frac{1}{3}|\mathcal{D}(k)| \end{array}\right. \tag{2.21}$$

En utilisant les coefficients $v^\alpha(k)$ (voir les Équations 2.19 et 2.20), qui permettent d'approximer le Laplacien, nous pouvons donner l'expression suivante pour la fonctionnelle discrète $\mathcal{F}_{D^2}$. Le coefficient $3/\sqrt{|\mathcal{D}(k)|}$ que nous avons introduit arbitrairement dans l'Équation 2.19, qui définit les coefficients $v^\alpha(k)$, se simplifie quand nous développons l'expression de la fonctionnelle discrète $\mathcal{F}_{D^2}$ (Équation 2.21).



A B



C D

Figure 2.10: *Le problème classique de la "selle de singe" (monkey saddle) consiste en la génération d'une surface de raccord connectant six plans qui partent dans différentes directions. A: Maillage de contrôle $\mathcal{M}^0$. Les cubes correspondent aux noeuds de contrôle, alors que les autres sommets peuvent bouger. B: Résultat après application de D.S.I. ; C: Résultat après une étape de subdivision ; D: Surface limite $\mathcal{M}^\infty$.*

$$\mathcal{F}_{D^2}(\varphi) \quad = \quad \sum_{\nu\in\{x,y,z\}} \quad \sum_{k\in\Omega} \mu_k . \left\{ \sum_{\alpha\in N(k)} v^\alpha(k).\varphi^\nu(k) \right\}^2 \tag{2.22}$$

Le minimum de cette fonctionnelle $\mathcal{F}_{D^2}(\varphi)$ en fonction de $\varphi^\nu(\alpha)$ est atteint si la dérivée $\partial\mathcal{F}_{D^2}/\partial\varphi^\nu(\alpha)$ est nulle pour toutes les composantes $\nu \in \{x, y, z\}$. Ceci engendre l'équation suivante (Équation 2.23), appelée *équation D.S.I. locale* [Mal92].

$$\varphi^{\nu}(\alpha) = -\frac{G^{\nu}(\alpha)}{g^{\nu}(\alpha)}$$

avec:

$$\left| \begin{array}{lcl} G^{\nu}(\alpha) & = & \sum\limits_{k \in N(\alpha)} \left\{ \mu(k).v^{\alpha}(k). \sum\limits_{\beta \in N(k)-\{\alpha\}} v^{\beta}(k).\varphi^{\nu}(\beta) \right\} \\ g^{\nu}(\alpha) & = & \sum\limits_{\alpha \in N(\alpha)} \mu(k).\left\{v^{\alpha}(k)\right\}^2 \end{array} \right. \tag{2.23}$$

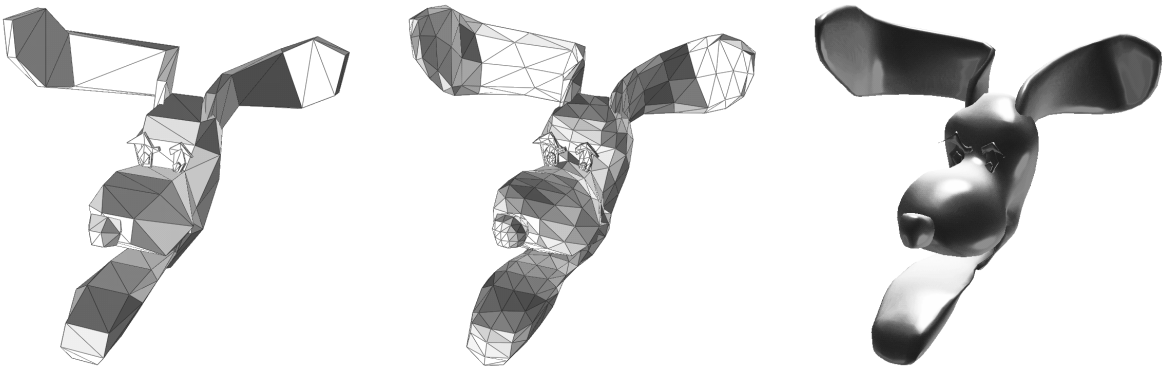
En utilisant l'Équation 2.23, l'algorithme suivant permet de calculer itérativement les coordonnées $(\varphi^x, \varphi^y, \varphi^z)$, tout en minimisant la fonctionnelle discrète $\mathcal{F}_{D^2}(\varphi)$ donnée Équation 2.22. Mallet [Mal89, Mal92] a démontré que cet algorithme converge vers une solution unique, à condition que les relations suivantes soient vérifiées (ce qui est le cas pour les expressions des coefficients $v^{\alpha}(k)$ décrites auparavant).

1. L'ensemble I des noeuds de contrôle est non vide.
2. $\forall k \in \Omega, \forall \alpha \in N(k) \quad v^{\alpha}(k) > 0$
3. $\forall k \in \Omega, \quad v^{k}(k) = \sum\limits_{\alpha \in N(k)-\{k\}} v^{\alpha}(k)$

soit I l'ensemble des sommets où φ est inconnue.
soit $\varphi_{[0]}$ une solution initiale approximée.
tant que (la convergence n'a pas été atteinte) **{**
  pour($\alpha \in I$ **) {**
    pour($\nu \in \{x, y, z\}$**)** $\varphi^{\nu}(\alpha) := -\frac{G^{\nu}(\alpha)}{g^{\nu}(\alpha)}$
  }
}

2.3.3 Lissage discret et surfaces de subdivision

L'algorithme présenté dans la section précédente est itératif, ce qui signifie qu'une bonne estimation de la solution initiale $\varphi_{[0]}$ lui sera très bénéfique en termes de nombre d'itérations nécessaires pour atteindre la convergence. Dans le cadre d'un modeleur interactif, nous pouvons considérer comme solution initiale la surface obtenue après un ensemble d'interventions de l'utilisateur, telles que modifier le maillage, bouger des noeuds de contrôle, ou encore vérrouiller/déverrouiller des sommets. En pratique, l'algorithme D.S.I. converge quasiment en temps réel après intervention de l'utilisateur (une surface de l'ordre de la dizaine de milliers de triangles est réinterpolée en quelques secondes sur une machine de



A B C

Figure 2.11: *Lissage d'un personnage style "dessin animé"; A: Maillage de contrôle $\mathcal{M}^0$; B: Maillage intermédiaire; C: Surface limite $\mathcal{M}^\infty$.*

type SGI R10000, des temps de calculs plus précis sont indiqués plus loin). En se fondant sur cette remarque, nous pouvons considérer D.S.I. comme un schéma de raffinement définissant une classe de surfaces de subdivision (voir Figure 2.11). Le maillage $\mathcal{M}^{i+1}$ est obtenu en subdivisant les polygones du maillage $\mathcal{M}^i$, et en le lissant par D.S.I. À chaque étape, seuls les sommets partagés avec $\mathcal{M}^0$ sont marqués comme noeuds de contrôle. Ceci permet de définir l'algorithme suivant:

soit $\mathcal{M}^0$ le *maillage de contrôle* inital.
marquer tous les sommets de $\mathcal{M}^0$ comme noeuds de contrôle.
tant que (la surface n'est pas assez lisse) **{**
  $\mathcal{M}^{i+1} \leftarrow \mathcal{M}^i$
  subdiviser les polygones de $\mathcal{M}^{i+1}$
  appliquer D.S.I. à $\mathcal{M}^{i+1}$
  $i \leftarrow i+1$
}

Les temps de calcul pour le chien sont indiqués dans le tableau suivant:

	étape 1	étape 2	étape 3
triangles	698	5408	10838
itérations	4	6	6
temps (s)	1	1	8

Comme nous le montrons Figure 2.12, nous pouvons subdiviser les triangles de manière adaptative, afin d'en générer de nouveaux uniquement dans les zones riches en détails, contrairement aux surfaces de subdivision classiques, qui nécessitent de densifier uniformément la totalité du maillage. La Figure 2.12-A

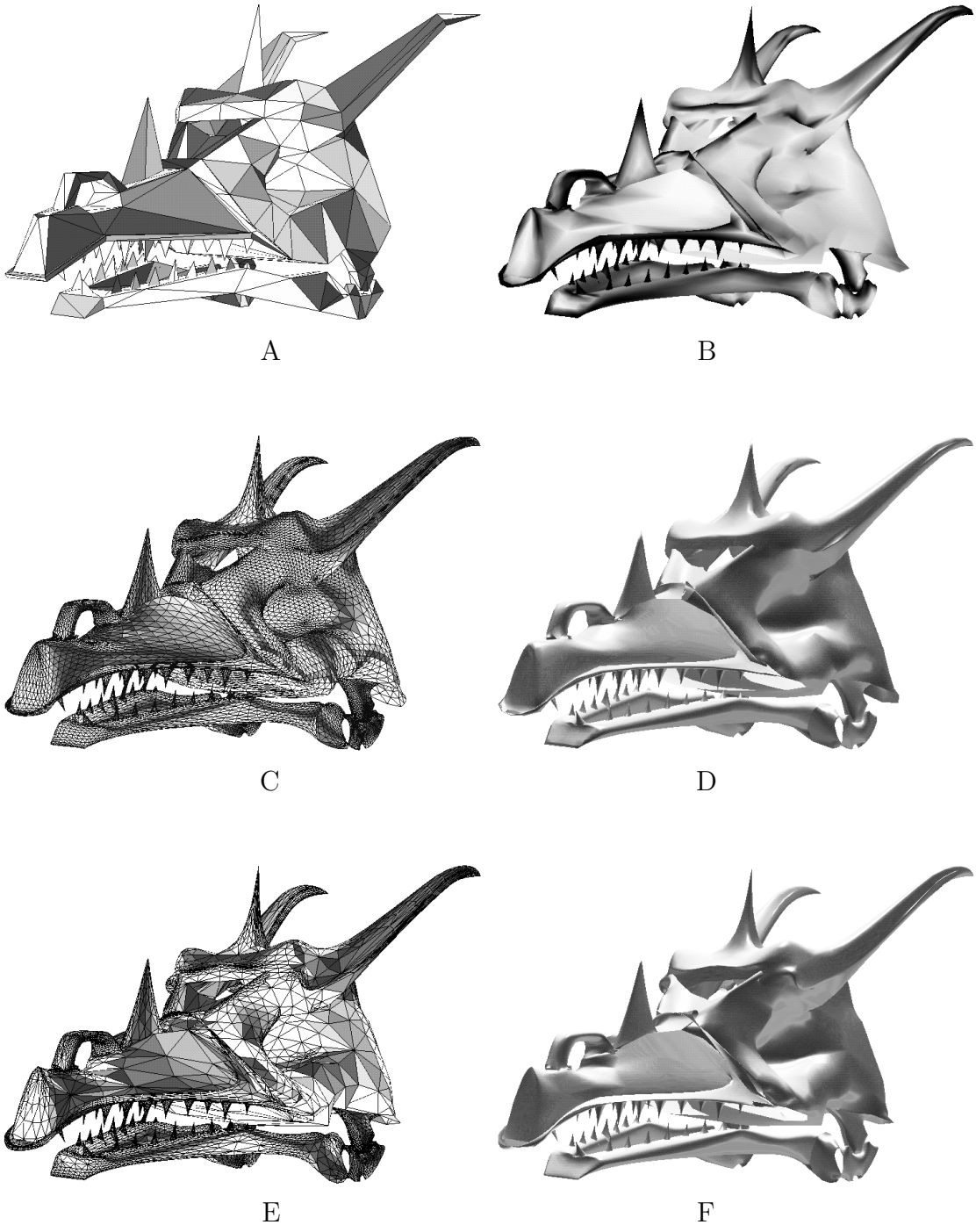


Figure 2.12: *Subdivision adaptative. A: Maillage de contrôle $\mathcal{M}^0$; B: Carte des courbures; C: Maillage $\mathcal{M}^3$; D: Rendu Phong de $\mathcal{M}^3$; E: Maillage $\tilde{\mathcal{M}}^3$, raffiné suivant la courbure; F: Rendu Phong de $\tilde{\mathcal{M}}^3$*

montre un maillage initial $\mathcal{M}^0$ contenant très peu de triangles (513 triangles). Nous montrons Figure 2.12-C le maillage $\mathcal{M}^3$ obtenu après la troisième étape, en subdivisant uniformément chaque triangle de la surface initiale en quatre nouveaux triangles. Cette surface compte 37952 triangles. La même surface est affichée en rendu de Phong [FvFH90] sur la Figure 2.12-D. Afin d'alléger les maillages générés, nous proposons de ne subdiviser que les triangles des zones où la courbure dépasse un certain seuil. La courbure sur une surface triangulée peut être estimée en appliquant par exemple les méthodes proposées dans [Ham] ou dans [Sam96]. Nous en montrons un exemple sur la Figure 2.12-B, où les niveaux de gris représentent les variations de cette courbure estimée (les zones sombres correspondent aux fortes courbures). La Figure 2.12-E montre le maillage $\tilde{M}^3$, obtenu après trois subdivisions adaptatives successives. En comparant les rendus de Phong des maillages $\mathcal{M}^3$ et $\tilde{M}^3$, nous pouvons voir que le même effet visuel est obtenu, pour un nombre de triangles bien moins important (24237 au lieu de 37952). L'algorithme permet d'obtenir ces surfaces au bout de 12 secondes dans le cas de la subdivision systématique, et de 10 secondes dans le cas de la subdivision adaptative. La différence de temps de calcul n'est pas très importante, ceci semble dû au temps de calcul pris par l'évaluation de la courbure mise en jeu dans la deuxième méthode. Ce temps de calcul pourrait être diminué en utilisant une estimation plus grossière de la courbure. Dans l'état actuel, cette méthode reste tout de même intéressante, car elle génère des triangulations plus légères en mémoire que celles obtenues avec une subdivision systématique.

Il convient de préciser que notre méthode ne définit pas à proprement parler une classe de surfaces de subdivision. En effet les schémas de raffinement impliqués dans de telles méthodes, permettant de calculer le maillage $\mathcal{M}^{i+1}$ à partir du maillage $\mathcal{M}^i$, sont déduits de méthodes d'insertions de sommets dans des maillages de contrôles de Splines. Dans de tels schémas, seuls les nouveaux points ont leur coordonnées modifiées, alors que dans notre méthode, tous les sommets peuvent bouger à chaque étape. Dans [ZS98], les différents points définissant une classe de surfaces de subdivision sont rappelés. Même si les surfaces que nous avons définies ne sont pas des surfaces de subdivision, les critères rappelés dans [ZS98] peuvent être utiles pour caractériser leur domaine de modélisation.

1. **Le maillage de contrôle $\mathcal{M}^0$ peut avoir une topologie arbitraire.**

   Il est clair que notre méthode remplit cette condition.

2. **La suite des maillages $\mathcal{M}^i$ récursivement raffinés doit converger vers une surface $\mathcal{M}^\infty$ qui soit lisse (à savoir au moins G^1).**

   L'existence et l'unicité du maillage $\mathcal{M}^i$ **à chaque étape** i a été démontrée dans [Mal89, Mal92]. Notre méthode de minimisation du Laplacien est une

méthode de lissage discret, ce qui signifie qu'à chaque étape i, tous les sommets peuvent se déplacer (mis à part les noeuds de contrôle), alors que dans les schémas stationnaires de subdivision, tous les sommets du maillage $\mathcal{M}^i$ sont "vérrouillés" lors du calcul de $\mathcal{M}^{i+1}$. Ainsi, la convergence et l'unicité pour chaque étape $\mathcal{M}^i$ signifie qu'il est toujours possible d'obtenir une surface lisse à un niveau arbitraire i. La convergence de la suite $\mathcal{M}^i$ a toujours été observée en pratique, mais la preuve formelle de cette convergence reste un problème ouvert. Toutefois, il est facile de montrer que dans le cas où la suite des $\mathcal{M}^i$ converge, alors la fonctionnelle discrète $\mathcal{F}_{D^2}(\varphi_i)$ tend vers la fonctionnelle $\mathcal{F}_{\Delta^2}(\varphi_i)$. Ceci signifie que si la suite des $\mathcal{M}^i$ converge vers une surface $\mathcal{M}^\infty$, alors cette surface $\mathcal{M}^\infty$ minimise le Laplacien de chacune de ses composantes.

3. **L'utilisateur doit avoir la possibilité de marquer certains côtés comme correspondant à des arrondis, des congés, ou à des angles plus ou moins marqués.**

   Comme nous le montrons plus loin, ces détails géométriques peuvent être introduit grâce à une modification locale des poids $v^\alpha(k)$.

4. **Le contrôle doit être local, à savoir, la modification d'un sommet devrait n'avoir une influence que dans une zone limitée autour de ce sommet.**

   Dans notre cas, à proprement parler, le contrôle n'est pas local. En effet, la modification de la position d'un des sommets se répercute sur toute la surface. Toutefois, à partir de la règle de mise à jour locale (voir Équation 2.23), nous pouvons voir que deux rangées consécutives de noeuds de contrôle agissent comme une barrière étanche, tout en définissant des conditions limites de type Hermite (définition du plan tangent à la surface). Ceci suggère une interface utilisateur similaire à celle proposée dans [KCVS98], où l'utilisateur choisit une *zone d'influence* sur la surface avant de la modifier.

5. **Les règles de subdivision doivent être simples et faciles à calculer. On doit pouvoir effectuer efficacement le calcul du maillage $\mathcal{M}^{i+1}$ en fonction de $\mathcal{M}^i$.**

   En choisissant la forme la plus simple pour les coefficients $v^\alpha(k)$, à savoir ceux que nous obtenons pour une paramétrisation locale symétrique: $v^\alpha(k) = 1$ pour $\alpha \neq k$ et $v^k(k) = -degre(k)$, le schéma de mise à jour d'un noeud devient relativement simple. De plus, ces coefficients peuvent être utilisés sur des maillages de topologie arbitraire. Les maillages que nous montrons

Figure 2.6, au début de la section, ont été obtenus en utilisant ces coefficients. Toutefois, dans le cas de surfaces triangulées, les meilleurs résultats sont obtenus si nous utilisons des paramétrisations plus complexes, telles les cartes exponentielles [WW94], ou encore une paramétrisation globale [LM98]. L'algorithme de subdivision itérative peut être vu comme un solveur multi-grille particulier pour l'équation D.S.I. (voir [Hac85] pour plus de détails sur les méthodes multi-grille). Il serait possible d'améliorer les performances en utilisant des approches multi-grille plus sophistiquées. Toutefois, l'algorithme simple que nous avons présenté fournit des performances acceptables permettant une édition interactive sur des surfaces comportant quelques dizaines de milliers de triangles.

La section suivante traite de la modélisation d'arrondis, de congés et d'angles plus ou moins marqués. Nous verrons ensuite comment fournir encore plus de flexibilité à l'utilisateur, en lui permettant d'ajuster la surface à un ensemble de points de données. D'une manière plus générale, nous prendrons en compte, au sens des moindres carrés, un ensemble de contraintes linéaires.

2.3.4 Modélisation d'angles, de congés et d'arrondis

Il est facile de modéliser des angles plus ou moins marqués en *modulant* localement les coefficients $v^\alpha(k)$ qui définissent le critère minimisé par D.S.I. (voir Équation 2.19). Un angle est alors défini comme un ensemble de sommets $\mathbf{C}$ connectés par un ensemble de côtés de la triangulation, formant une courbe polygonale. Nous munissons chaque angle d'un coefficient $s_{\mathbf{C}}$, spécifiant la forme de l'angle. Pour des grandes valeurs de $s_{\mathbf{C}}$, l'angle obtenu pourra devenir arbitrairement pointu. Des valeurs plus faibles de $s_{\mathbf{C}}$ vont engendrer des angles plus émoussés, permettant de modéliser des congés et arrondis. La nouvelle expression des coefficients $v^\alpha_{\mathbf{C}}(k)$ à appliquer à un sommet k lorsqu'on introduit un angle $\mathbf{C}$ est donnée ci-dessous Équation 2.24.

$$\begin{cases} v^\alpha_{\mathbf{C}}(k) &= \frac{1}{1+s_{\mathbf{C}}}.v^\alpha(k) & \text{si } k \in \mathbf{C} \text{ et } \alpha \notin \mathbf{C} \\ v^k_{\mathbf{C}}(k) &= \sum_{\alpha \in N(k)-\{k\}} v^\alpha_{\mathbf{C}}(k) & \text{si } k \in \mathbf{C} \\ v^\alpha_{\mathbf{C}}(k) &= v^\alpha(k) & \text{partout ailleurs} \end{cases} \tag{2.24}$$

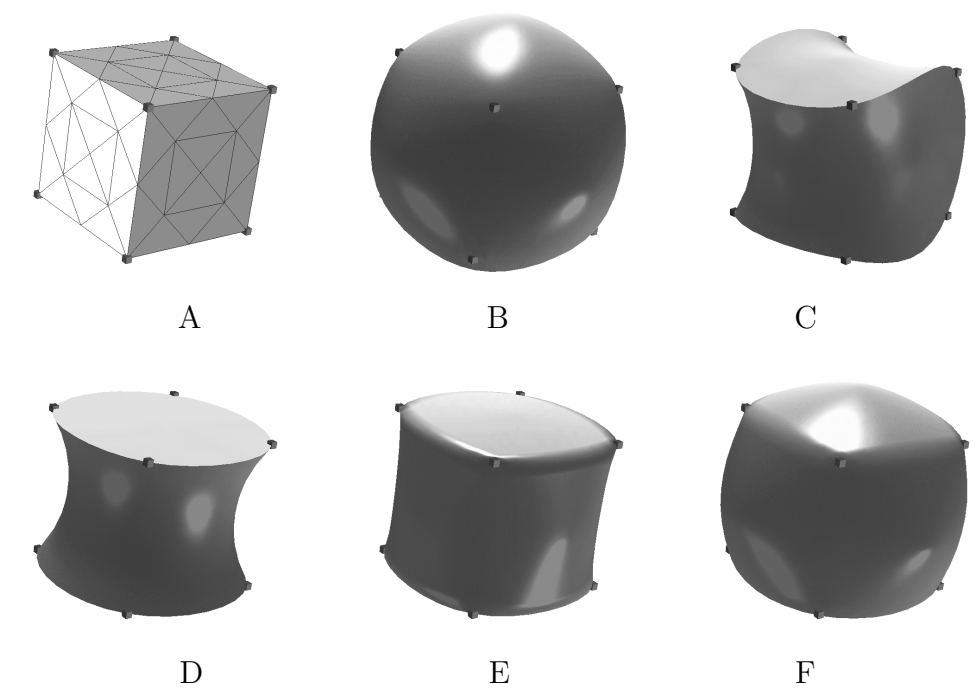


Figure 2.13: *Modélisation d'angles plus ou moins marqués. A: Maillage initial $\mathcal{M}^0$ représentant un cube. Ses huit sommets ont été marqués comme noeuds de contrôle. B: Surface limite $\mathcal{M}^\infty$; C: Un ensemble de côtés a été marqué comme un angles ne devant pas être lissés ; D,E,F: Augmentation progressive de la continuité des angles, permettant de modéliser des arrondis.*

Un coin est alors défini comme un angle **C** qui ne comporte qu'un seul sommet. Quand des angles sont utilisés en même temps que l'algorithme de subdivision présenté dans la section précédente, les sommets insérés sur un côté appartenant à un angle **C** sont ajoutés à **C**. La Figure 2.14 montre comment cette technique permet de modéliser des objets complexes, tels que la main d'un personnage. Les temps de calcul correspondant sont indiqués dans le tableau suivant:

	étape 1	étape 2	étape 3
triangles	2776	11104	44416
itérations	4	8	4
temps (s)	3	8	10

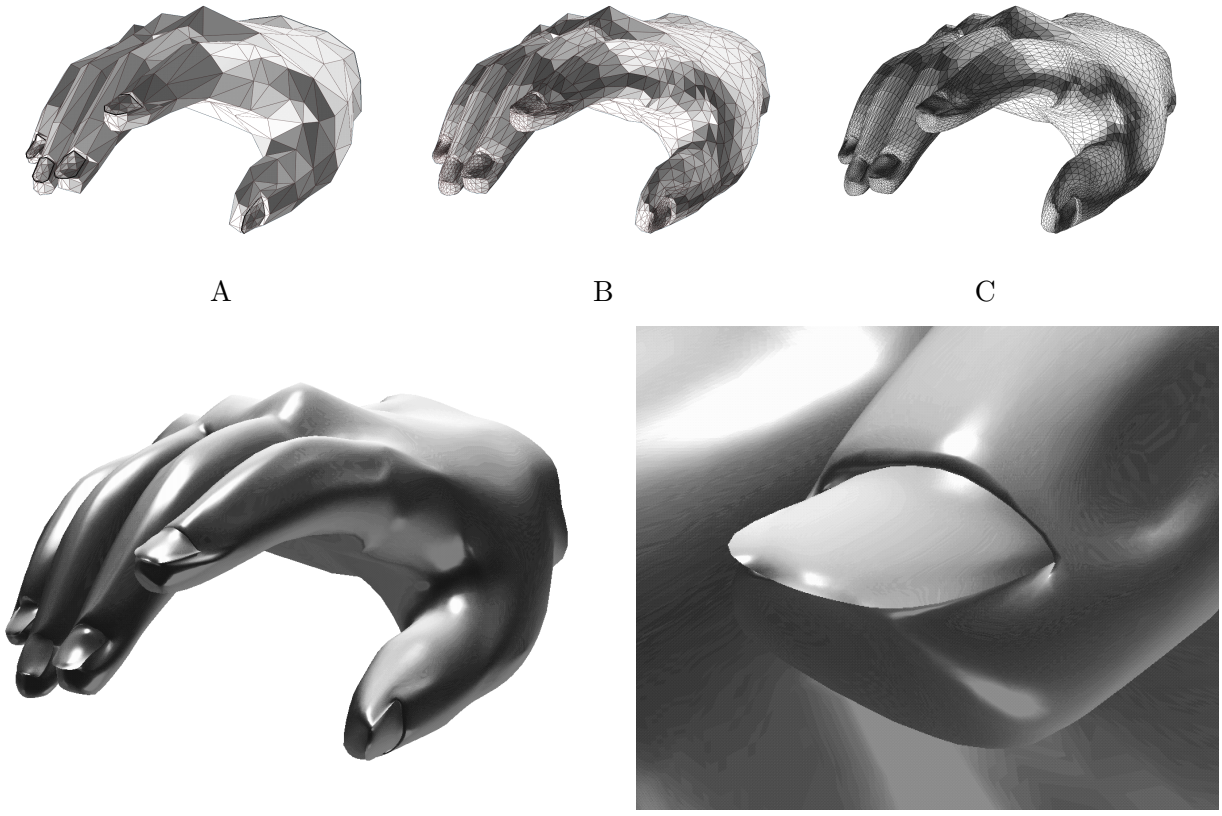


Figure 2.14: *Modélisation d'une main, la condition de lissage est relachée au niveau des bords des ongles; A: Maillage de contrôle $\mathcal{M}^0$; B: Maillage $\mathcal{M}^1$; C: Maillage $\mathcal{M}^2$; D: Surface limite $\mathcal{M}^\infty$; E: Zoom sur le pouce.*

2.4 Ajustement aux données et autres contraintes

La possibilité de modéliser des angles, arrondis et congés permet à l'utilisateur de simplifier son maillage de contrôle $\mathcal{M}^0$ d'une manière notable, en rendant possible la modélisation de zones de forte courbure sans nécessiter de densifier le maillage de contrôle dans ces zones. Toutefois, la seule manière de modifier la géométrie d'une surface modélisée par cette méthode reste d'agir directement sur le maillage de contrôle $\mathcal{M}^0$. Dans le domaine de la modélisation variationnelle, l'ajustement aux données est un problème qui a été largement étudié (voir l'introduction au début du chapitre). Ce problème consiste à modifier une surface pour qu'elle passe le plus près possible d'un ensemble arbitraire de points de données. Nous allons montrer à présent comment étendre notre méthode pour qu'elle puisse fournir ce type de fonctionnalités, tout en offrant la possiblité de prendre en compte des vecteurs normaux éventuellement associés aux points de données.

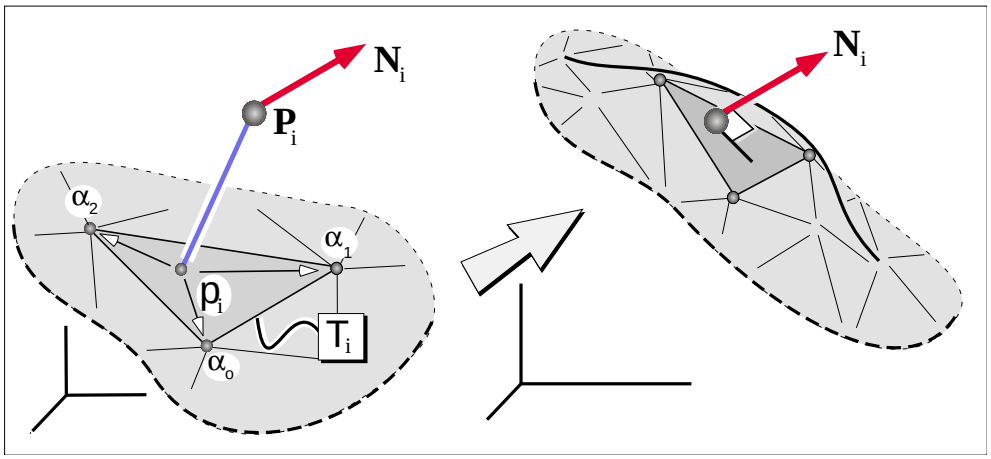


Figure 2.15: *Contraintes de position et de normale. Étant donné un point* $\mathbf{P}_i$*, attirant la surface en un point* $\mathbf{p}_i$ *donné, on souhaite que la surface soit interpolée, tout en bougeant* $\mathbf{p}_i$ *vers* $\mathbf{P}_i$*. En même temps, nous pouvons éventuellement rendre la surface orthogonale en* $\mathbf{p}_i$ *à un vecteur donné* $\mathbf{N}_i$*.*

2.4.1 Ajustement à un ensemble de points de données

Comme nous le montrons Figure 2.15, étant donné un ensemble de points $\mathbf{P}_i$, attirant la surface aux points $\mathbf{p}_i$, l'objectif de l'ajustement aux données consiste à minimiser la somme des carrés des distances entre les points $\mathbf{P}_i$ et les points $\mathbf{p}_i$. De plus, chaque point de donnée $\mathbf{P}_i$ peut être muni d'un vecteur normal $\mathbf{N}_i$ à prendre en compte (le triangle $\mathbf{T}_i$ contenant le point $\mathbf{p}_i$ doit devenir orthogonal à $\mathbf{N}_i$). Nous avons déjà évoqué le problème d'ajustement aux données dans la section 2.2.1, où les surfaces à ajuster étaient représentées par des fonctions polynomiales. Dans notre cas, puisque les surfaces sont représentées sous forme discrète, il est possible d'exprimer les contraintes liées à l'ajustement aux données par des relations linéaires. Ceci nous permet ainsi de définir les *contraintes de position* $c_{\mathbf{P}_i^x}$, $c_{\mathbf{P}_i^y}$ et $c_{\mathbf{P}_i^z}$ et les *contraintes de normale* $c^1_{\mathbf{N}_i}$, $c^2_{\mathbf{N}_i}$ and $c^3_{\mathbf{N}_i}$. Les coefficients $(\lambda_1, \lambda_2, \lambda_3)$ dénotent les coordonnées barycentriques de $\mathbf{p}_i$ dans $\mathbf{T}_i$. Les points $(\alpha_1, \alpha_2, \alpha_3)$ sont les trois sommets de $\mathbf{T}_i$.

A B C

Figure 2.16: *Ajustement aux données. A: Maillage $\mathcal{M}^0$ approximant grossièrement un visage. Les courbes ont été dessinées par l'utilisateur pour raffiner le modèle. B: Surface limite $\mathcal{M}^\infty$. La surface a été* attirée *par l'ensemble des lignes. C: Un maillage intermédiare $\mathcal{M}^i$, ajusté au courbes.*

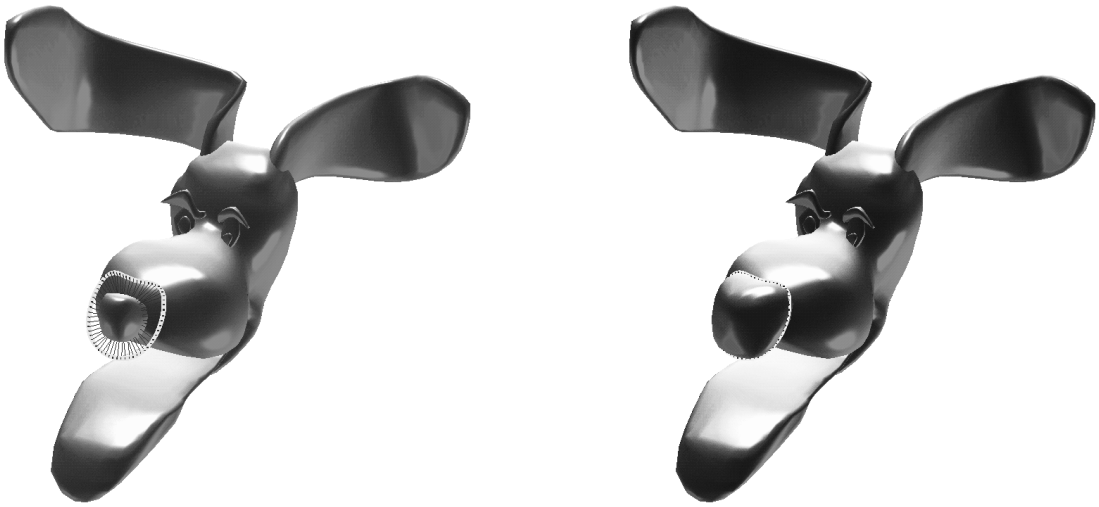
$$\left\{\begin{array}{l} \lambda_1.\varphi^x(\alpha_1) + \lambda_2.\varphi^x(\alpha_2) + \lambda_3.\varphi^x(\alpha_3) = \mathbf{P}_i^x \quad (c_{\mathbf{P}_i^x}) \\ \lambda_1.\varphi^y(\alpha_1) + \lambda_2.\varphi^y(\alpha_2) + \lambda_3.\varphi^y(\alpha_3) = \mathbf{P}_i^y \quad (c_{\mathbf{P}_i^y}) \\ \lambda_1.\varphi^z(\alpha_1) + \lambda_2.\varphi^z(\alpha_2) + \lambda_3.\varphi^z(\alpha_3) = \mathbf{P}_i^z \quad (c_{\mathbf{P}_i^z}) \end{array}\right.$$

$$\left\{\begin{array}{rcl} \sum\limits_{\nu\in\{x,y,z\}} \{(\varphi^\nu(\alpha_3) - \varphi^\nu(\alpha_2)).\mathbf{N}^\nu\} & = & 0 \quad (c^1_{\mathbf{N}_i}) \\ \sum\limits_{\nu\in\{x,y,z\}} \{(\varphi^\nu(\alpha_1) - \varphi^\nu(\alpha_3)).\mathbf{N}^\nu\} & = & 0 \quad (c^2_{\mathbf{N}_i}) \\ \sum\limits_{\nu\in\{x,y,z\}} \{(\varphi^\nu(\alpha_2) - \varphi^\nu(\alpha_1)).\mathbf{N}^\nu\} & = & 0 \quad (c^3_{\mathbf{N}_i}) \end{array}\right. \tag{2.25}$$

Nous pouvons remarquer que ces contraintes combinent **linéairement** les coefficients de φ en un ensemble de sommets de la surface. La forme générale de ce type de contrainte est donnée Équation 2.26 ci-dessous, où ν dénote l'une des trois composantes de φ.

Afin d'équilibrer les équations, il convient de préciser que ces contraintes doivent être *normalisées*, c'est à dire qu'elles doivent satisfaire la relation suivante: $\sum_\alpha \sum_\nu \{A_c^\nu(\alpha)\}^2 = 1$.

$$\sum_{\alpha\in\Omega} \sum_{\nu\in\{x,y,z\}} A_c^\nu(\alpha).\varphi^\nu(\alpha) \quad = \quad b_c \tag{2.26}$$



A B

Figure 2.17: *Ajustement aux données effectué sur le nez du personnage; A: Les points de données attirent la surface dans la direction des segments noirs; B: Résultat obtenu après ré-interpolation.*

En utilisant ce formalisme, les trois contraintes de position $c^x_{\mathbf{P}_i}$, $c^y_{\mathbf{P}_i}$ et $c^z_{\mathbf{P}_i}$ à honorer pour prendre en compte un point de données $\mathbf{P}_i = \{\mathbf{P}^x_i, \mathbf{P}^y_i, \mathbf{P}^z_i\}$ sont données par:

$$\forall \nu \in \{x, y, z\}, \quad \begin{cases} \forall \alpha \in \{\alpha_1, \alpha_2, \alpha_3\}, & A_{c^\nu_{\mathbf{P}_i}}(\alpha_i) = \lambda_i / a \\ \forall \alpha \notin \{\alpha_1, \alpha_2, \alpha_3\}, & A_{c^\nu_{\mathbf{P}_i}} = 0 \\ b_{c^\nu_{\mathbf{P}_i}} = \mathbf{P}^\nu_i / a \\ a = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2} \end{cases} \tag{2.27}$$

Chaque vecteur normal $\mathbf{N}_i$ contraignant un triangle $\mathbf{T}_i = \{\alpha_1, \alpha_2, \alpha_3\}$ engendre trois contraintes de normale $c^1_{\mathbf{N}_i}$, $c^2_{\mathbf{N}_i}$ et $c^3_{\mathbf{N}_i}$. Nous supposons ici que la normale spécifiée est unitaire:

$$\left\{\begin{array}{l} \forall \nu = \{x, y, z\} \\ \quad A^{\nu}_{c^1_{\mathbf{N}_i}}(\alpha_1) = 0 \; ; \qquad A^{\nu}_{c^1_{\mathbf{N}_i}}(\alpha_2) = -\frac{\mathbf{N}^{\nu}_i}{\sqrt{2}} \; ; \quad A^{\nu}_{c^1_{\mathbf{N}_i}}(\alpha_3) = \frac{\mathbf{N}^{\nu}_i}{\sqrt{2}} \\ \quad A^{\nu}_{c^2_{\mathbf{N}_i}}(\alpha_1) = \frac{\mathbf{N}^{\nu}_i}{\sqrt{2}} \; ; \qquad A^{\nu}_{c^2_{\mathbf{N}_i}}(\alpha_2) = 0 \; ; \qquad A^{\nu}_{c^2_{\mathbf{N}_i}}(\alpha_3) = -\frac{\mathbf{N}^{\nu}_i}{\sqrt{2}} \\ \quad A^{\nu}_{c^3_{\mathbf{N}_i}}(\alpha_1) = -\frac{\mathbf{N}^{\nu}_i}{\sqrt{2}} \; ; \quad A^{\nu}_{c^3_{\mathbf{N}_i}}(\alpha_2) = \frac{\mathbf{N}^{\nu}_i}{\sqrt{2}} \; ; \qquad A^{\nu}_{c^3_{\mathbf{N}_i}}(\alpha_3) = 0 \\ \\ \forall \alpha \notin \{\alpha_1, \alpha_2, \alpha_3\}, \quad A^{\nu}_{c^1_{\mathbf{N}_i}}(\alpha) = A^{\nu}_{c^2_{\mathbf{N}_i}}(\alpha) = A^{\nu}_{c^2_{\mathbf{N}_i}}(\alpha) = 0 \\ \\ b_{c^1_{\mathbf{N}_i}} = b_{c^2_{\mathbf{N}_i}} = b_{c^3_{\mathbf{N}_i}} = 0 \end{array}\right. \tag{2.28}$$

Nous allons à présent montrer comment notre méthode peut être modifiée pour prendre en compte les contraintes linéaires ainsi définies.

2.4.2 Lissage variationnel contraint

La fonctionnelle discrète $\mathcal{F}^*(\varphi)$ définie Équation 2.29 ci-dessous caractérise à la fois le lissage de la surface et le degré de violation des contraintes. Le terme $\mathcal{F}_{D^2}$ correspond à l'intégrale du carré du Laplacien (voir Équation 2.22). Nous avons ajouté le second terme $\rho(\varphi)$ pour prendre en compte l'ensemble des contraintes $\mathcal{C}$, où chaque point de données $\mathbf{P}_i$ engendre les contraintes de position $c_{\mathbf{P}^x_i}$, $c_{\mathbf{P}^y_i}$ et $c_{\mathbf{P}^z_i}$ et/ou les contraintes de normales $c^1_{\mathbf{N}_i}$, $c^2_{\mathbf{N}_i}$ et $c^3_{\mathbf{N}_i}$. Comme cela est souvent fait en ajustement aux données, le degré de violation des contraintes $\rho(\varphi)$ est pondéré par un coefficient ϕ appelé *facteur d'ajustement*[15], qui permet à l'utilisateur de règler l'importance relative accordée au respect des contraintes par rapport au terme de lissage. De plus, chaque contrainte individuelle est modulée par un coefficient ϖ_c, qui permet de régler son importance par rapport aux autres contraintes:

$$\begin{array}{rcl} \mathcal{F}^*(\varphi) & = & \mathcal{F}_{D^2}(\varphi) + \phi.\rho(\varphi) \\ \\ \rho(\varphi) & = & \displaystyle\sum_{c \in \mathcal{C}} \varpi_c . \left\{ \left(\sum_{\nu \in \{x,y,z\}} \sum_{\alpha \in \Omega} A^{\nu}_c(\alpha).\varphi^{\nu}(\alpha) \right) - b_c \right\}^2 \end{array} \tag{2.29}$$

Il est possible de minimiser la nouvelle fonctionnelle $\mathcal{F}^*(\varphi)$ ainsi définie, en utilisant un algorithme similaire à celui que nous avons présenté dans la section précédente, dont la convergence vers une solution unique a été également

[15] fitting factor

démontrée par Mallet [Mal89, Mal92, Mal99]. Le schéma de mise à jour d'un sommet doit être complété avec des termes supplémentaires, correspondant à la prise en compte des contraintes (voir les Équations 2.30 et 2.31). Les termes $G^\nu(\alpha)$ et $g^\nu(\alpha)$, correspondant au Laplacien, ne sont pas modifiés par l'introduction des contraintes (leur expression a été donnée Équation 2.23).

$$\varphi^\nu(\alpha) := -\frac{G\nu(\alpha)+\phi.\Gamma^\nu(\alpha)}{g^\nu(\alpha)+\phi.\gamma^\nu(\alpha)}$$

$$\begin{cases} \Gamma^\nu(\alpha) &= \sum\limits_{c\in\mathcal{C}} \varpi_c.\Gamma^\nu_c(\alpha) \\ \\ \gamma^\nu(\alpha) &= \sum\limits_{c\in\mathcal{C}} \varpi_c.\gamma^\nu_c(\alpha) \end{cases} \tag{2.30}$$

Chaque contrainte individuelle c induit les termes explicités Équation 2.31 ci-dessous. Nous pouvons remarquer que le terme $x^\nu_c(\alpha)$ de couplage des composantes est toujours nul pour des contraintes qui ne combinent pas les composantes x, y, z de φ, comme dans le cas des contraintes de position $c_{\mathbf{P}^\nu_i}$.

$$\begin{cases} \Gamma^\nu_c(\alpha) &= A^\nu_c(\alpha).\left\{\sum\limits_{\beta\neq\alpha} A^\nu_c(\beta).\varphi^\nu(\beta) - b_c + x^\nu_c(\alpha)\right\} \\ \\ x^\nu_c(\alpha) &= \sum\limits_{\eta\neq\nu}\sum\limits_{\beta\in\Omega} A^\eta_c(\beta).\varphi^\eta(\beta) \\ \\ \gamma^\nu_c(\alpha) &= \{A^\nu_c(\alpha)\}^2 \end{cases} \tag{2.31}$$

Comme nous le montrons sur les Figures 2.16 et 2.17, cette méthode d'interpolation contrainte peut être elle aussi utilisée avec l'algorithme de subdivision précédemment présenté. Dans le cas de la Figure 2.16 (le visage), la surface comporte 6304 triangles, et a été ajustée aux courbes de contraintes au bout de 26 itérations en 12 secondes. Dans ce cas, au moment où le maillage $\mathcal{M}^{i+1}$ est initialisé à partir du maillage $\mathcal{M}^i$, nous devons affecter les contraintes aux bons sommets. Les sommets en question ainsi que les coordonnées barycentriques associées peuvent être facilement déterminés à partir des coordonnées barycentriques $\lambda_1, \lambda_2, \lambda_3$ des contraintes de position au niveau de $\mathcal{M}^i$.

Remarque:

Dans la section 2.2.1, nous avons évoqué le problème de l'ajustement aux données dans un contexte continu, où les surfaces sont représentées par des fonctions polynomiales. Ce problème consiste à trouver une fonction φ respectant "au mieux" un ensemble de contraintes (à savoir, au sens des moindres carrés). En général, comme nous l'avons indiqué en section 2.2.1, la surface solution du problème de l'ajustement aux données est souvent représentée par une fonction φ, exprimée

dans une base (B_i) de fonctions. La méthode d'ajustement aux données consistera alors à résoudre le système suivant:

$$\begin{aligned} \varphi &= \sum_i X_i B_i \\ (A^t A + \epsilon R)X &= A^t B \end{aligned} \tag{2.32}$$

dont la solution X correspond à la fonction φ cherchée, exprimée dans la base des B_i, et où les contraintes à respecter s'expriment par les relations linéaires $AX = B$. La matrice R est un terme de régularisation, permettant d'assurer que le système a toujours une solution, ou encore que la surface possède certaines propriétés géométriques (comme dans le cas du lissage variationnel). Le scalaire ϵ est souvent choisi de manière empirique, comme une fraction de la trace de la matrice. Ce paramètre est également parfois laissé accessible à l'utilisateur, celui-ci pouvant ainsi faire un compromis entre les critères géométriques correspondant à la matrice R et le critère d'ajustement aux données.

La méthode D.S.I. [Mal92, Mal99] sur laquelle se fonde notre approche peut alors être considérée comme l'équivalent **discret** de cette approche. Dans ce contexte, la fonction φ est exprimée dans une base de fonctions **discrètes** $(B_i)_{1 \leq i \leq N}$, où N dénote le nombre de sommets de la triangulation:

$$\begin{aligned} & B_i : \Omega \rightarrow I\!R \\ & \\ & \forall 1 \leq j \neq i \leq N, B_i(\alpha_j) = 0 \\ & B_i(\alpha_i) = 1 \end{aligned} \tag{2.33}$$

Une contrainte D.S.I. correspond à la relation $AX = B$, et le critère de rugosité joue le même rôle que la matrice R. Le facteur ϕ de respect des contraintes correspond au coefficient ϵ (ou plus exactement, à l'inverse d' ϵ). Pour cette raison, notre méthode peut être qualifiée de *lissage variationnel discret et contraint*.

2.5 Bilan et perspectives

Dans cette partie, nous avons présenté une nouvelle méthode de lissage variationnel, exprimée dans un contexte discret. Notre approche se fondant sur la méthode D.S.I., elle permet de traiter directement des surfaces triangulées. Cette étude a permis de mieux comprendre les relations entre D.S.I. et d'autres types d'approches. Par rapport à des méthodes similaires, notre approche permet de prendre en compte des contraintes linéaires, honorées au sens des moindres carrés. Par exemple, on peut ajuster une surface triangulée à un ensemble de points de données, ayant, ou n'ayant pas de normales spécifiées. Ceci permet d'utiliser, sur des surfaces triangulées de topologie arbitraire, les outils d'édition habituellement disponibles dans le cadre de la modélisation variationnelle [Gre94, MS92, Sei98].

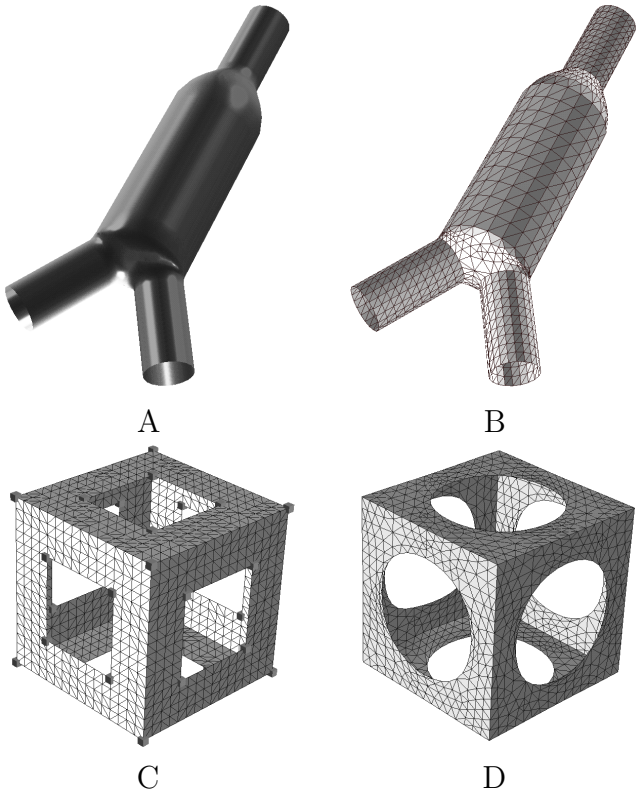


Figure 2.18: *A: Modélisation de surfaces de raccords entre un ensemble de cylindres, surface limite $\mathcal{M}^\infty$; B: Maillage intermédiaire; C: Un cube, dont les côtés ont été marqués comme angles et les sommets comme coins; D: Résultat obtenu en appliquant notre méthode.*

De plus, notre méthode met en jeu des calculs plus simples que dans [WW94], et permet l'utilisation de n'importe quelle paramétrisation pour estimer le Laplacien.

Quand nous l'utilisons comme une méthode de subdivision, comme tout autre approche de lissage discret, notre méthode offre plus de flexibilité que les schémas stationnaires. À n'importe quelle étape du processus, il est ainsi possible de vérrouiller, déverrouiller ou déplacer n'importe quel sommet de la surface afin d'avoir un contrôle plus fin. De plus, comme la règle de mise à jour des sommets est la même quel que soit le sommet considéré, les points extraordinaires ne requièrent plus un traitement particulier. Cette indépendance par rapport à la connectivité des sommets permet également d'envisager un raffinement adaptatif du maillage, à savoir la subdivision des polygones seulement dans les zones de forte courbure.

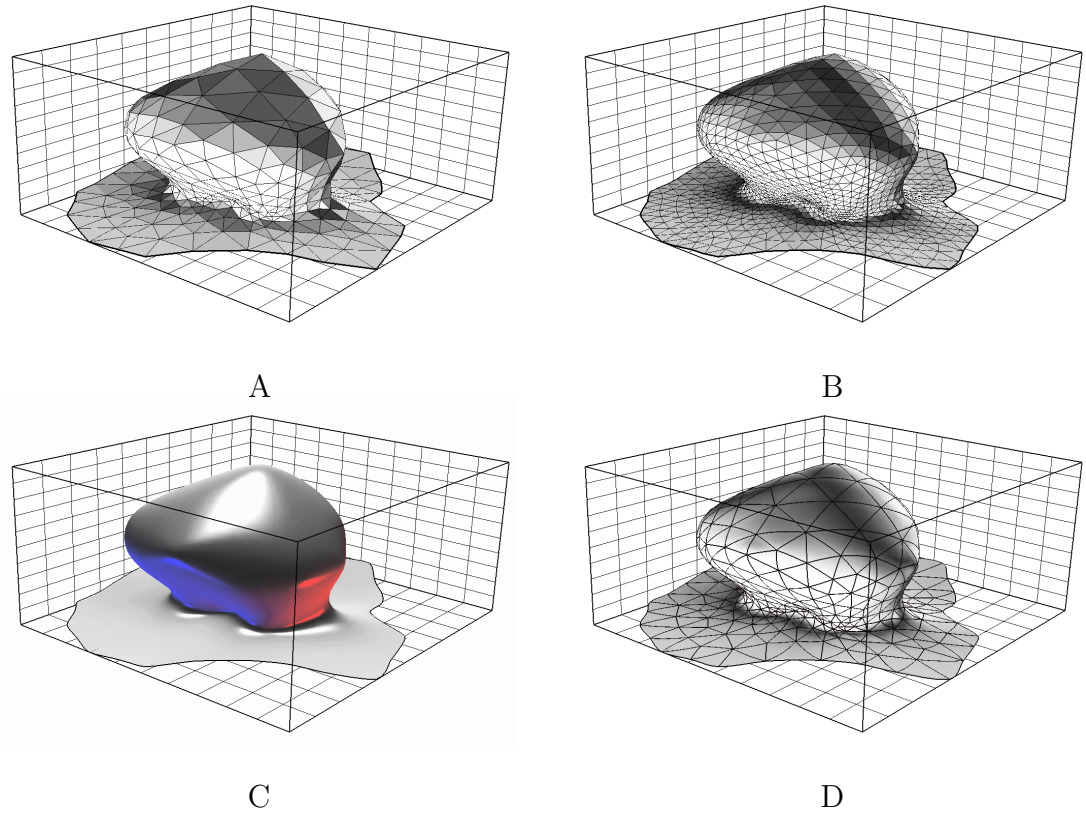


Figure 2.19: *Lissage d'une surface géologique, représentant un dôme de sel. A: maillage de contrôle $\mathcal{M}^0$; B: maillage $\mathcal{M}^1$; C: surface limite $\mathcal{M}^\infty$; D: surface de Gregory obtenue à partir du même maillage $\mathcal{M}^0$.*

Notre méthode fournit un outil interactif de création et d'édition de surfaces triangulées. Les formes relativement complexes que nous montrons Figure 2.18 ont été aisément obtenues en utilisant le modeleur. Pour générer une surface de raccord entre des tuyaux (Figure 2.18-A,B), les méthodes décrites dans [Cnr97] et dans [Nll98] peuvent être utilisées. Cette surface de raccord peut ensuite être lissée par notre méthode.

Nous montrons également des applications possibles à la géologie numérique. Ainsi, la Figure 2.19 montre un dôme de sel représenté par une surface triangulée (Figure 2.19-A) et lissé par notre méthode (Figure 2.19-B,C). Le résultat obtenu est sensiblement identique à une surface modélisée à l'aide de triangles de Gregory (voir Section 2.2), qui peut être construite par exemple à l'aide des méthodes introduites dans [Lev95a, SML98, Seg98]. La courbe du bord des deux surfaces est identique, et correspond à une Spline cubique.

Sur la Figure 2.20, nous présentons un exemple d'application à la modélisation d'horizons faillés. Dans ce cas précis, les données fournies par Gaz de France sont de deux natures différentes: des marqueurs de puits, symbolisés sur la

Figure 2.20-A par des carrés noirs, par lesquels la surface doit passer, et des données sismiques, représentées par des lignes en pointillés, que la surface doit honorer. Dans notre cas, comme nous pouvons le voir sur les points de données, la surface présente des discontinuités, qui correspondent à des failles géologiques. Pour représenter ces failles, les approches classiques consistent à appliquer des méthodes d'intersections de surfaces, afin de découper la surface représentant l'horizon par un ensemble de surfaces qui correspondent aux failles, à la manière d'un emporte-pieces. Cette dernière opération est relativement coûteuse, et présente l'inconvénient de générer des triangles de formes aplaties, qui peuvent poser des problèmes de stabilité numérique. Ici, nous proposons une approche différente, fondée sur la possibilité de relâcher la condition de continuité G^1 dans certaines zones. Ainsi, à partir d'une solution automatiquement déterminée à partir d'une carte de courbures (Figure 2.20) et de connaissances géologiques de la zone étudiée, l'utilisateur peut choisir les courbes qui correspondrons aux traces de failles (représentées par des points blancs sur la Figure 2.20-E). En appliquant une dizaine d'itérations supplémentaires, le résultat de la Figure 2.20-F est obtenu. Sur cette figure, nous pouvons voir que les failles correspondent bien à des zones de rupture de la continuité G^1. Ceci donne plus de degrés de liberté pour honorer les données dans ces zones.

Nos recherches futures vont s'orienter vers les aspects concernant la multi-résolution. Dans ce dernier type d'approches, telle celle présentée dans [KCVS98], il serait intéressant d'étudier l'influence des contraintes linéaires à un niveau de résolution arbitraire. Par exemple, il serait alors possible de lisser une surface riche en détails de texture, et de l'ajuster à des points de données sans gommer ces détails. Ce type de représentation est également lié à la compression de maillages [Dee95]. Il semble possible d'utiliser notre méthode comme un *prédicteur*, ce qui signifierait que seule la différence de position entre la surface interpolée et les données devrait être stockée. Si nous utilisons cette approche comme une description progressive dans un environnement en réseau, l'algorithme itératif permettra une transition continue entre les différents niveaux de détail au fur et à mesure de leur chargement.

L'approximation d'autres familles de fonctionnelles constitue également un aspect important, que nous étudierons dans des recherches futures. Dans ce chapitre, nous nous sommes concentrés sur le cas particulier du Laplacien, qui permet d'obtenir des surfaces relativement lisses. Si nous souhaitons des formes encore plus lisses, nous pouvons envisager d'approximer des fonctionnelles d'ordre supérieur, comme le Hessien (voir [Car76]). Plusieurs fonctionnelles ont ainsi été étudiées et approximée sur des maillages discrets par Brakke [Bra92b, Bra92a] (voir aussi [PP93]), ce qui lui a permis la réalisation du logiciel Evolver[Evo]. D'autres approches ont consisté à minimiser la courbure, comme celle proposée par Leger dans [LMTR94]. De telles approches permettent d'obtenir des surfaces

plus lisses, mais le temps de convergence est beaucoup plus important (20 minutes pour une surface comptant 500 triangles, dans le cas de la méthode proposée par Leger). Il serait intéressant de voir dans quelle mesure ce type de fonctionnelles pourrait être approximé par notre méthode, en utilisant éventuellement des voisinages plus grands autour de chaque sommet. Ceci fournirait, en plus des fonctionnalités offertes par le logiciel Evolver et par la méthode de Leger, la possibilité de prendre en compte des contraintes linéaires.

Notre méthode permet d'approximer le Laplacien, en utilisant n'importe quelle paramétrisation locale pour un sommet et ses voisins. Le choix de cette paramétrisation locale en lui-même peut être un problème complexe, et l'expérience nous a montré que les meilleurs résultats étaient obtenus avec les cartes exponentielles. Toutefois, dans la littérature, il manque toujours une justification rigoureuse pour le choix de ces paramétrisations locales. Dans le troisième chapitre, nous montrerons une méthode de paramétrisation globale pouvant être utilisée, si la surface est homéomorphe à un sous-ensemble d'un disque. Dans le cas général, où il n'est pas possible de définir une paramétrisation globale, nous souhaiterions définir une manière mieux justifiée pour choisir les paramétrisations locales. Si les maillages utilisés sont des triangulations de Delaunay, la méthode d'interpolation des *voisins naturels* [Sib80, Sib81] fournit des fonctions de continuité C^1, présentant certaines propriétés utiles pour les méthodes d'éléments finis [BS95, Suk98, SMB98, SM99, Tra94, Wat94]. Nous pouvons alors penser à définir les coefficients $v^\alpha(k)$ intervenant dans notre méthode de manière à faire approximer par une surface triangulée l'interpolant *voisins naturels*, ce qui pourrait fournir une manière "raisonnable" de choisir les paramétrisations locales.

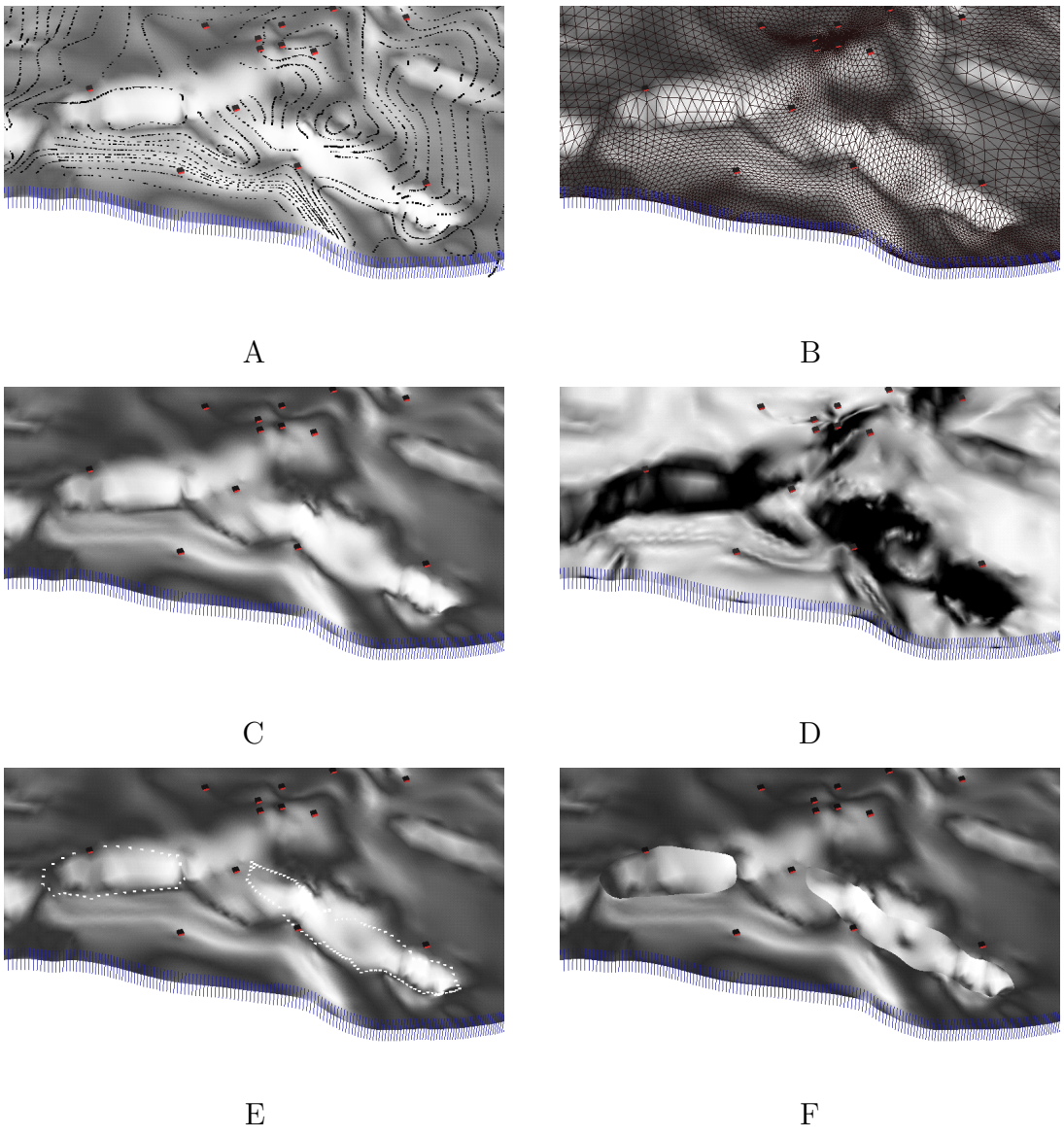
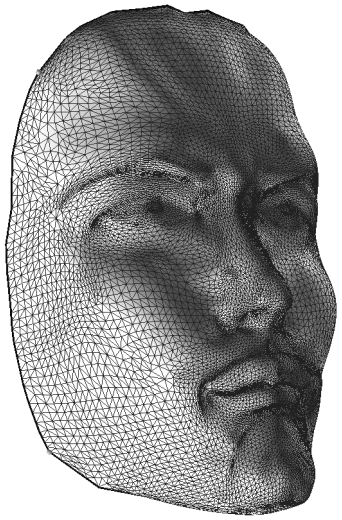
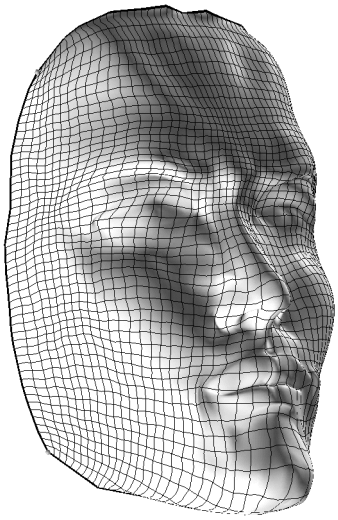


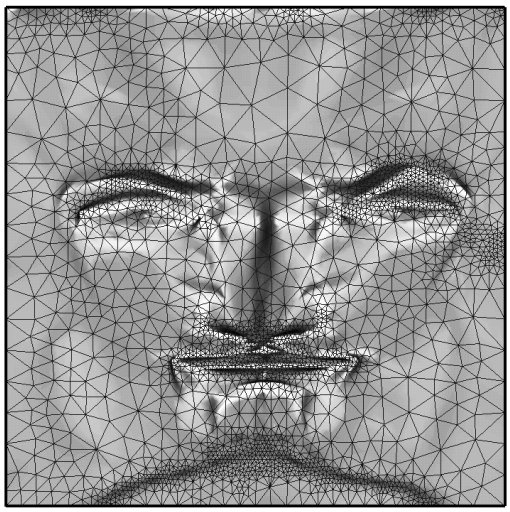
Figure 2.20: *Modélisation d'un horizon géologique faillé (données fournies par Gaz de France). A: Surface adaptée aux données. Les carrés noirs correspondent aux données de puits, et les lignes en pointillés représentent les données sismiques. B: La surface et son maillage, adaptée aux données. C: La surface limite est G^1 partout, y compris aux alentours des failles, ce qui limite son réalisme géologique. D: une cartographie de la courbure maximale de la surface aide à localiser les failels. E: l'utilisateur a identifié les traces des failles, en éditant un résultat obtenu à partir de la carte des courbures. F: surface obtenue en tenant compte des traces de failles, où la continuité G^1 est relachée.*

Chapitre 3

Paramétrisation contrainte







3.1 Introduction

Dans ce chapitre, nous introduisons un ensemble de nouvelles méthodes permettant de construire sous contraintes une paramétrisation d'une surface triangulée complexe. Ces méthodes vont nous permettre de construire des grilles 3D à partir de surfaces triangulées, afin de définir une représentation précise des propriétés physiques associées aux modèles du sous-sol. Ce type de représentation permet de mettre en œuvre un ensemble de simulations et de calculs numériques, dans l'objectif d'optimiser le stockage de gaz au sein des réservoirs naturels. La Section 3.5.4 donne plus de détails en ce qui concerne ce cas particulier, et présente un cas d'étude réel. Dans ce contexte, un modèle est défini par un ensemble de primitives, décrivant de manière précise la géométrie des couches géologiques. Afin de permettre certaines simulations numériques, ce modèle est ensuite peint par des valeurs, qui représentent des propriétés physiques (par exemple la porosité de la roche). Les primitives géométriques sont elles-mêmes constituées par un ensemble d'éléments discrets, comme les triangles d'une surface triangulée, ou encore les cellules hexaédrales d'une grille 3D structurée. Lorsque des valeurs de propriétés doivent être associées au modèle, elles sont le plus souvent stockées au niveau des noeuds de la discrétisation, ce qui impose que la finesse de cette discrétisation corresponde à la fois à la taille des détails géométriques et à la variabilité des propriétés à représenter. En utilisant ce type de représentations, le maillage doit donc être raffiné en fonction de la propriété présentant les détails les plus fins, ce qui cause à la fois une augmentation considérable de la consommation mémoire et un ralentissement notable des algorithmes agissant sur la structure. Pour cette raison, nous proposons une autre manière de représenter les propriétés physiques, en *découplant* leur support de celui utilisé pour la géométrie du modèle. Ce découplage est réalisé à l'aide d'une fonction appelée une *paramétrisation*, mettant l'objet en correspondance avec un *domaine paramétrique*, où les propriétés peuvent facilement être représentées avec une précision arbitraire. Cette notion est connue dans le monde du graphisme par ordinateur sous le nom de *placage de textures*, les propriétés physiques en question correspondant à la couleur de l'objet.

Cette partie reprend les résultats que nous avons déjà présentés dans [LM98], en approfondissant certains aspects. Par exemple, nous verrons comment les deux principales méthodes de paramétrisation connues peuvent être exprimées dans le formalisme que nous avons défini. Nous avons également étudié des applications de notre méthode dans d'autres domaines que celui de la représentation des propriétés et du placage de textures, ainsi qu'une caractérisation quantitative du résultat obtenu en ce qui concerne la répartition des distortions. Les coordonnées (u, v) de texture définies en chaque sommet de la triangulation sont déterminées par un algorithme itératif d'optimisation, qui permet d'honorer un ensemble de contraintes exprimant la minimisation des distortions. Nous

étudierons également une manière de caractériser les déformations introduites par une paramétrisation. Par rapport aux méthodes classiques d'optimisation globale, notre méthode offre plus de flexibilité, en permettant, par exemple, à l'utilisateur de spécifier les zones de la surface où les distortions doivent être minimisées en priorité. L'approche modulaire décrite dans cette partie permet de définir une paramétrisation adaptée à chaque besoin spécifique des utilisateurs. Par exemple, sur la surface, il est possible d'aligner la texture avec un ensemble de courbes iso-paramétriques quelconques. De plus, la paramétrisation peut être rendue continue à travers les éventuelles "déchirures" de la surface. Il est alors possible de construire en une seule fois une paramétrisation pour des surfaces complexes qui auraient dû être découpées en plusieurs parties si l'on avait utilisé une approche classique. D'autres contraintes peuvent être prises en compte par notre méthode, dès lors qu'elles peuvent être exprimées par des relations linéaires (ou linéralisables). Cette méthode a été intégrée dans le noyau du logiciel Gocad, dédié à la modélisation d'objets géologiques. Dans ce contexte, comme nous le montrerons plus loin, différentes applications de la méthode sont exploitées, comme la possibilité d'effectuer des calculs dans l'espace (u, v) ainsi défini, la génération de grilles présentant les propriétés requises par l'analyse en éléments finis, la conversion de surfaces triangulées en surfaces polynomiales, ou encore le dépliage de couches géologiques. Notre méthode pourrait également avoir un impact important dans le domaine du graphisme et des systèmes de dessins en 3D[1].

La technique du placage de textures[2] est beaucoup utilisée pour améliorer la richesse visuelle des images générées par ordinateur. Les scènes modélisées sont constituées d'un ensemble de surfaces trop pauvres en détails pour représenter des objets d'une manière réaliste, et ce en raison à la fois du coût mémoire et du temps nécessaire à la réalisation de modèles détaillés. Par exemple, il ne serait pas raisonnable de vouloir modéliser toutes les écailles de la peau d'un dinosaure. Le placage de texture consiste en la mise en correspondance de chaque surface 3D avec une image 2D à l'aide d'une fonction appelée une *paramétrisation*. Une telle paramétrisation associe à chaque point d'une surface une paire de coordonnées (u, v) désignant un pixel de la texture. Par exemple, pour une sphère, la latitude et la longitude définissent une paramétrisation triviale. Cette technique a été introduite pour la première fois par Catmull [Cat74], et appliquée à des surfaces bicubiques en utilisant un algorithme récursif de subdivision. Malheureusement, ces méthodes sont connues pour engendrer de fortes distortions. Par exemple, dans le cas de la sphère mentionné précédemment, l'image sera fortement altérée aux alentours des pôles.

[1] 3D paint systems
[2] texture mapping

Des approches pour minimiser ces distortions ont premièrement été tentées dans [PD85] et [BS86], en séparant le procédé en deux étapes: la texture est déjà plaquée sur une surface intermédiaire simple, telle une boîte ou un cylindre, pour laquelle le placage de texture est trivial. Cette surface intermédiaire est alors projetée sur l'objet à texturer. Le choix de la surface intermédiaire ainsi que son orientation ont tous deux une grande influence sur le résultat final, ce qui signifie qu'une grande part d'interaction avec l'utilisateur s'avère nécessaire. De plus, pour des surfaces présentant des formes complexes, il peut être très difficile d'obtenir un résultat satisfaisant.

Une autre idée consiste à considérer qu'affecter des coordonnées de texture à une surface ou bien la déplier sont deux opérations équivalentes. Une telle technique est décrite dans [SMSW86]. L'idée consiste à déplier une surface constituée de polygones, en partant d'une graine choisie par l'utilisateur. Une idée similaire a été développée dans [BVI91]. Cette dernière méthode permet le dépliage d'une surface paramétrique en autorisant l'apparition de discontinuités quand la courbure géodésique dépasse un certain seuil. Nous montrerons dans la Section 3.5.5 que l'équivalence entre paramétrisation et dépliage peut être utilisée en géologie pour tester la validité de modèles du sous-sol.

Il est également possible de minimiser les distortions dues au placage de textures en utilisant des techniques d'optimisation. Par exemple, dans la méthode décrite dans [ML88], les auteurs proposent de construire une paramétrisation d'une surface en partant d'une grille de points régulièrement échantillonés sur la surface. La grille est alors optimisée itérativement en minimisant un critère de distortion global. Krishnamurthy *et. al.* [KL96] proposent une approche similaire, pour convertir une surface triangulée en un ensemble de surfaces B-Splines. Dans le cas de surfaces triangulées, un placage de texture peut être considéré comme la donnée des coordonnées (u, v) aux sommets d'une triangulation. Cette approche suggère d'utiliser des résultats de la théorie des graphes [Tut60], par exemple les *cartes harmoniques*[3], comme cela a été proposé dans [EDD+95]. Cette dernière méthode consiste à minimiser un critère appelé *dispersion métrique*, qui permet de préserver les aires. Malheureusement, cette méthode ne permet pas de préserver les angles, et l'approximation introduite dans [EDD+95] entraine parfois la construction d'une fonction non inversible. Les limitations inhérentes à l'utilisation des cartes harmoniques ont été contournées dans [LSS+98] et [DKT98], en utilisant une approche de type subdivision pour lisser la paramétrisation (les méthodes de subdivision ont été évoquées dans la partie 2). Malheureusement, cette dernière méthode permet difficilement de prendre en compte des informations fournies par l'utilisateur. Même s'il est possible de marquer certains côtés de la surface comme frontière entre sous-domaines, assurer

[3]Harmonic Maps

une correspondance fine entre des détails de la surface et des détails de la texture reste très difficile en utilisant ce type d'approches.

Floater a également proposé dans [Flo97] une solution meilleure que les cartes harmoniques, qui se traduit par une autre manière de choisir les coordonnées (u, v) aux sommets d'une triangulation. Cette dernière méthode est une généralisation des *cartes barycentriques*[4], méthode également introduite dans [Tut60]. Les coordonnées (u, v) de texture sont alors définies comme la solution d'un systême linéaire, où chaque point (u, v) est une combinaison convexe de ses voisins. Pour les courbes, une paramétrisation sans distortions correspond à la notion d'*abscisse curviligne*. Floater [Flo97] propose une manière de choisir les coefficients de ces combinaisons convexes qui permet de définir pour les surfaces l'équivalent de l'abscisse curviligne. Cette famille de méthodes de placage de textures par optimisation globale fournit de bons résultats pour un ensemble de surfaces simples, mais des limitations apparaissent rapidement dès lors que l'on veut les appliquer à des surfaces complexes (comme celles rencontrées en géologie numérique ou en graphisme par ordinateur). Par exemple, comme la plupart des surfaces ne sont pas développables, des distortions vont toujours apparaître. Dans ce cas, l'utilisateur aimerait pouvoir spécifier la distribution de ces distortions.

Une toute autre famille de méthodes consiste à définir une paramétrisation comme le résultat d'un ensemble d'opérations combinatoires effectuées sur un pavage de l'espace. Ce point de vue a été exploré par Edelsbrunner *et. al.* dans [EW97], pour construire des *cartogrammes*. Un cartogramme est une carte géographique déformée, où les déformations véhiculent de l'information (par exemple, l'aire des pays peut être déformée en fonction du nombre de leurs habitants). Pour atteindre cet objectif, les auteurs construisent un homéomorphisme de $\mathbb{R}^2$ dans lui même, défini par une suite d'opérations faisant apparaître ou disparaître des triangles afin de déformer la carte.

Dans cette partie, nous proposons une nouvelle méthode d'optimisation globale. Par rapport à d'autres méthodes similaires, elle se fonde sur une approche *modulaire*, ce qui permet d'adapter la paramétrisation sous-jacente aux besoins de l'utilisateur. Par exemple, il est possible de définir l'importance relative accordée à la préservation des distances par rapport à celle des angles. Ces différents *poids* associés aux critères à minimiser peuvent varier sur la surface, ce qui permet à l'utilisateur de choisir les *zones d'intérêt* où il est souhaitable et prioritaire de minimiser les distortions. Ainsi, dans le cas d'un visage, l'utilisateur souhaitera minimiser les distortions au niveau des yeux, du nez et de la bouche, zones "clef" plus riches en détails. De plus, notre méthode permet de rendre la paramétrisation continue à travers les "déchirures" éventuelles de la surface, ce

[4] barycentric mapping

qui permet de texturer en une seule fois une surface complexe, sans nécessiter de découpage comme dans d'autres méthodes [Blo85]. D'autre part, notre méthode est plus automatique que d'autres méthodes interactives [Ped95, PM94, MYV93], qui nécessitent beaucoup d'interventions de la part de l'utilisateur. Lorsque ces méthodes sont utilisées, la paramétrisation doit être partiellement, voire totalement définie par l'utilisateur.

Dans la section 3.2 de ce chapitre, nous introduisons les différentes notions mises en jeu dans le cadre du placage de textures, et nous montrons comment une paramétrisation peut être générée en utilisant un algorithme itératif d'optimisation. Dans la section 3.3, nous traitons le problème d'un placage de texture sans distortion, et les critères correspondants y sont explicités. Ces critères sont exprimés sous forme de contraintes linéaires, et nous montrons comment modifier l'algorithme itératif précédemment introduit afin de les honorer. Ceci permet de définir un algorithme général pouvant être facilement étendu afin de prendre en compte des informations fournies par l'utilisateur, comme nous le montrons dans la section 3.4. Par exemple, le respect de courbes iso-paramétriques et la continuité de la paramétrisation à travers des déchirures éventuelles de la surface y sont abordés. La section 3.5 montre ensuite comment la notion de paramétrisation d'une surface triangulée peut servir de base à de nombreux algorithmes, comme par exemple des méthodes d'optimisation de maillages, ou encore de dépliage de surfaces.

3.2 Paramétrisation d'une surface triangulée

Dans cette section, nous définissons la notion de paramétrisation d'une surface triangulée. Nous présentons une nouvelle méthode pour construire une telle paramétrisation, ainsi qu'un algorithme itératif d'optimisation. La section suivante montre alors comment les distortions peuvent être minimisées.

3.2.1 Paramétrisation $\mathbf{x}(u,v)$ et paramétrisation inverse $\Phi(x,y,z)$

Comme nous le montrons Figure 3.1, étant donnée une surface $\mathbf{S}$ de $I\!R^3$, une *paramétrisation* $\mathbf{x}(\mathbf{u},\mathbf{v})$ de $\mathbf{S}$ est une fonction bijective mettant un sous-ensemble $\mathcal{D}$ de $I\!R^2$ en correspondance avec la surface $\mathbf{S}$.

$$(u,v) \in \mathcal{D} \quad \rightarrow \quad \mathbf{x}(u,v) = \begin{bmatrix} \mathbf{x}^x(u,v) \\ \mathbf{x}^y(u,v) \\ \mathbf{x}^z(u,v) \end{bmatrix}$$

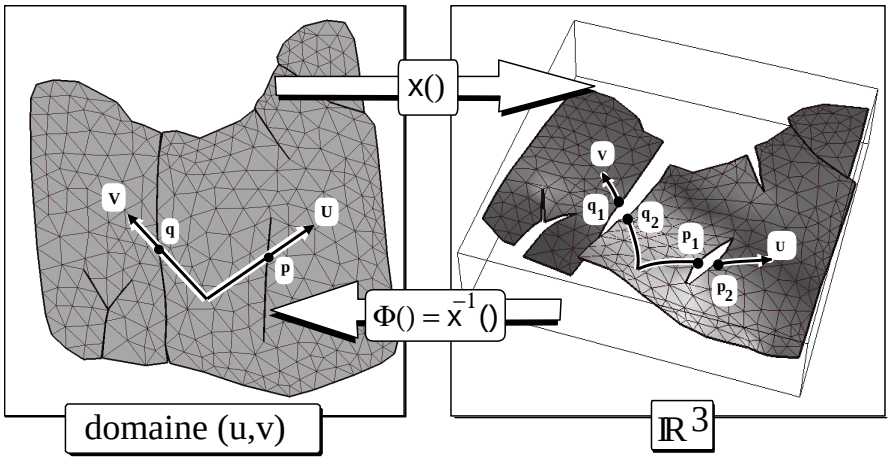


Figure 3.1: *Paramétrisation* $\mathbf{x}(u,v)$ *et paramétrisation inverse* $\Phi(x,y,z)$ *mettant en correspondance une surface* $\mathbf{S}$ *de* $\mathbb{R}^3$ *avec le domaine* $\mathcal{D} \subset \mathbb{R}^2$.

Nous allons nous intéresser plus particulièrement à une classe de paramétrisations $\mathbf{x}(u,v)$ associées à des surfaces présentant des "déchirures" (par exemple une surface géologique faillée). Comme nous le verrons dans la section 3.5, il peut être utile de mettre en correspondance les deux bords de chacune des déchirures avec la même courbe dans le domaine (u,v). Par exemple, sur la Figure 3.1, les points $\mathbf{p}_1$ et $\mathbf{p}_2$ sont mis en correspondance avec le point $\mathbf{p}$. La même chose s'applique à $\mathbf{q}_1$ et $\mathbf{q_2}$ qui sont mis en correspondance avec le point $\mathbf{q}$. Il convient de préciser qu'une telle paramétrisation n'est définie que sur l'intérieur de la surface, sans quoi ce ne serait plus une fonction inversible.

Étant donnée une paramétrisation $\mathbf{x}(u,v)$, les notions suivantes peuvent être définies:

- L'ensemble $\mathcal{D} \subset \mathbb{R}^2$ est appelé le *domaine* (u,v), ou encore *domaine paramétrique* .
- Par définition, $\mathbf{x}(u,v)$ est bijective. Elle a donc une fonction inverse $\Phi = \mathbf{x}^{-1}$, que l'on appelle une *paramétrisation inverse*[5] de la surface $\mathbf{S}$:

$$(x,y,z) \in \mathbf{S} \quad \rightarrow \quad \Phi(x,y,z) = \begin{bmatrix} \Phi^u(x,y,z) \\ \\ \Phi^v(x,y,z) \end{bmatrix} = \begin{bmatrix} u(x,y,z) \\ \\ v(x,y,z) \end{bmatrix}$$

Dans la plupart des cas, les applications utilisant cette notion de paramétrisation correspondent à différents traitements effectués sur une surface **déjà existante** (voir par exemple la section 3.5). Pour cette raison, il est plus facile de définir les différentes notions dont nous avons besoin en partant de la surface,

[5]Dans la littérature anglaise, le terme *mapping* est également utilisé

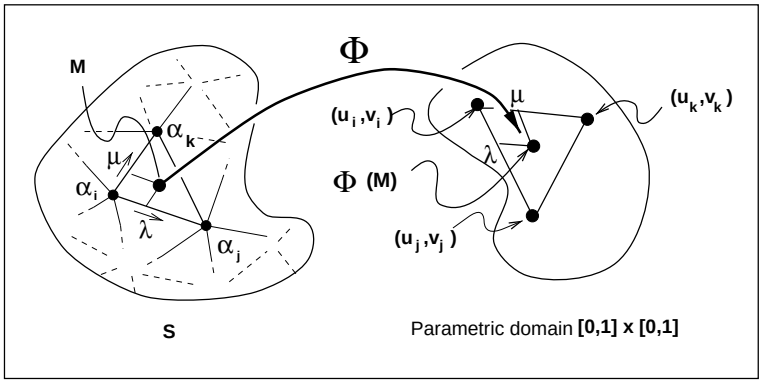


Figure 3.2: *Paramétrisation inverse* Φ *interpolée sur une surface triangulée.*

plutôt que de prendre le domaine (u, v) comme point de départ. Par la suite, nous nous intéresserons donc à la paramétrisation inverse $\Phi(x, y, z)$ plutôt qu'à la paramétrisation $\mathbf{x}(u, v)$. Même si les deux approches sont équivalentes, raisonner en terme de paramétrisation inverse permet des explications plus claires. Par exemple, dans le cadre du placage de textures, le domaine paramétrique $\mathcal{D}$ est *peint* par une image appelée une *texture*. C'est alors la paramétrisation inverse qui nous intéresse, car étant donné un point de la surface $\mathbf{S}$, elle va nous permettre de retrouver le point de la texture qui lui correspond dans $\mathcal{D}$. Si une surface a une paramétrisation $\mathbf{x}$ définie (comme dans le cas des surfaces polynomiales évoquées dans la deuxième partie), l'inverse $\Phi = \mathbf{x}^{-1}$ de cette paramétrisation fournit "naturellement" une fonction pouvant être utilisée pour le placage de textures. Catmull a appliqué cette technique à des splines cubiques dans [Cat74], où il utilise un schéma de subdivision récursive afin d'éviter l'inversion directe de la paramétrisation.

Dans ce qui suit, nous considérons que la surface $\mathbf{S}$ est munie d'une triangulation $\mathcal{G} = \{\Omega, \mathcal{T}\}$, où Ω dénote l'ensemble des sommets de la triangulation, et $\mathcal{T}$ correspond aux triangles de $\mathcal{G}$, définis comme des triplets de sommets. Afin de simplifier les notations, Ω sera identifié avec l'intervalle d'entiers $[1 \dots M]$, où $M = card(\Omega)$ dénote le nombre de sommets de la triangulation. La position géométrique d'un sommet $\alpha \in \Omega$ est notée $\mathbf{p}(\alpha)$.

Pour ce type de surfaces, il semble naturel de définir une paramétrisation inverse Φ à partir de ses valeurs aux sommets Ω de la triangulation $\mathcal{G}$. Cette information peut alors être stockée comme un ensemble de couples (u_i, v_i), où $1 \leq i \leq M$. La manière de choisir ces valeurs (u_i, v_i) sera explicitée par la suite (voir Section 4). Ceci définit une fonction discrète $\varphi : \Omega \rightarrow \mathbf{R}^2$ telle que $\forall \alpha_i \in \Omega, \varphi(\alpha_i) = \{\varphi^u(\alpha_i), \varphi^v(\alpha_i)\} = (u_i, v_i)$. Comme on peut le voir sur la Figure 3.2, une paramétrisation inverse Φ peut alors être définie par morceaux comme l'interpolation linéaire de φ sur chacun des triangles $\mathbf{T} = (\alpha_i, \alpha_j, \alpha_k)$ de $\mathcal{T}$.

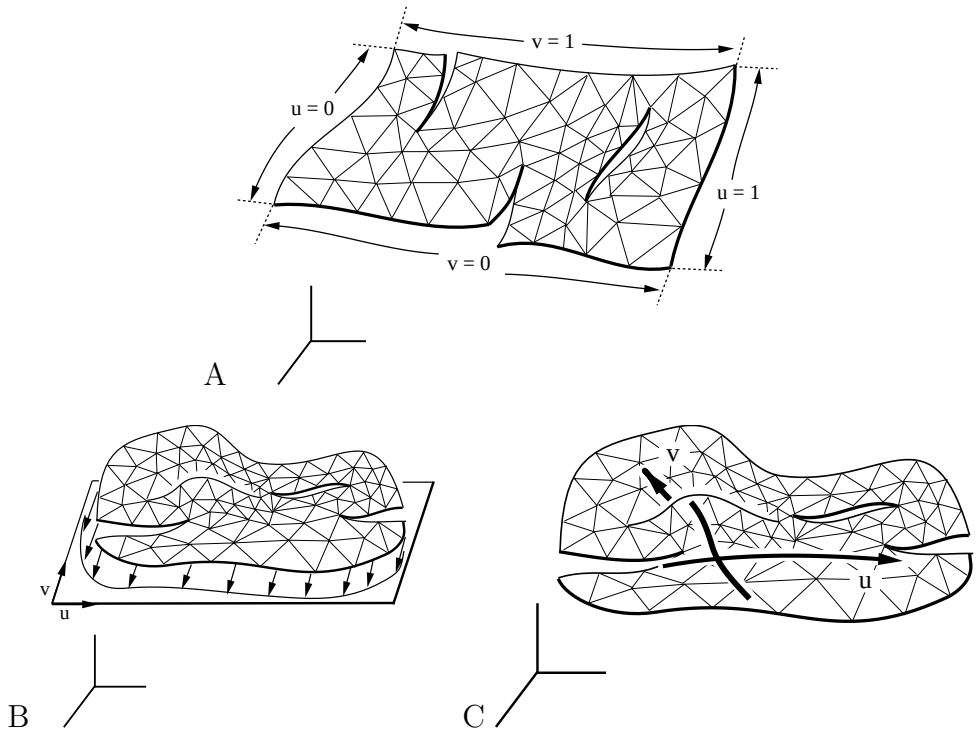


Figure 3.3: *Informations fournies par l'utilisateur pour déterminer une paramétrisation.*

Pour chaque point $\mathbf{p}$ de $\mathbf{T}$, Φ est alors donnée par:

$$\left\{ \begin{array}{rcrcl} \Phi(\mathbf{p}) & = & (1-\lambda-\mu) & . & \varphi(\alpha_i) \\ & + & \lambda & . & \varphi(\alpha_j) \\ & + & \mu & . & \varphi(\alpha_k) \end{array} \right. \tag{3.1}$$

où:

- λ et μ désignent les coordonnées barycentriques locales du point $\mathbf{p}$ par rapport au triangle $\mathbf{T}$.
- $\varphi(\alpha_i) = (u_i, v_i) \quad , \quad \varphi(\alpha_j) = (u_j, v_j) \quad , \quad \varphi(\alpha_k) = (u_k, v_k)$

Pour une surface $\mathbf{S} \in I\!R^3$ donnée, il existe une infinité de paramétrisations possibles pour cette surface. Suivant les objectifs de l'utilisateur, il sera souhaitable que la paramétrisation construite satisfasse certaines propriétés. Par exemple, comme nous le verrons dans la suite, il peut être utile de construire une paramétrisation respectant les distances et les angles. D'autre part, suivant le domaine d'applications, l'utilisateur peut souhaiter fixer les coordonnées (u, v) pour certains ensembles de noeuds, organisés suivant les configurations suivantes:

1. mettre en correspondance quatre bords de la surface avec le carré unité $[0,1]\times[0,1]$ (voir Figure 3.3-A), si par exemple une grille régulière doit être construite à partir de la surface;

2. mettre en correspondance le bord de la surface avec un polygone de $\mathbb{R}^2$. Par exemple, la projection du bord sur son plan de composantes principales peut être utilisée pour définir ce polygone. Les coordonnées locales dans ce plan définissent une base de coordonnées (u,v), utilisée pour les points du bord (voir Figure 3.3-B).

3. définir une base curviligne (Figure 3.3-C), ce qui peut être particulièrement intéressant dans le cadre de problèmes géologiques de dépliage de surfaces (voir Section 3.5).

Pour chacune de ces configurations fournies par l'utilisateur, le problème de la paramétrisation consistera à assigner aux autres sommets de la triangulation des coordonnées (u,v) de manière à interpoler (ou extrapoler dans la configuration 3) celles qui ont été fournies par l'utilisateur. Ce problème est très similaire à celui de l'ajustement aux données, que nous avons évoqué dans le chapitre précédent (Section 2.2.1). Ici, la condition de préservation des distances et des angles joue le rôle de la matrice de régularisation R.

La plupart des techniques de paramétrisation existantes [EDD+95, Flo97], sur lesquelles nous reviendrons plus loin, ne permettent de prendre en compte que les deux premières configurations. De plus, ces dernières métodes requièrent que les coordonnées (u,v) des sommets du bord définissent un polygone convexe dans l'espace paramétrique. De plus, en ce qui concerne la configuration 2, il n'est possible de définir une paramétrisation que si la courbe du bord ne s'auto-intersecte pas quand elle est projetée sur son plan de composantes principales. La méthode que nous proposons, ainsi que celle proposée par Hormann *et. al.* dans [HG98], permettent toutes deux de prendre en compte les trois types de configurations, sans aucune restriction concernant la forme du bord de la surface. De plus, lorsque l'utilisateur utilise la troisième configuration en tant que contraintes, comme la paramétrisation est laissée libre sur le bord, le système a plus de degrés de liberté pour minimiser les distortions. Nous traiterons également de la prise en compte de différents types de contraintes. Ce dernier point permet plus de souplesse et d'interactivité que la méthode proposée dans [HG98].

Les cartes harmoniques, décrites dans ce paragraphe, et la paramétrisation de Floater, introduites dans la section suivante, sont deux méthodes couramment utilisées pour construire une paramétrisation d'une surface triangulée. Nous introduirons par la suite un formalisme général, englobant ces deux méthodes, et permettant également de définir de nouveaux critères. Les cartes harmoniques

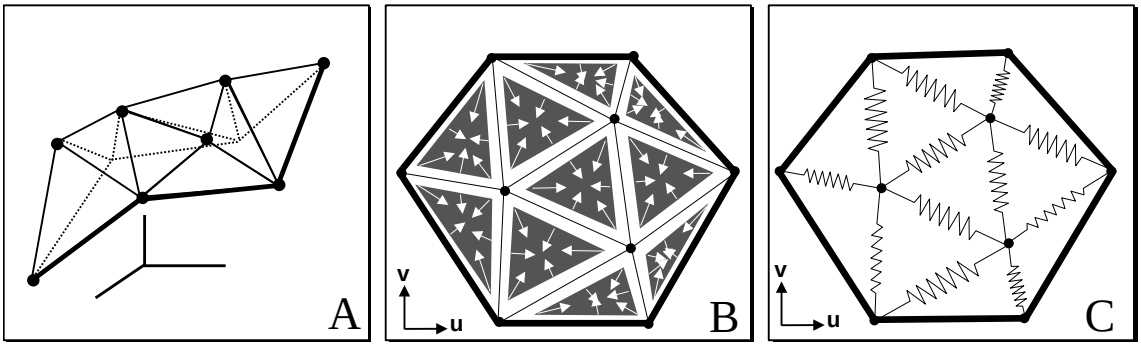


Figure 3.4: *La notion de carte harmonique d'une surface de $\mathbb{R}^3$ (A) est définie par la minimisation d'une énergie de triangles élastiques (B), souvent approximée par des ressorts (C).*

représentent l'une des méthodes les plus connues pour construire une paramétrisation d'une surface triangulée, c'est pourquoi nous présentons ici leur définition, ainsi que les problèmes posés par ce type de méthodes.

Comme nous le montrons Figure 3.4, une carte harmonique d'une surface de $\mathbb{R}^3$ (Figure 3.4-A) est définie en considérant que les triangles de la surface sont faits d'un matériau élastique. Les sommets du bord de la surface sont arbitrairement positionnés dans le domaine (u, v) sur un polygône convexe, et les autres sommets atteignent une position d'équilibre qui minimise l'énergie élastique des triangles (voir Figure 3.4-B). La carte harmonique est alors définie comme étant la configuration d'énergie minimum du système. Le système d'équations correspondant est relativement complexe et difficile à résoudre. Pour cette raison, Eck *et. al.* ont proposé dans [EDD+95] de remplacer chaque paire de triangles élastiques, partageant un côté, par un ressort qui rassemble l'énergie élastique des deux triangles considérés.

Soit $d(\alpha_i, \alpha_j)$ la longueur du côté (α_i, α_j), et $A(\alpha_i, \alpha_j, \alpha_k)$ l'aire du triangle qui connecte α_i, α_j et α_k. L'énergie élastique à minimiser est définie ainsi:

$$\left\{\begin{array}{lcl} E_{harm}(\varphi) & = & \displaystyle\sum_{(\alpha_i,\alpha_j)\in\mathcal{E}} K^2(\alpha_i,\alpha_j).||\varphi(\alpha_i)-\varphi(\alpha_j)||^2 \\ & & \\ \text{avec:} & & \\ & & \\ K^2(\alpha_i,\alpha_j) & = & \frac{1}{2}.\{d^2(\alpha_i,\alpha_{k_1})+d^2(\alpha_j,\alpha_{k_1})-d^2(\alpha_i,\alpha_j)\}/A(\alpha_i,\alpha_j,\alpha_{k_1}) \quad + \\ & & \\ & & \frac{1}{2}.\{d^2(\alpha_i,\alpha_{k_2})+d^2(\alpha_j,\alpha_{k_2})-d^2(\alpha_i,\alpha_j)\}/A(\alpha_i,\alpha_j,\alpha_{k_2}) \end{array}\right. \tag{3.2}$$

où α_{k_1} et α_{k_2} dénotent les deux sommets opposés des deux triangles qui partagent le côté (α_i, α_j), et $\varphi(\alpha)$ dénote les coordonnées (u, v) associées au noeud α.

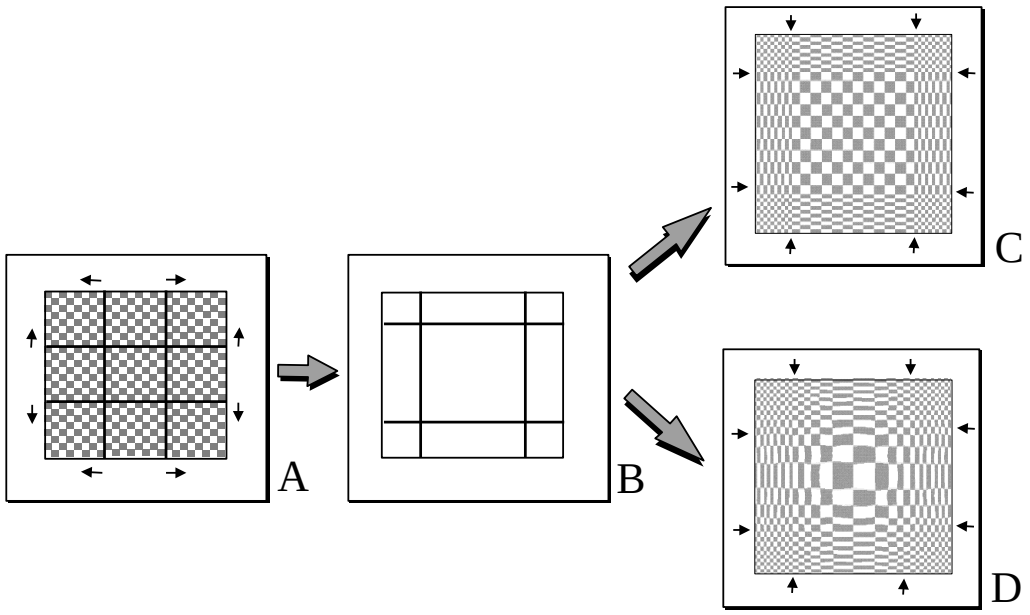


Figure 3.5: *Une surface élastique peinte avec un motif en damier est fixée sur un ensemble de barres rigides (A). Ces barres sont ensuite déplacées suivant la direction des flèches (B). Comparaison entre le résultat obtenu avec les cartes harmoniques (C) et notre critère (D). Ce dernier produit une meilleure continuité autour des contraintes (indiquées par des flèches).*

D'autres choix possibles pour les "raideurs des ressorts" $K^2(\alpha_i, \alpha_j)$ sont évoqués dans [HG98].

Cette fonction objective est parfois également nommée *dispersion métrique*, car elle permet de caractériser la manière dont la paramétrisation altère les aires. Nous verrons plus loin comment caractériser plus finement une paramétrisation, en quantifiant la manière dont les distances et les angles sont modifiés.

3.2.2 Paramétrisation par optimisation globale

Étant donnée une triangulation $\mathcal{G} = \{\Omega, \mathcal{T}\}$, nous voulons donner à chaque sommet $\alpha \in \Omega$ un couple de coordonnées (u, v). Les cartes harmoniques que nous avons vues précédemment permettent d'atteindre cet objectif. Cependant, d'autres méthodes globales, que nous allons voir dans cette section, fournissent un meilleur résultat en ce qui concerne la prise en compte de contraintes. La Figure 3.5 montre comment ces méthodes permettent d'interpoler des sommets dont les coordonnées (u, v) ont été positionnées. D'une manière intuitive, une surface élastique est peinte avec un damier et fixée sur un ensemble de barres rigides (Figure 3.5-A). Nous nous intéressons à présent à la manière dont le damier se déforme si les barres transverses sont déplacées (Figure 3.5-B). Les

cartes harmoniques fournissent comme résultat une interpolation bi-linéaire entre les contraintes (3.5-C), qui fait clairement apparaître celles-ci comme des discontinuités dans la taille des carrés. Nous allons voir à présent comment obtenir le résultat que nous montrons sur la Figure 3.5-D. Sur cette figure, une répartition plus régulière de la taille des carrés sur la surface a été obtenue.

Afin d'obtenir un meilleur résultat que celui fournit par les cartes harmoniques, Floater a étudié d'une manière plus générale le problème de la paramétrisation des surfaces triangulées. Il a montré dans [Flo97] qu'il suffit de vérifier les deux conditions suivantes pour définir une paramétrisation:

1. L'image $\varphi(\partial\mathbf{S})$ du bord $\partial\mathbf{S}$ de la surface $\mathbf{S}$ par φ dans le domaine (u, v) est un polygone convexe.
2. Chaque noeud interne est une combinaison convexe de ses voisins.

Nous devons garder à l'esprit que ces deux conditions sont **suffisantes**, mais non nécessaires, pour définir une paramétrisation. Nous montrerons dans la section 3.4.1 comment la première de ces conditions peut être remplacée par une condition moins restrictive.

D'une manière plus formelle, la seconde condition est donnée par l'Équation 3.3:

$$\forall k \in \Omega, \sum_{\alpha \in N(k)} v^{\alpha}(k).\varphi(\alpha) = 0 \tag{3.3}$$

où:

- L'ensemble $N(k)$ dénote l'ensemble des sommets directement connectés à k, y compris k.
- Les scalaires $v^{\alpha}(k)$ sont des coefficients **donnés** tels que:

$$\begin{cases} \forall \alpha \in N(k) - \{k\} & v^{\alpha}(k) & > & 0 \\ \forall k \in \Omega & v^{k}(k) & = & -\sum\limits_{\substack{\alpha \in N(k) \\ \alpha \neq k}} v^{\alpha}(k) \neq 0 \end{cases} \tag{3.4}$$

Une fois que les sommets du bord de la surface ont été mis en correspondance avec un polygone convexe dans l'espace paramétrique, des couples de coordonnées (u, v) doivent être affectés aux sommets internes. Plutôt que de construire φ en résolvant l'Équation 3.3, comme cela est fait dans les approches classiques [Flo97], la méthode décrite ici consiste à minimiser *au sens des moindres carrés* un critère global, en honorant simultanément un ensemble de contraintes

linéraires, comme nous le montrerons dans la section suivante. L'algorithme est fondé sur l'interpolation lisse discrète D.S.I.[6], décrite par Mallet dans [Mal89, Mal92, Mal99]. Le critère minimisé par la méthode D.S.I. est appelé la *rugosité* R, définie par l'Équation 3.5:

$$R(\varphi) = \sum_{k\in\Omega} \sum_{\nu\in\{u,v\}} \left\{ \sum_{\alpha\in N(k)} v^{\alpha}(k).\varphi^{\nu}(k) \right\}^2 \tag{3.5}$$

Le minimum de cette fonctionnelle est atteint si $\partial R(\varphi)/\partial\varphi^{\nu}(\alpha) = 0$ pour chaque $\alpha \in \Omega$ et pour chacune des deux composantes $\nu \in \{u, v\}$ de φ. Une fois tous les calculs correspondants effectués, cette condition peut se ré-écrire comme ci-dessous:

$$\varphi^{\nu}(\alpha) = -\frac{G^{\nu}(\alpha|\varphi)}{g^{\nu}(\alpha)}$$

avec:

$$\left| \begin{array}{lcl} G^{\nu}(\alpha|\varphi) & = & \displaystyle\sum_{k\in N(\alpha)} \left\{ v^{\alpha}(k). \sum_{\substack{\beta\in N(k)\\ \beta\neq\alpha}} v^{\beta}(k).\varphi^{\nu}(\beta) \right\} \\ g^{\nu}(\alpha) & = & \displaystyle\sum_{\alpha\in N(\alpha)} \{v^{\alpha}(k)\}^2 \end{array} \right. \tag{3.6}$$

L'algorithme suivant calcule itérativement l'ensemble des coefficients (u, v) qui minimise la rugosité de φ donnée dans l'Équation 3.5. Mallet a démontré dans [Mal89] que la rugosité de φ possède un minimum unique, vers lequel converge cet algorithme, à condition qu'au moins un sommet α de la surface ait sa valeur $\varphi(\alpha)$ fixée et que le choix des coefficients $v^{\alpha}(k)$ correspondent à des combinaisons convexes des voisins de k (voir Équation 3.4). Nous montrerons plus loin dans ce document comment cette méthode de paramétrisation peut être améliorée par la définition de contraintes linéaires.

soit I l'ensemble des sommets où φ est inconnue
soit $\varphi_{[0]}$ une solution initiale approximée
tant que la convergence n'a pas été atteinte **{**
  pour $\alpha \in I$ **{**
    pour $\nu \in \{u, v\}$ **{**
      $\varphi^{\nu}(\alpha) := -\frac{G^{\nu}(\alpha)}{g^{\nu}(\alpha)}$
    }

[6]Discrete Smooth Interpolation

```
    }
}
```

Nous avons déjà rencontré dans la deuxième partie les coefficients de pondération $v^\alpha(k)$, qui définissent une *paramétrisation locale* autour du sommet k. Comme nous l'avons déjà mentionné, pour ces paramétrisations locales, plusieurs choix sont possibles. L'un de ces choix décrit dans [Flo97], appelé la pondération *préservatrice des formes*[7], assure que dans l'espace paramétrique, la position d'un sommet relativement à ses voisins "ressemble" à la géométrie locale autour du sommet considéré. Notre approche se démarque de ce type de méthodes, car nous avons choisi de séparer les critères de minimisation des distortions de celui qui assure qu'une paramétrisation valide est construite. Comme nous le verrons plus loin, ceci permet d'obtenir un contrôle plus fin de la façon dont la surface est paramétrisée. Pour cette raison, nous utiliserons la forme la plus simple de combinaison barycentrique pour le choix des coefficients $v^\alpha(k)$, appelée *pondération harmonique* et définie comme suit:

$$v^\alpha(k) = \begin{cases} 1 & \text{si} \quad \alpha \in N(k) - \{k\} \\ -degre(k) & \text{si} \quad \alpha = k \end{cases}$$

où $degre(k)$ dénote le nombre de voisins de k. Il est certes possible d'utiliser l'une des pondérations plus sophistiquées mentionnées auparavant (par exemple la pondération gaussienne, ou encore la pondération préservatrice de formes) puisque toutes vérifient l'Équation 3.4. Cependant, nous montrerons dans la prochaine section que le même effet de minimisation des distortions peut être obtenu à l'aide de contraintes linéaires, tout en offrant une plus grande flexibilité.

3.3 Paramétrisation sans distortion

Dans cette section, nous allons décrire l'ensemble de critères à minimiser afin de définir un placage de textures sans distortions. En quelques mots, ces critères assurent que les distances et les angles sont préservés au travers de la paramétrisation (voir Figure 3.6). Sur une surface courbe, la notion de distance entre deux points est définie comme la longueur d'une géodésique passant par ces deux points, à savoir la longueur de la courbe la plus courte tracée sur la surface et passant par les deux points concernés (voir par exemple [Pet85a]). Dans notre cas, nous laissons ces géodésiques traverser les déchirures de la surface. Ces conditions de non-distortion peuvent être définies de manière équivalente par la perpendicularité et par un espacement homogène des courbes iso-paramétriques tracées sur la surface.

[7] shape preserving weighting

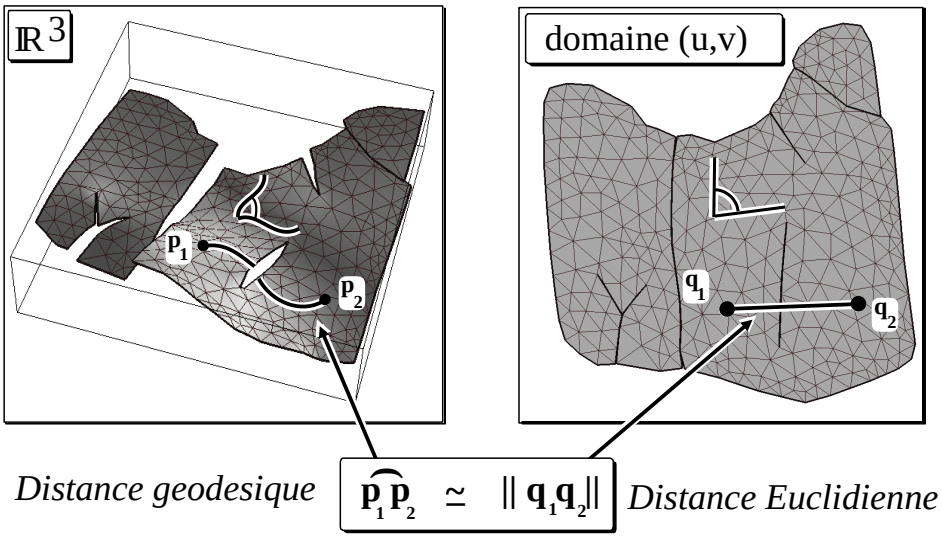


Figure 3.6: *Une paramétrisation sans distortion doit préserver les distances et les angles.*

Autrement dit, les gradients des composantes en u et en v de la paramétrisation inverse Φ doivent être perpendiculaires et constants sur toute la surface (voir [Car76]).

La paramétrisation de Floater, introduite dans la section précédente, fournit une solution correcte au problème de paramétrisation des surfaces triangulées, tout en présentant des propriétés de non-distortion utiles. Toutefois, dans certaines circonstances, il serait souhaitable d'avoir un contrôle plus fin sur la paramétrisation. Par exemple, suivant les applications, l'utilisateur souhaitera accorder plus d'importance à la préservation des distances, ou encore à la préservation des angles à travers la paramétrisation. De plus, la nécessité de mettre le bord de la surface en correspondance avec un polygone convexe peut apparaître comme une limitation des méthodes classiques, car les surfaces pour lesquelles nous souhaitons définir une paramétrisation peuvent avoir des bords de forme arbitraire.

La caractérisation mathématique d'une transformation $\mathbf{x}(u, v)$ qui altère les distances et les angles implique la définition de son *tenseur métrique fondamental* $\mathbf{G}$. Le tenseur métrique fondamental d'une fonction $\mathbf{x}(\mathbf{u}, \mathbf{v})$ est une matrice 2×2, définie en chaque point (u_0, v_0) du domaine paramétrique $\mathcal{D}$ (voir [Gre96, Car76, Boe88, MP77, Sto69]):

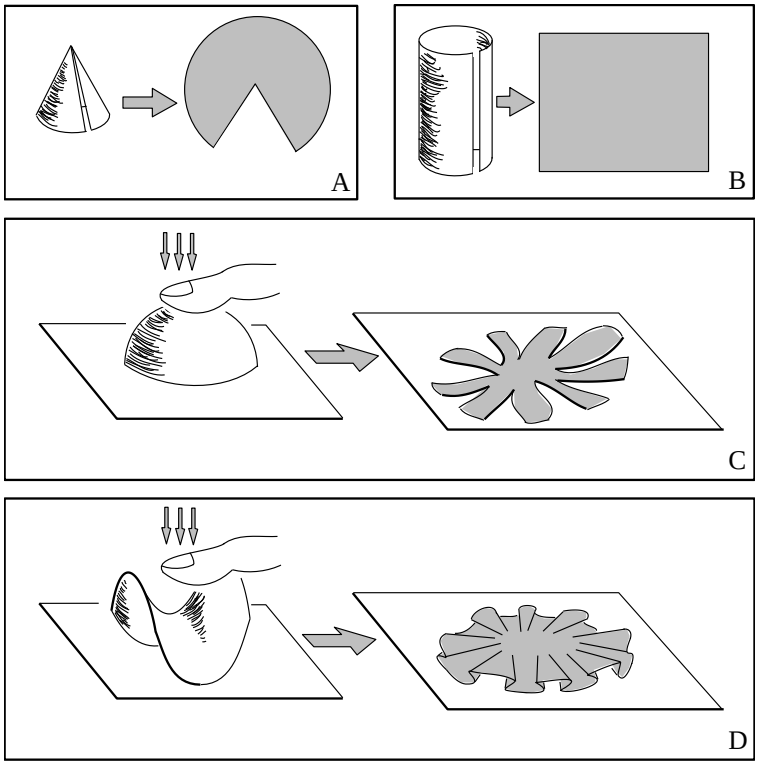


Figure 3.7: *Un cylindre et un cône peuvent être mis à plat sans déformations (voir A et B respectivement). Ce sont des surfaces développables. Ce n'est pas le cas de la plupart des surfaces, telles qu'une sphère (C) ou une selle de cheval (D) pour lesquelles toute tentative de mise à plat causera des déchirures (C) ou des recouvrements (D).*

$$\forall (u_0, v_0) \in \mathcal{D}, \quad \mathbf{G}(u_0, v_0) \quad = \quad \begin{bmatrix} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right)^2 (u_0, v_0) & \frac{\partial \mathbf{x}}{\partial \mathbf{v}} . \frac{\partial \mathbf{x}}{\partial \mathbf{u}} (u_0, v_0) \\ \\ \frac{\partial \mathbf{x}}{\partial \mathbf{u}} . \frac{\partial \mathbf{x}}{\partial \mathbf{v}} (u_0, v_0) & \left(\frac{\partial \mathbf{x}}{\partial \mathbf{v}}\right)^2 (u_0, v_0) \end{bmatrix} \tag{3.7}$$

Si cette matrice $\mathbf{G}$, qui caractérise la paramétrisation $\mathbf{x}(u, v)$, est égale à la matrice unité sur tout le domaine paramétrique, alors $\mathbf{x}(u, v)$ est une paramétrisation *sans distortion* (voir Équation 3.8).

$$\forall (u_0, v_0) \in \mathcal{D}, \quad G(u_0, v_0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

Seules les surfaces d'une classe particulière - dites *développables* - peuvent être munies d'une telle paramétrisation sans distortion. Les surfaces développables représentent un ensemble très limité, et peuvent être définies intuitivement comme l'ensemble des surfaces que l'on peut obtenir en déformant un matériau non-élastique (comme une feuille de papier par exemple). Ainsi, un cylindre, un cône,

ou une plaque de tôle ondulée sont des surfaces développables, alors que ce n'est pas le cas d'une sphère (voir Figure 3.7 et [Pet85c]). Plus précisément, une surface est dite développable si sa courbure totale est nulle en tout point. Comme la plupart des surfaces que nous allons rencontrer ne seront pas développables, nous allons minimiser le critère de non-distortion au sens des moindres carrés. De plus, l'Équation 3.8 peut être considérée comme deux conditions différentes, obtenues en séparant les zéros et les uns de la matrice unité 2×2, ce qui engendre les Équations 3.9 et 3.10 respectivement.

$$\forall (u_0, v_0) \in \mathcal{D}, \quad \frac{\partial \mathbf{x}}{\partial \mathbf{u}} . \frac{\partial \mathbf{x}}{\partial \mathbf{v}} (u_0, v_0) = 0 \tag{3.9}$$

$$\forall (u_0, v_0) \in \mathcal{D}, \quad \begin{cases} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}} \right)^2 (u_0, v_0) &= 1 \\ \\ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right)^2 (u_0, v_0) &= 1 \end{cases} \tag{3.10}$$

Ces deux conditions permettent de définir ce que nous appellerons les *contraintes isométriques*, permettant de préserver la métrique à travers la paramétrisation, à savoir les distances et les angles. Plus précisément, l'Équation 3.9 correspond à la préservation des angles, et assure que les familles de courbes iso-u et iso-v seront perpendiculaires les unes aux autres (voir Figure 3.9 plus loin). L'Équation 3.10 caractérise la préservation des distances, et impose un espacement constant des courbes iso-paramétriques sur la surface (voir Figure 3.10 plus loin). Comme nous pouvons le voir, ces contraintes de non-distortions mettent en jeu les dérivées de la paramétrisation, et d'une manière plus génerale, elles concernent le *gradient* d'une fonction interpolée sur une surface triangulée. Nous montrerons alors comment définir cette notion de gradient dans le cas d'une surface triangulée, et comment l'algorithme présenté dans la section précédente peut être modifié pour prendre en compte ces critères.

3.3.1 Gradient d'une fonction discrète φ interpolée sur une triangulation $\mathcal{G}$

Les conditions de respect des angles et de respect des distances mettent en jeu le tenseur géométrique fondamental, déterminé à partir des dérivées de la paramétrisation $\mathbf{x}$ par rapport à u et v. Comme nous l'avons mentionné précédemment, il est plus facile de caractériser la paramétrisation inverse Φ que la paramétrisation φ, même si les deux approches sont équivalentes. Ceci permet de déduire les relations que la discrétisation φ de Φ sur la surface triangulée doit vérifier.

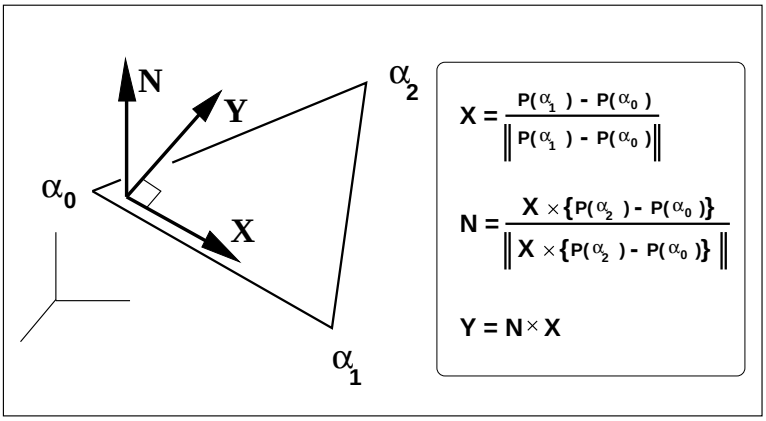


Figure 3.8: *Base orthonormée locale (***X***,* **Y***) d'un triangle* $\mathbf{T} = (\alpha_0, \alpha_1, \alpha_2)$.

les conditions de non-distortion peuvent alors se traduire localement en terme de gradient des composantes en u et v de la paramétrisation inverse Φ.

Comme nous pouvons le voir sur la Figure 3.8, chaque triangle $\mathbf{T} = (\alpha_0, \alpha_1, \alpha_2)$ de $\mathcal{T}$ peut être muni d'une base orthonormée locale $(\mathbf{p}(\alpha_0), \mathbf{X}, \mathbf{Y})$. La fonction $\varphi_T^\nu(X, Y)$ dénote l'interpolation linéaire de φ^ν sur le triangle $\mathbf{T}$, où l'indice $\nu \in \{u, v\}$ représente l'une des deux composantes de φ, et où (X, Y) sont les coordonnées locales exprimées dans la base orthonormée $(\mathbf{p}(\alpha_0), \mathbf{X}, \mathbf{Y})$ de $\mathbf{T}$.

Dans cette base, il est facile de vérifier que le gradient de φ_T^ν est constant sur chaque triangle $\mathbf{T}$, et peut s'exprimer comme une combinaison linéaire des valeurs de φ_T^ν aux trois sommets du triangle $\mathbf{T}$. Les six coefficients $D_X(\alpha_j)$ et $D_Y(\alpha_j)$ correspondants, donnés dans l'Équation 3.11 ci-dessous, ne dépendent que de la géométrie du triangle $\mathbf{T}$.

$$\left| \begin{array}{rcl} \frac{\partial \varphi_T^\nu}{\partial X} & = & \sum_{j=0}^{2} D_X(\alpha_j).\varphi^\nu(\alpha_j) \\ \\ \frac{\partial \varphi_T^\nu}{\partial Y} & = & \sum_{j=0}^{2} D_Y(\alpha_j).\varphi^\nu(\alpha_j) \end{array} \right. \tag{3.11}$$

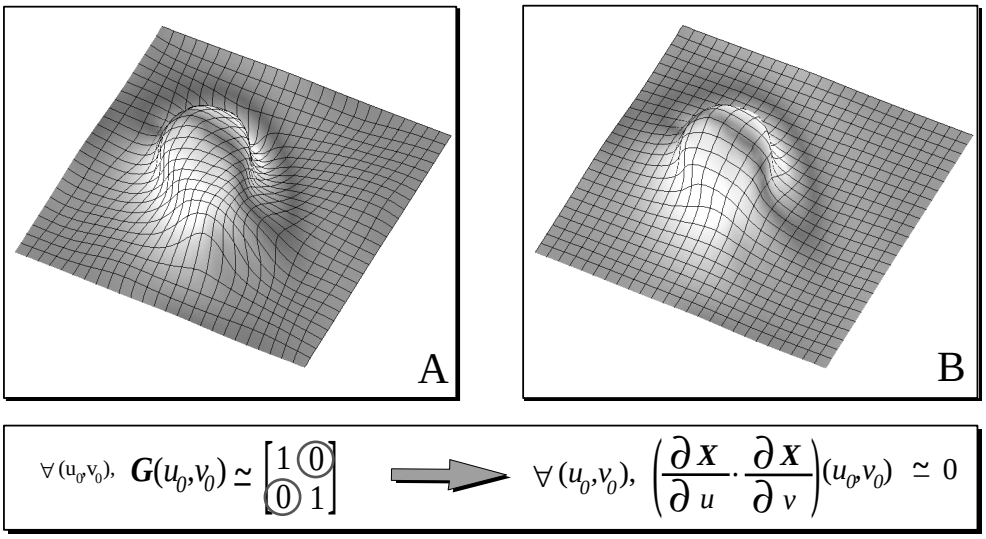


Figure 3.9: *Effet de la contrainte de respect des angles. Les deux familles de courbes iso-u et iso-v sont rendues perpendiculaires l'une par rapport à l'autre.*

$$\text{avec:} \begin{cases} \begin{array}{rcl} D_X(\alpha_0) & = & (y_1 - y_2)/d \\ D_X(\alpha_1) & = & (y_2 - y_0)/d \\ D_X(\alpha_2) & = & (y_0 - y_1)/d \\ D_Y(\alpha_0) & = & (x_2 - x_1)/d \\ D_Y(\alpha_1) & = & (x_0 - x_2)/d \\ D_Y(\alpha_2) & = & (x_1 - x_0)/d \end{array} \\ \\ d = (x_1 - x_0).(y_2 - y_0) - (x_2 - x_0).(y_1 - y_0) \\ \\ \left| \begin{array}{rcl} x_j & = & (\mathbf{p}(\alpha_j) - \mathbf{p}(\alpha_0)).\mathbf{X} \\ y_j & = & (\mathbf{p}(\alpha_j) - \mathbf{p}(\alpha_0)).\mathbf{Y} \end{array} \right. \quad \forall j \in \{0,1,2\} \end{cases}$$

Il est alors possible, en utilisant cette définition du gradient de φ, de ré-écrire les équations correspondant au respect des angles et au respect des distances. Ainsi, la condition de respect des angles, ou encore d'orthogonalité des courbes iso-u et iso-v dans un triangle donné (Équation 3.9) s'écrit :

$$\begin{bmatrix} \frac{\partial \varphi_T^u}{\partial X} & \frac{\partial \varphi_T^u}{\partial Y} \end{bmatrix} . \begin{bmatrix} \frac{\partial \varphi_T^v}{\partial X} \\ \\ \frac{\partial \varphi_T^v}{\partial Y} \end{bmatrix} = 0 \tag{3.12}$$

Supposons que la composante φ^u soit fixée et que l'on souhaite déterminer φ^v, en rendant les courbes iso-v perpendiculaires aux iso-u. Si l'on remplace dans l'équation de perpendicularité (Équation 3.12) le gradient de φ^v par son

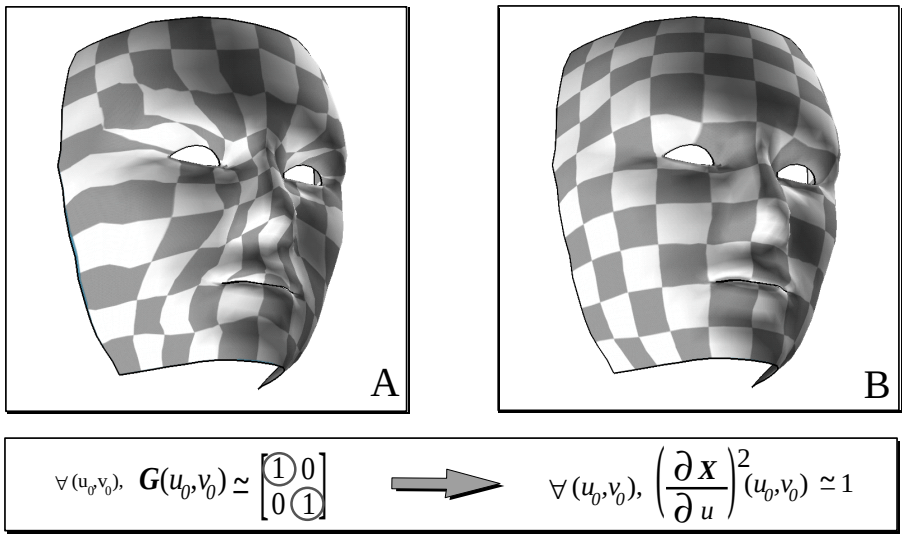


Figure 3.10: *Effet de la contrainte de respect des distances. La taille des carrés est rendue homogène sur l'ensemble de la surface.*

expression (Équation 3.11), alors on obtient la relation suivante (Équation 3.13), qui combine **linéairement** les valeurs de φ^v aux trois sommets $(\alpha_0, \alpha_1, \alpha_2)$ de $\mathbf{T}$. Et réciproquement, la condition à vérifier quand la composante φ^u est calculée peut être obtenue en échangeant u et v dans l'Équation 3.13 ci-dessous.

$$\sum_{j\in\{0,1,2\}} \left\{ \varphi^v(\alpha_j).\left(\frac{\partial \varphi^u_T}{\partial X}.D_X(\alpha_j) + \frac{\partial \varphi^u_T}{\partial Y}.D_Y(\alpha_j) \right) \right\} \quad = \quad 0 \tag{3.13}$$

L'autre condition caractérisant une paramétrisation sans distortion concerne le respect des distances, qui peut s'écrire ainsi :

$$\left[\frac{\partial \varphi^u_T}{\partial X} \quad \frac{\partial \varphi^u_T}{\partial Y} \right]^2 \quad = \quad 1 \tag{3.14}$$

Comme nous allons le voir par la suite, les conditions liant les valeurs de φ par des relations **linéaires** sont plus faciles à respecter, ce qui n'est pas le cas de cette dernière condition de respect des distances. Pour cette raison, nous allons considérer une condition légèrement plus forte, qui impose que le gradient de chacune des composantes de φ ne doit pas trop varier d'un triangle à l'autre (leur module va donc être également constant). Afin d'exprimer cette condition, nous définissons pour chaque couple $(\mathbf{T}, \tilde{\mathbf{T}})$ de triangles adjacents une base commune (voir Figure 3.8). Le vecteur $\mathbf{X}$ est partagé par les deux bases, et $\tilde{\mathbf{Y}}$ est tel que $\mathbf{Y}$ et $\tilde{\mathbf{Y}}$ deviendraient colinéaires si la paire de triangles $(\mathbf{T}, \tilde{\mathbf{T}})$ était dépliée le long de leur côté commun $[\alpha_0, \alpha_1]$. Une autre manière de considérer cette condition est de dire qu'elle impose un espacement constant des courbes iso-paramétriques tracées sur la surface.

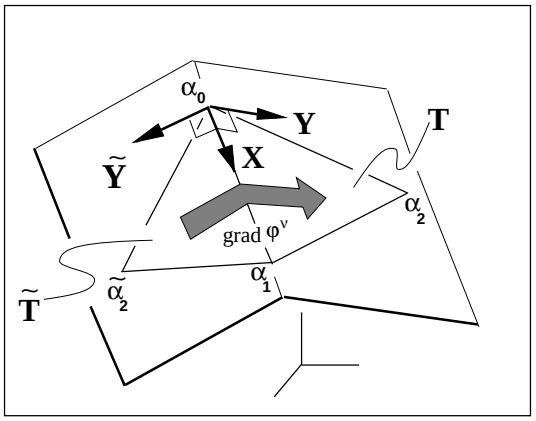


Figure 3.11: *Gradient constant à travers le côté commun de deux triangles* $\mathbf{T}$ *et* $\tilde{\mathbf{T}}$.

L'espacement constant des iso-paramétriques est vérifié, si et seulement si, pour chaque côté de $\mathcal{T}$, la relation suivante est vérifée :

$$\begin{cases} \dfrac{\partial \varphi^u_T}{\partial X} = \dfrac{\partial \varphi^u_{\tilde{T}}}{\partial X} \quad ; \quad \dfrac{\partial \varphi^u_T}{\partial Y} = -\dfrac{\partial \varphi^u_{\tilde{T}}}{\partial \tilde{Y}} \\ \dfrac{\partial \varphi^v_T}{\partial X} = \dfrac{\partial \varphi^v_{\tilde{T}}}{\partial X} \quad ; \quad \dfrac{\partial \varphi^v_T}{\partial Y} = -\dfrac{\partial \varphi^v_{\tilde{T}}}{\partial \tilde{Y}} \end{cases} \tag{3.15}$$

En remplaçant dans cette équation les gradients de φ^u et φ^v par leurs expressions dans $\mathbf{T}$ et $\tilde{\mathbf{T}}$, on obtient les quatre relations **linéaires** suivantes (voir Équation 3.16), caractérisant les deux composantes en X et en Y des gradients de φ^u et φ^v. Le terme δ_W prend en compte le fait que $\mathbf{Y}$ et $\tilde{\mathbf{Y}}$ pointent dans des directions opposées.

$$\begin{array}{l|l} & \varphi^\nu(\alpha_0).\left\{D_W(\alpha_0) + \delta_W.\tilde{D_W}(\alpha_0)\right\} + \\ \forall \nu \in \{u,v\}, & \varphi^\nu(\alpha_1).\left\{D_W(\alpha_1) + \delta_W.\tilde{D_W}(\alpha_1)\right\} + \\ \forall W \in \{X,Y\}, & \varphi^\nu(\alpha_2).\{D_W(\alpha_2)\} + \\ & \varphi^\nu(\tilde{\alpha_2}).\left\{\delta_W.\tilde{D_W}(\tilde{\alpha_2})\right\} = 0 \end{array} \tag{3.16}$$

$$\text{où} \quad \delta_W = \begin{cases} -1 & \text{si} \quad W = X \\ +1 & \text{si} \quad W = Y \end{cases}$$

3.3.2 Prise en compte de contraintes linéaires

Nous avons montré Section 2 comment D.S.I. peut être utilisé afin de construire une paramétrisation d'une surface triangulée. Maintenant, nous souhaitons

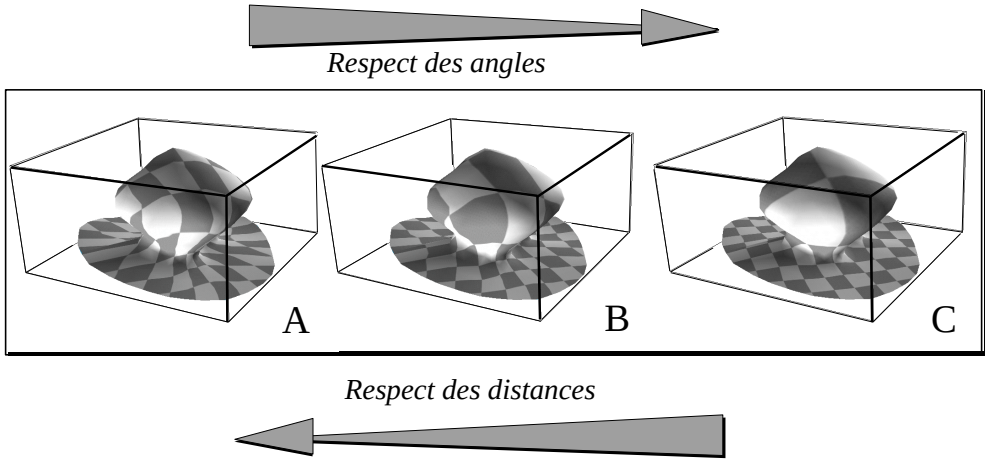


Figure 3.12: *Pour une surface non-développable telle un dôme de sel, l'utilisateur doit faire un choix entre le respect des distances (A) et le respect des angles (C). Un compromis a été atteint (B) en donnant la même importance aux deux contraintes. Il est possible de passer continument de (A) à (C), en faisant varier l'importance relative des deux contraintes.*

prendre en compte les deux critères de non-distortion introduits dans la section précédente, à savoir le critère de préservation des angles et des distances. Comme nous l'avons montré, ces deux critères peuvent s'écrire sous forme d'un ensemble de relations linéaires. Dans le cas général, la surface que nous considèrons n'est pas développable, il n'est alors pas possible d'honorer strictement ces contraintes. Pour cette raison, nous allons les respecter au sens des moindres carrés, ce qui permettra de *minimiser* les distortions. Nous donnons la forme générale d'une telle contrainte dans l'Équation 3.17 ci-dessous :

$$\sum_{\alpha \in \Omega} \{A_{c^\nu}(\alpha).\varphi^\nu(\alpha)\} \quad = \quad b_{c^\nu} \tag{3.17}$$

où les valeurs $A_{c^\nu}(\alpha)$ et le scalaire b_{c^ν} sont des coefficients **fixés** qui définissent la contrainte c.

Si nous considérons l'Équation 3.13, correspondant à la perpendicularité des courbes iso-paramétriques dans le triangle $\mathbf{T} = (\alpha_0, \alpha_1, \alpha_2)$, nous obtenons deux contraintes $c^u_{\mathbf{T}}$ et $c^v_{\mathbf{T}}$ à honorer quand nous interpolons φ^u et φ^v respectivement. L'Équation 3.18 ci-dessous donne l'expression de la contrainte $c^v_{\mathbf{T}}$. La contrainte jumelle $c^u_{\mathbf{T}}$ peut être obtenue en permutant les indices u et v dans cette équation.

$$\left|\begin{array}{lcl} \forall j \in \{0,1,2\}, & & \\ & A_{c^v_{\mathbf{T}}}(\alpha_j) = \frac{\partial \varphi^u_{\mathcal{T}}}{\partial X}.D_X(\alpha_j) + \frac{\partial \varphi^u_{\mathcal{T}}}{\partial Y}.D_Y(\alpha_j) & \\ \forall \alpha \notin \{\alpha_0,\alpha_1,\alpha_2\}, & & \\ & A_{c^v_{\mathbf{T}}}(\alpha) = 0 & \\ & b_{c^v_{\mathbf{T}}} = 0 & \end{array}\right. \tag{3.18}$$

Le critère d'homogénéité exprimé dans l'Équation 3.16 peut être écrit sous la forme de quatre contraintes $c^{uX}_{\mathbf{E}}$, $c^{uY}_{\mathbf{E}}$, $c^{vX}_{\mathbf{E}}$ et $c^{vY}_{\mathbf{E}}$, définies par l'Équation 3.19 plus loin. Ces contraintes doivent être prises en compte au niveau de chaque côté $\mathbf{E} = (\alpha_0, \alpha_1)$ de la triangulation $\mathcal{G}$ (sauf sur le bord). Les sommets α_2 et $\tilde{\alpha_2}$ correspondent aux sommets des deux triangles $\mathbf{T}$ et $\tilde{\mathbf{T}}$ qui ne sont pas une extrémité de leur côté commun $\mathbf{E}$.

$$\left|\begin{array}{lcl} A_{c^{\nu W}_{\mathbf{E}}}(\alpha_0) & = & \left\{D_W(\alpha_0) + \delta_W.\tilde{D_W}(\alpha_0)\right\} \\ A_{c^{\nu W}_{\mathbf{E}}}(\alpha_1) & = & \left\{D_W(\alpha_1) + \delta_W.\tilde{D_W}(\alpha_1)\right\} \\ A_{c^{\nu W}_{\mathbf{E}}}(\alpha_2) & = & D_W(\alpha_2) \\ A_{c^{\nu W}_{\mathbf{E}}}(\tilde{\alpha_2}) & = & \delta_W.\tilde{D_W}(\tilde{\alpha_2}) \\ A_{c^{\nu W}_{\mathbf{E}}}(\alpha) & = & 0 \quad \forall \alpha \notin \{\alpha_0,\alpha_1,\alpha_2,\tilde{\alpha_2}\} \\ b_{c^{\nu W}_{\mathbf{E}}} & = & 0 \end{array}\right. \tag{3.19}$$

avec:

$$\nu \in \{u,v\} \quad ; \quad W \in \{X,Y\} \quad ; \quad \delta_W = \left\{\begin{array}{ll} -1 & \text{if} \quad W = X \\ +1 & \text{if} \quad W = Y \end{array}\right.$$

Comme décrit dans [Mal89, Mal92, Mal99], le critère de *rugosité* minimisé par D.S.I. peut être généralisé pour prendre en compte un ensemble $\mathcal{C}$ de contraintes linéaires au sens des moindres carrés. Dans notre cas, cet ensemble de contraintes $\mathcal{C}$ comprend les contraintes de respect des angles, définies pour chaque triangle ainsi que les contraintes de respect des distances définies pour chaque côté de la triangulation $\mathcal{G}(\Omega, \mathcal{T})$ (voir Équation 3.20, où $\mathcal{E}$ dénote l'ensemble des côtés de la triangulation).

$$\mathcal{C} = \left(\bigcup_{\mathbf{T}\in\mathcal{T}} \{c^u_{\mathbf{T}}, c^v_{\mathbf{T}}\} \right) \cup \left(\bigcup_{\mathbf{E}\in\mathcal{E}} \{c^{u_X}_{\mathbf{E}}, c^{u_Y}_{\mathbf{E}}, c^{v_X}_{\mathbf{E}}, c^{v_Y}_{\mathbf{E}}\} \right) \tag{3.20}$$

La *rugosité généralisée* $R^*(\varphi)$, prenant en compte le degré de violation des contraintes $\mathcal{C}$, est donnée par l'Équation 3.21 ci-dessous. Cette expression comprend les termes déjà définis lors de l'introduction de la rugosité (voir l'Équation 3.5 dans la Section 3), ainsi que des termes additionnels, explicités plus loin, correspondant à la prise en compte des contraintes linéaires.

$$\begin{aligned} R^*(\varphi) = R(\varphi) \quad & + \\ \phi. \sum_{c\in\mathcal{C}} \varpi_c. & \left\{ \left(\sum_{\nu\in\{u,v\}} \sum_{\alpha\in\Omega} A^\nu_c(\alpha).\varphi^\nu(\alpha) \right) - b_c \right\}^2 \end{aligned} \tag{3.21}$$

Dans l'Équation 3.21, le terme $R(\varphi)$ correspond à la rugosité (voir l'Équation 3.5), et le second terme représente le degré de violation des contraintes linéaires. Chacune de ces contraintes c est pondérée par un coefficient donné $\varpi_c > 0$, permettant d'ajuster l'importance relative d'une contrainte par rapport aux autres. Par exemple, il est possible de faire en sorte qu'une paramétrisation respecte *de préférence* les angles plutôt que les distances. La figure 3.12 montre l'influence de ces coefficients ϖ_c affectés aux contraintes. Ainsi, l'utilisateur peut régler de manière très fine l'aspect global de la paramétrisation.

De plus, comme chaque triangle $\mathbf{T}$ et chaque côté $\mathbf{E}$ a une contrainte individuelle, ainsi qu'un coefficient ϖ_c associé, l'utilisateur peut sélectionner les zones de la surface où les distortions doivent être minimisées par ordre de préférence. Le coefficient $\phi \in]0, +\infty[$ est un paramètre donné, appelé le *facteur de respect des contraintes*[8], qui permet de régler l'importance accordée au respect de l'ensemble des contraintes par rapport à la minimisation de la rugosité.

La fonctionelle $R^*(\varphi)$ est une forme quadratique, dont le minimum est atteint si $\partial R^*(\varphi)/\partial\varphi^\nu(\alpha) = 0$ pour $\nu \in \{u, v\}$ et pour tout $\alpha \in \Omega$. L'Équation suivante, obtenue après avoir réordonné les termes, a donc sa solution qui minimise le critère $R^*(\varphi)$:

$$\varphi^\nu(\alpha) = -\frac{G^\nu(\alpha|\phi) + (\phi.\varpi).\Gamma^\nu(\alpha|\varphi)}{g^\nu(\alpha) + (\phi.\varpi).\gamma^\nu(\alpha)} \tag{3.22}$$

$$\left| \begin{array}{lcl} \Gamma^\nu(\alpha|\varphi) & = & \sum_{c\in\mathcal{C}} \varpi_c.\Gamma^\nu_c(\alpha|\varphi) \\ \\ \gamma^\nu(\alpha) & = & \sum_{c\in\mathcal{C}} \varpi_c.\gamma^\nu_c(\alpha) \end{array} \right. \tag{3.23}$$

[8] fitting factor

avec:

$$\begin{cases} \Gamma_c^\nu(\alpha|\varphi) &= A_c^\nu(\alpha).\left\{\sum_{\beta\neq\alpha} A_c^\nu(\beta).\varphi^\nu(\beta) - b_c\right\} \\ \gamma_c^\nu(\alpha) &= A_c^\nu(\alpha)^2 \end{cases} \tag{3.24}$$

La contrainte de respect des angles telle qu'elle est définie ici nécessite une modification de l'algorithme D.S.I. itératif. Les deux boucles internes ont été interverties. À chaque itération, la composante φ^u est interpolée tandis que φ^v est considérée comme étant constante, puis les rôles de φ^u et φ^v sont intervertis. L'algorithme défini ci-dessous calcule itérativement les coordonnées (u, v) des sommets de la triangulation tout en respectant l'ensemble de contraintes spécifiées.

soit I l'ensemble des sommets où φ est inconnue
soit $\varphi_{[0]}$ une solution initiale approximée
tant que la convergence n'a pas été atteinte **{**
  pour $\nu \in \{u, v\}$ **{**
    pour $\alpha \in I$ **{**
      $\varphi^\nu(\alpha) := -\frac{G^\nu(\alpha)+\Gamma^\nu(\alpha|\phi)}{g^\nu(\alpha)+\gamma^\nu(\alpha)}$
    }
  }
}

3.3.3 Relations avec les méthodes existantes

De manière évidente, notre méthode permet de construire une paramétrisation de Floater, si nous choisissons comme coefficients $v^\alpha(k)$ ceux qui définissent la paramétrisation préservatrice des formes (voir [Flo97]). La méthode de Floater devient alors un cas particulier de notre formalisme. En ce qui concerne les cartes harmoniques, que nous avons évoquées dans la première section, elles peuvent également être facilement intégrées dans le formalisme de notre méthode. Si nous considérons les équations qui définissent le critère à minimiser, rappelées ci-dessous, nous pouvons constater que la relation mise en jeu est une somme de carrés de combinaisons linéaires des valeurs de φ aux sommets de la triangulation.

$$\left\{\begin{array}{lcl} E_{harm}(\varphi) & = & \sum\limits_{(\alpha_i,\alpha_j)\in\mathcal{E}} K^2(\alpha_i,\alpha_j).||\varphi(\alpha_i)-\varphi(\alpha_j)||^2 \\ & & \\ \text{avec:} & & \\ & & \\ K^2(\alpha_i,\alpha_j) & = & \frac{1}{2}.\{d^2(\alpha_i,\alpha_{k_1})+d^2(\alpha_j,\alpha_{k_1})-d^2(\alpha_i,\alpha_j)\}/A(\alpha_i,\alpha_j,\alpha_k) \quad + \\ & & \\ & & \frac{1}{2}.\{d^2(\alpha_i,\alpha_{k_2})+d^2(\alpha_j,\alpha_{k_2})-d^2(\alpha_i,\alpha_j)\}/A(\alpha_i,\alpha_j,\alpha_k) \end{array}\right. \tag{3.25}$$

où k_1 et k_2 dénotent les deux sommets opposés des deux triangles qui partagent le côté (α_i, α_j).

Autrement dit, le critère de dispersion métrique qui définit les cartes harmoniques peut être considéré comme un ensemble de contraintes linéaires $\mathbf{c} = \mathbf{c}(\alpha_i, \alpha_j)$ définies à chaque côté $\mathcal{E} = (\alpha_i, \alpha_j)$ de la triangulation $\mathcal{G}$ par:

$$\sum_{\alpha\in\Omega} A_{\mathbf{c}}(\alpha).\varphi(\alpha) = b_{\mathbf{c}}$$

avec:

$$\left\{\begin{array}{lcl} A_{\mathbf{c}}(\alpha_i) & = & K(\alpha_i,\alpha_j) \\ & & \\ A_{\mathbf{c}}(\alpha_j) & = & -K(\alpha_i,\alpha_j) \\ & & \\ A_{\mathbf{c}}(\alpha) & = & 0 \quad \forall\alpha\notin\{\alpha_i,\alpha_j\} \\ & & \\ b_{\mathbf{c}} & = & 0 \end{array}\right.$$

Ceci permet de définir les termes $\gamma_{\mathbf{c}}$ et $\Gamma_{\mathbf{c}}$ à prendre en compte dans la forme locale de l'équation D.S.I. (voir la section précédente).

$$\left\{\begin{array}{lcll} \gamma_{\mathbf{c}}(\alpha) & = & K^2(\alpha_i,\alpha_j) & \forall\alpha\in\{\alpha_i,\alpha_j\} \\ & & & \\ \gamma_{\mathbf{c}}(\alpha) & = & 0 & \forall\alpha\notin\{\alpha_i,\alpha_j\} \\ & & & \\ \Gamma_{\mathbf{c}}(\alpha_i) & = & -K^2(\alpha_i,\alpha_j).\varphi(\alpha_j) & \\ & & & \\ \Gamma_{\mathbf{c}}(\alpha_j) & = & -K^2(\alpha_i,\alpha_j).\varphi(\alpha_i) & \\ & & & \\ \Gamma_{\mathbf{c}}(\alpha_j) & = & 0 & \forall\alpha\notin\{\alpha_i,\alpha_j\} \end{array}\right.$$

En utilisant D.S.I. avec un grand facteur de respect des contraintes, cet ensemble de contraintes va avoir un plus grand impact que la rugosité, et la solution obtenue sera une carte harmonique. Il est également possible de mod-

uler les coefficients de pondération des contraintes, ce qui permet par exemple d'obtenir une solution à mi-chemin entre la paramétrisation de Floater et les cartes harmoniques. Notre formalisme permet donc d'exprimer les critères d'autres méthodes, et d'en définir de nouveaux, comme nous l'avons montré en Section 3, où nous avons introduit les critères de préservation des distances et des angles.

3.4 Paramétrisation interactive

Les contraintes que nous avons définies jusqu'à maintenant permettent à l'utilisateur d'avoir un contrôle *global* sur la paramétrisation. Même si les contraintes de respect des angles et des distances peuvent être localement pondérées afin de spécifier les zones de la surface où les distortions doivent être minimisées en priorité, ceci peut ne pas être suffisant pour certaines applications, qui requièrent un contrôle plus fin de la paramétrisation. Par exemple, il peut être nécessaire d'aligner les détails de la texture avec des détails du modèle. Ceci peut être réalisé en spécifiant un ensemble de points de la surface pour lesquels les points correspondants de la texture ont été fixés. Par exemple, nous montrerons comment une courbe tracée sur la surface en 3D peut être mise en correspondance avec une courbe dessinée sur la texture, ce qui permet par exemple de mettre en correspondance les traits des visages d'acteurs virtuels avec des traits selectionnés sur des photographies de visages, utilisées comme textures. De plus, la surface à texturer peut présenter des déchirures, ou bien même être composée de plusieurs composantes connexes. Dans ce cas, l'utilisateur préfèrera utiliser une seule paramétrisation plutôt que d'avoir à découper la surface en plusieurs carreaux, comme dans [Blo85]. Dans le cadre de la géologie numérique, si la surface munie d'une paramétrisation doit être utilisée pour construire des grilles (voir Section 3.5.4), il peut être utile d'aligner les déchirures de la surface, qui correspondent aux failles, avec des courbes iso-paramétriques. Ceci permet d'éviter la formation de "marches d'escalier" sur la grille, au niveau de ces failles.

3.4.1 Spécifier une courbe iso-paramétrique

Comme nous le montrons Figure 3.13, nous considèrons qu'une courbe polygonale $L = \{\mathbf{p}_0, \ldots, \mathbf{p}_m\}$ est donnée, ainsi qu'une valeur u_0 du paramètre u. Nous décrivons ici les contraintes à honorer pour faire en sorte que la courbe iso-paramétrique définie par $(u = u_0)$ corresponde à la projection de la ligne polygonale L sur la surface $\mathbf{S}$. Nous allons ainsi associer à chaque point $\mathbf{p}_i$ de L une contrainte assurant que la courbe iso-paramétrique $u = u_0$ de la paramétrisation φ passe près de la projection $\mathbf{p}'_i$ de $\mathbf{p}_i$ sur $\mathbf{S}$.

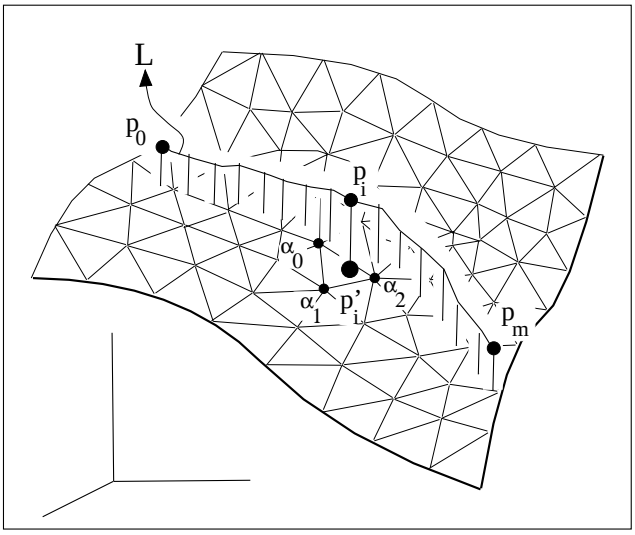


Figure 3.13: *Spécification d'une courbe iso-paramétrique.*

Le triangle $\mathbf{T} = (\alpha_0, \alpha_1, \alpha_2)$ est l'unique triangle de $\mathbf{S}$ qui contient $\mathbf{p}'_i$, et $(\lambda_0, \lambda_1, \lambda_2)$ dénotent les coordonnées barycentriques de $\mathbf{p}'_i$ dans $\mathbf{T}$. La relation linéaire à honorer est alors donnée par l'Équation 3.26 ci-dessous:

$$\left| \begin{array}{l} \sum\limits_{j\in\{0,1,2\}} \lambda_j.\varphi^u(\alpha_j) = u_0 \\ \\ \text{avec:} \quad \left\{ \begin{array}{l} \sum\limits_{j\in\{0,1,2\}} \lambda_j.\mathbf{p}(\alpha_j) = \mathbf{p}'_i \\ \\ \sum\limits_{j\in\{0,1,2\}} \lambda_j = 1 \end{array} \right. \end{array} \right. \tag{3.26}$$

L'expression de cette contrainte sous la forme générale des contraintes D.S.I. (voir Équation 3.17, section précédente) est donnée plus loin (Équation 3.27). Une telle contrainte est définie pour chaque point $\mathbf{p}_i$ de la courbe polygonale L à honorer.

$$\left\{ \begin{array}{lcll} A_{c_{\mathbf{p}_i}}(\alpha_j) & = & \lambda_j & \forall j \in \{0,1,2\} \\ A_{c_{\mathbf{p}_i}}(\alpha) & = & 0 & \forall \alpha \notin \{\alpha_0, \alpha_1, \alpha_2\} \\ b_{c_{\mathbf{p}_i}} & = & u_0 & \end{array} \right. \tag{3.27}$$

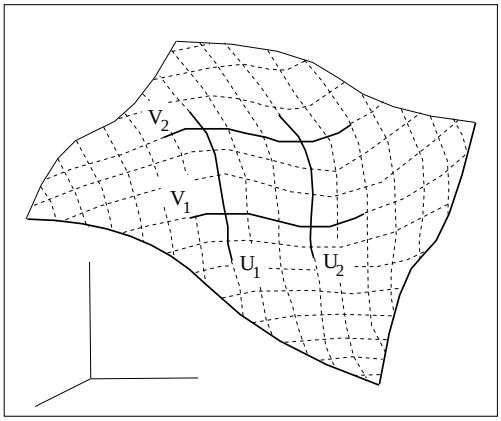


Figure 3.14: *Extrapolation d'une paramétrisation à partir de quatre courbes iso-paramétriques.*

Remarque 1:

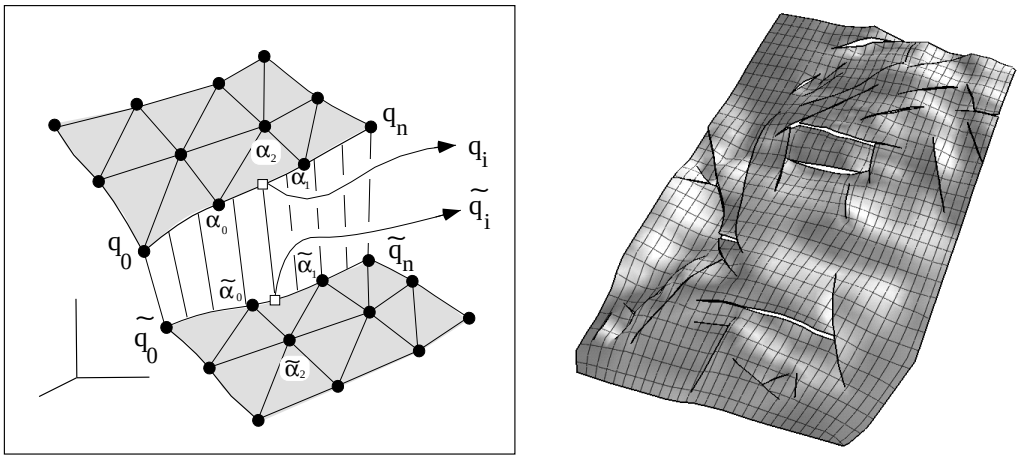
Comme nous l'avons remarqué en Section 2, les deux conditions **suffisantes** rappelées ci-dessous assurent que la fonction discrète φ définit bien une paramétrisation (à savoir une fonction bijective):

1. L'image du bord de la surface à travers φ dans le domaine (u, v) est un polygone convexe.
2. Chaque sommet interne est une combinaison convexe de ses voisins dans le domaine (u, v).

L'introduction des contraintes qui assurent le respect des angles et des distances (ou encore que les courbes iso-paramétriques sont orthogonales et espacées de manière homogène) autorise à remplacer la première condition par une condition moins restrictive. Comme nous pouvons le voir sur la Figure 3.14, il suffit alors de définir quatre arcs de courbes iso-paramétriques $\{u_1, u_2, v_1, v_2\}$, en utilisant la contrainte mentionnée auparavant. Ainsi, en rendant l'algorithme capable de se comporter non seulement comme un *interpolateur*, mais aussi comme un *extrapolateur*, permettant de construire des paramétrisations de surfaces à bords arbitraires, en laissant la paramétrisation non spécifiée le long des bords. De plus, comme la paramétrisation du bord est laissée libre, la méthode aura plus de degrés de liberté pour minimiser les distortions.

Remarque 2:

Cette contrainte peut être également utilisée pour spécifier les coordonnées (u, v) d'un point isolé de la surface (une contrainte iso-paramétrique limitée à un point). Nous verrons par la suite comment cette possibilité peut être utilisée dans le cadre de problèmes de placage de textures, pour faire correspondre des détails



A B

Figure 3.15: *A: Éléments entrant en jeu lors de la connection des deux bords d'une déchirure; B: Paramétrisation d'une surface géologique, construite à partir de données fournies par Gaz de France.*

de la texture avec des détails du modèle. Ceci peut être utilisé pour déplier des surfaces (voir Section 3.5.5).

3.4.2 Paramétrisation d'une surface discontinue

Nous considérons à présent des surfaces qui présentent des discontinuités, des déchirures, et pouvant même avoir plusieurs composantes connexes. Nous souhaitons définir une fonction de paramétrisation inverse Φ qui soit continue à travers ces déchirures. Plutôt que d'utiliser plusieurs éléments de surfaces distincts recollés ensuite, comme dans [Blo85], nous considérons ici la surface comme étant un élément unique, (telle qu'elle était avant d'être découpée), comme suggéré dans [CEM97]. L'ensemble de contraintes décrit plus loin permet de choisir des coordonnées (u, v) aux sommets d'une triangulation de telle sorte que les deux bords d'une discontinuité soient mis en correspondance avec la même courbe dans l'espace (u, v), à travers la fonction de paramétrisation inverse interpolée Φ. En d'autres termes, les déchirures sont *recousues* dans l'espace (u, v).

Comme nous le montrons sur la Figure 3.15-A, deux ensembles de points $\{\mathbf{q}_i, i = 0 \dots n\}$ et $\{\tilde{\mathbf{q}}_i, i = 0 \dots n\}$ sont échantillonés le long des deux bords de la discontinuité. Nous montrons maintenant comment assurer la correspondance de la paramétrisation au niveau de chaque paire de points $(\mathbf{q}_i, \tilde{\mathbf{q}}_i)$. Plus précisément, nous souhaitons respecter les conditions suivantes :

$$\forall \nu \in \{u,v\} \left| \begin{array}{lcl} (1) & \varphi_T^\nu(\mathbf{q}_i) & = \varphi_{\tilde{T}}^\nu(\tilde{\mathbf{q}}_i) \\ (2) & \mathbf{grad}\varphi_T^\nu & = \mathbf{grad}\varphi_{\tilde{T}}^\nu \end{array} \right. \tag{3.28}$$

où T et $\tilde{T}$ dénotent les deux triangles qui contiennent $\mathbf{q}_i$ et $\tilde{\mathbf{q}}_i$ respectivement. Le gradient $\mathbf{grad}\varphi_T^\nu$ est calculé comme décrit précédemment en Section 3 (voir Équation 3.11), en utilisant la base orthonormée locale que nous pouvons voir Figure 3.8.

En utilisant les méthodes introduites dans les deux sections précédentes, nous pouvons facilement traduire ces deux conditions en contraintes linéaires $c^\nu_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}$ et $c^{\nu W}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}$, dont les coefficients sont donnés Équations 3.29 et 3.30 respectivement.

$$\left| \begin{array}{lcll} A_{c^\nu_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\alpha_j) & = & \lambda_j & \forall j \in \{0,1\} \\ A_{c^\nu_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\tilde{\alpha_j}) & = & -\tilde{\lambda_j} & \forall j \in \{0,1\} \\ A_{c^\nu_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\alpha_j) & = & 0 & \forall \alpha \notin \{\alpha_0,\alpha_1,\tilde{\alpha_0},\tilde{\alpha_1}\} \\ b_{c^\nu_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\alpha_j) & = & 0 & \end{array} \right. \tag{3.29}$$

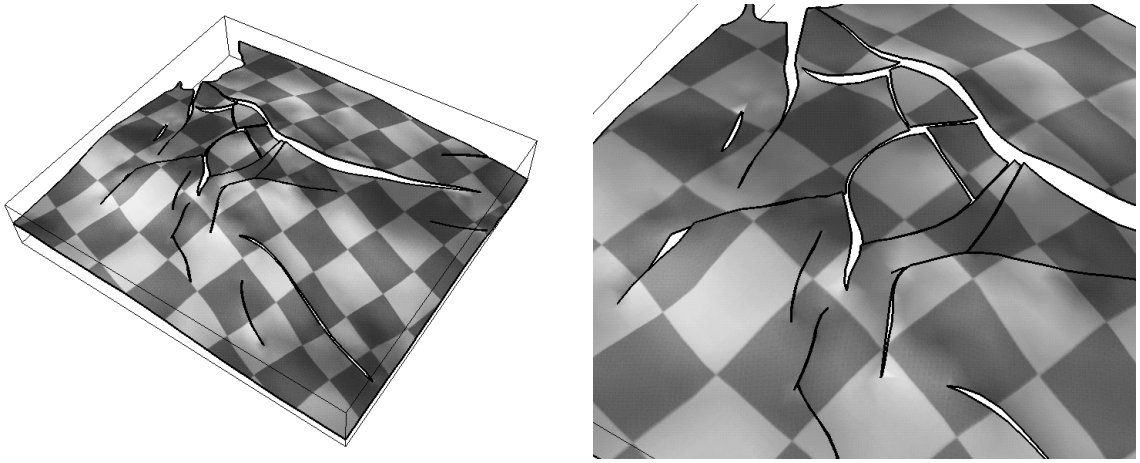
où $\lambda_{j,j\in\{0,1\}}$ et $\tilde{\lambda}_{j,j\in\{0,1\}}$ dénotent les coordonnées barycentriques de $\mathbf{q}_i$ dans $[\mathbf{p}(\alpha_0), \mathbf{p}(\alpha_1)]$ et $\tilde{\mathbf{q}}_i$ dans $[\mathbf{p}(\tilde{\alpha_0}), \mathbf{p}(\tilde{\alpha_1})]$ respectivement.

Les quatre contraintes $c^{uX}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}$, $c^{uY}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}$, $c^{vX}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}$, et $c^{vY}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}$ engendrées par l'Équation 3.30 assurent un gradient constant de la paramétrisation à travers la discontinuité. Autrement dit, la courbe iso-paramétrique qui passe par les points $\mathbf{q}_i$ et $\tilde{\mathbf{q}}_i$ a ses tangentes en ces deux points approximativement parallèles.

$$\left| \begin{array}{lcll} A_{c^{\nu W}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\alpha_j) & = & D_j & \forall j \in \{0,1,2\} \\ A_{c^{\nu W}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\tilde{\alpha_j}) & = & \delta_W.\tilde{D_j} & \forall j \in \{0,1,2\} \\ A_{c^{\nu W}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\alpha) & = & 0 & \forall \alpha \notin \{\alpha_0,\alpha_1,\alpha_2,\tilde{\alpha_0},\tilde{\alpha_1},\tilde{\alpha_2}\} \\ b_{c^{\nu W}_{\mathbf{q}_i,\tilde{\mathbf{q}}_i}}(\alpha) & = & 0 & \end{array} \right. \tag{3.30}$$

avec:

$$\nu \in \{u,v\} \quad ; \quad W \in \{X,Y\} \quad ; \quad \delta_W = \left\{ \begin{array}{ll} -1 & \text{if} \quad W = X \\ +1 & \text{if} \quad W = Y \end{array} \right.$$



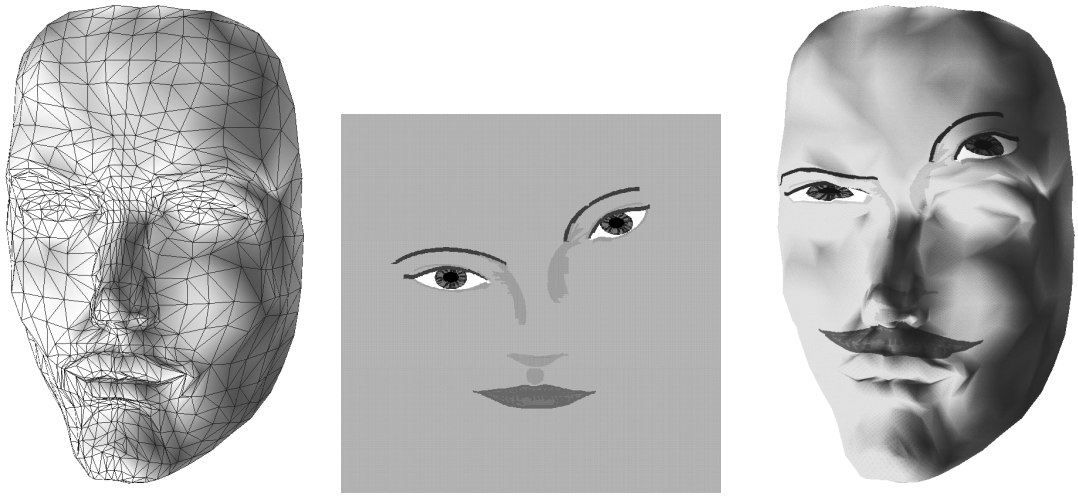
A B

Figure 3.16: *A: Une autre surface géologique plus complexe du point de vue des relations entre failles (données fournies par ELF-GRC); B: Zoom sur la surface, mettant en évidence la continuité du damier à travers les failles.*

Nous montrons quelques exemples de résultats obtenus sur des surfaces géologiques, en utilisant cette méthode. La Figure 3.15-B montre une surface construite à partir de données fournies par Gaz de France. Sur cette surface, les courbes iso-paramétriques ont été tracées, montrant la continuité de la paramétrisation à travers les failles. Une fois munie de cette paramétrisation, cette surface pourra servir de support à des propriétés physiques, ainsi qu'à des calculs numériques. Nous présenterons ces résultats en Section 3.5.4, ainsi qu'une méthode permettant de construire une grille 3D à partir d'une surface ainsi munie d'une paramétrisation. La Figure 3.16 présente un autre exemple de surface géologique faillée munie d'une paramétrisation continue à travers les failles.

3.5 Applications et résultats

Dans cette section, nous présentons différentes applications ainsi que les résultats associés. Plusieurs disciplines peuvent bénéficier d'une méthode de paramétrisation souple et efficace, comme le placage de textures, la création de maillages pour éléments finis, la génération de maillages optimaux suivant certains critères à partir d'un maillage donné, ou encore le dépliage de surfaces. Ces différentes applications ont été étudiées dans le cadre du projet Gocad, afin de permettre la modélisation précise et de certaines structures géologiques. Néanmoins, afin de faciliter la compréhension, nous utiliserons souvent comme exemples des surfaces telles que des visages humains, pour lesquels l'intuition reconnaît immédiatement les éléments caractéristiques.



A B C

Figure 3.17: *Plaquer sur un modèle (A) une texture arbitraire (B) peut poser certains problèmes de mise en correspondance (C), car rien ne garantit à priori que les détails de la texture vont correspondre aux détails du modèle.*

De plus, ceci montre des applications possibles de nos méthodes dans des contextes différents de celui de la géologie numérique.

3.5.1 Placage de textures sous contraintes

Comme nous le montrons sur la Figure 3.17, étant donné une texture et une surface, l'objectif est de plaquer la texture sur la surface en garantissant la correspondance entre certains éléments caractéristiques, spécifiés par l'utilisateur. Dans notre exemple, nous avons volontairement choisi une texture (Figure 3.17-B) ayant une forme très différente de l'objet à texturer (Figure 3.17-A), ce qui montre de manière évidente les problèmes rencontrés (Figure 3.17-C). Notre objectif est alors de générer une paramétrisation telle que les éléments caractéristiques de la texture et de la surface se correspondent. Afin d'atteindre cet objectif, Shirman *et. al.* [SK97] proposent de considérer une paramétrisation inverse comme la composée de deux fonctions :

- Une fonction π de $\mathbb{R}^3$ vers $\mathbb{R}^2$, mettant en correspondance la surface $\mathbf{S}$ avec le domaine paramétrique $\mathcal{D}$. Cette fonction n'a pas de contraintes particulières à respecter, si ce n'est qu'elle doit être inversible.

- Une fonction τ, appelée un τ-*mapping* , mettant le domaine paramétrique en correspondance avec lui même. Selon Shirman *et. al.*, c'est cette fonction τ qui va permettre de mettre les détails de la texture en correspondance avec ceux du modèle.

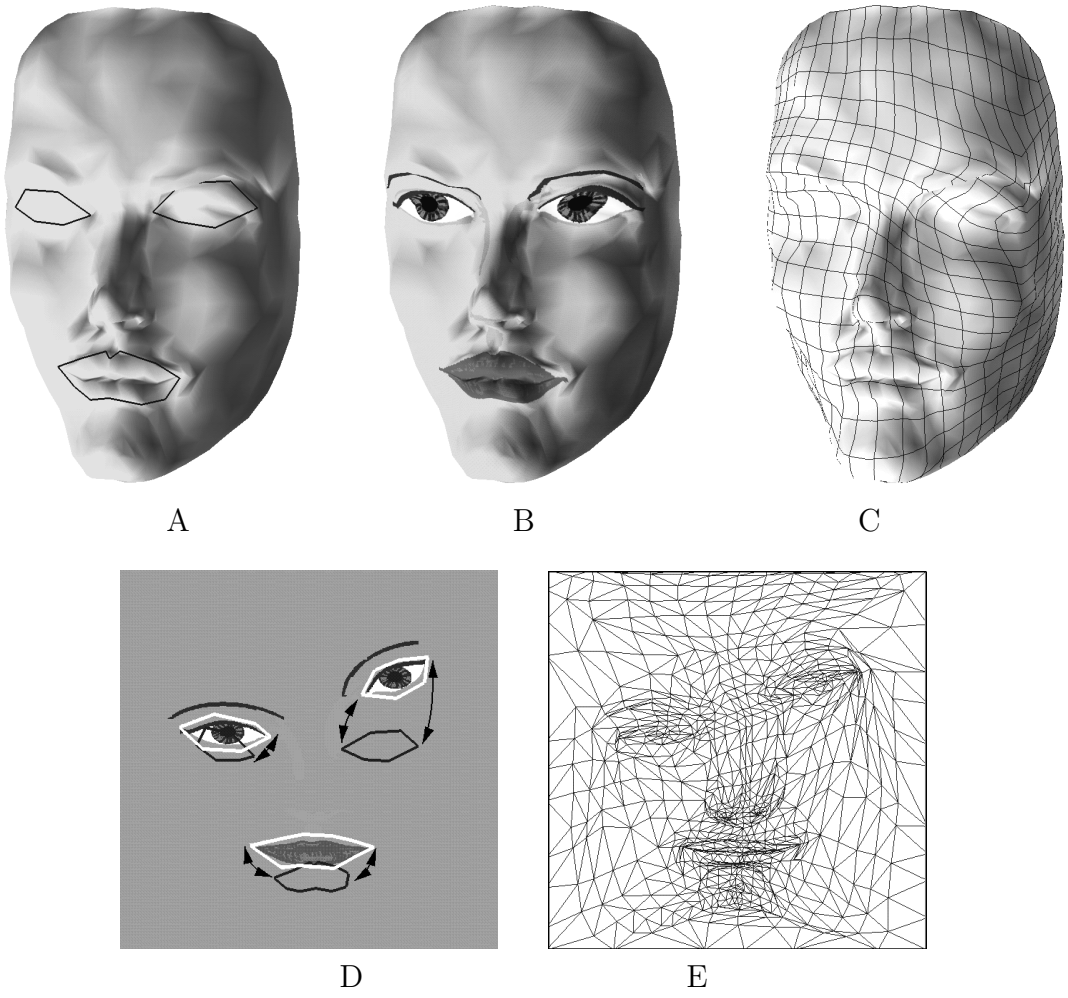


Figure 3.18: *Selection des éléments caractéristiques du modèle par l'utilisateur (A); Mise en correspondance avec ceux de la texture (D); Résultat (B); Courbes iso-paramétriques (C) et domaine paramétrique (E).*

La fonction τ peut être définie suivant différentes approches, souvent appelées *morphing* ou encore *warping* dans la littérature. Ces auteurs justifient cette approche par le fait que la construction de la fonction π est un problème difficile, et que dans un contexte de modeleur interactif, il est plus facile de mettre à jour la fonction τ en temps réel.

En réalité, comme nous le montrons ici, il n'est pas plus coûteux de contraindre une fonction π, mettant en correspondance la surface $\mathbf{S}$ avec le domaine paramétrique $\mathcal{D} \subset I\!R^2$, que de contraindre une fonction τ définie de $\mathcal{D}$ vers $\mathcal{D}$. À première vue, le premier problème semble être un problème tridimensionnel, mais ce n'est qu'en apparence, car la surface $\mathbf{S}$ est un objet bidimensionnel, que l'on cherche à mettre en correspondance avec le domaine $\mathcal{D}$: un autre objet bidimensionnel. Nous pouvons alors considérer que construire en une seule

étape une paramétrisation contrainte est l'équivalent, sur des surfaces courbes, des problèmes de déformations d'images (morphing/warping) traditionnels.

En utilisant la contrainte définie dans la partie précédente, il est possible de spécifier les coordonnées (u, v) que l'on souhaitera attribuer à n'importe quel point $\mathbf{p}$ de la surface $\mathbf{S}$ (même si le point $\mathbf{p}$ ne correspond pas à un sommet de la triangulation). Les éléments caractéristiques du modèle seront alors définis comme un ensemble de points $\mathbf{p}_i$, pour lesquels les coordonnées de texture (u_i, v_i) ont été spécifiées. Nous souhaitons alors construire une paramétrisation inverse Φ, telle que $\Phi(\mathbf{p}_i) = (u_i, v_i)$. Chacun des points $\mathbf{p}_i$ définit alors deux contraintes $c^u_{\mathbf{p}_i}$ et $c^v_{\mathbf{p}_i}$ exprimant cette condition (l'expression de la contrainte correspondante est donnée Équation 3.26, dans la section précédente).

Le problème reste alors de donner un moyen à l'utilisateur pour définir l'ensemble des contraintes à mettre en place, afin d'exprimer cette correspondance entre éléments caractéristiques. Nous supposons qu'une paramétrisation Φ^0 est déjà disponible, et a été construite en appliquant par exemple notre méthode, sans définir de contraintes. Sur la Figure 3.18-A, nous pouvons voir que l'utilisateur a séléctionné sur le modèle un ensemble de courbes polygonales, correspondant aux éléments caractéristiques. Ces courbes sont ensuite transformées dans le domaine (u, v) en utilisant la fonction de paramétrisation inverse Φ^0. Ces courbes sont dessinées en noir dans le domaine (u, v) (Figure 3.18-D). Comme nous pouvons nous y attendre, elles ne correspondent pas aux éléments caractéristiques de la texture. Pour chaque sommet de ces courbes, l'utilisateur va déterminer les coordonnées (u, v) souhaitées pour le sommet en question, par exemple en déplaçant les courbes noires de manière à les faire coïncider avec les courbes blanches (voir Figure 3.18-D). Le système, qui aura gardé en mémoire la position initiale des courbes, va les replacer à leur position initiale dans $\mathbb{R}^3$. Il est alors facile de mettre en place les deux contraintes $c^u_{\mathbf{p}_i}$ et $c^v_{\mathbf{p}_i}$ pour chaque sommet $\mathbf{p}_i$ des éléments caractéristiques. Après interpolation, nous obtenons le résultat de la Figure 3.18-B. Les yeux et la bouche de la texture ont bien été placés au niveau des éléments correspondants sur le modèle. Nous montrons également les courbes iso-paramétriques Figure 3.18-C. Ces courbes montrent bien de quelle manière la texture a été "tordue", pour remettre l'oeil gauche au bon emplacement. Inversement, si nous transformons la surface triangulée dans l'espace (u, v), nous pouvons voir que les éléments caractéristiques de la surface et de la texture se correspondent (Figure 3.18-E).

L'aspect itératif de l'algorithme introduit Section 3 joue ici un rôle important, permettant d'utiliser à chaque étape le résultat de l'étape précédente comme solution initiale. En effet, l'utilisateur pourra contraindre les éléments caractéristiques du modèle un par un, et contrôler le résultat à chaque étape. Comme l'introduc-

tion d'un élément nouveau modifie peu la paramétrisation, peu d'itérations seront nécessaires pour mettre à jour la paramétrisation à partir de celle obtenue à l'étape précédente. Ainsi, sur un ordinateur comportant un processeur de type R4000 à 250 MHz, pour le modèle qui comporte approximativement 6000 triangles, la paramétrisation initiale Φ^0 est construite en une trentaine de secondes, et chaque mise à jour à la suite de l'introduction d'un nouvel élément caractéristique prend quelques secondes.

3.5.2 Triangulation adaptative et décimation de maillages

Les surfaces que nous manipulons dans un modeleur 3D sont de sources diverses, résultat de différents algorithmes appliqués à différentes sortes de surfaces. Pour en citer quelques exemples, nous avons souvent à manipuler des surfaces résultant:

- de l'échantillonage de surfaces paramétriques;
- de la numérisation d'objets à l'aide de scanners 3D;
- de surfaces géologiques, définies par un grand ensemble de points, obtenus par des méthodes sismiques;
- de la triangulation d'iso-surfaces par la méthode *Marching Cubes* [LC89] ou ses dérivés;
- de la triangulation d'ensembles de points parfois très denses ...

Ces surfaces ont parfois un nombre de triangles très important (Figure 3.19-A), et l'utilisateur souhaitera alors les *décimer*, à savoir supprimer des triangles sans altérer la géométrie de l'objet, pour en obtenir une représentation géométriquement équivalente et moins lourde en mémoire (Figure 3.19-D). Comme nous l'avons mentionné dans la première partie, les approches classiques pour ce type de problèmes consistent à partir de la surface initiale, comportant un grand nombre de triangles, et à appliquer itérativement une opération permettant de supprimer localement un ensemble de triangles (voir [Dee95, KCVS98, SZL92, TGHL98, LSS+98]). Ces méthodes sont également utilisées pour construire une représentation multi-résolution des surfaces [EDD+95, LDW97, ZSS97].

L'inconvénient majeur de ce type d'approche est qu'elles sont très dépendantes de la triangulation initiale, puisque le maillage décimé est obtenu à partir de modifications de celle-ci. Nous proposons ici une approche différente, qui nous permet de nous libérer de cette triangulation initiale, en opérant dans le domaine paramétrique (u, v). Si nous considérons une triangulation arbitraire du domaine (u, v), alors son image par la paramétrisation $\mathbf{x}$ est une triangulation dont tous les sommets sont sur la surface initiale $\mathbf{S}$. Cette propriété nous permet de considérer

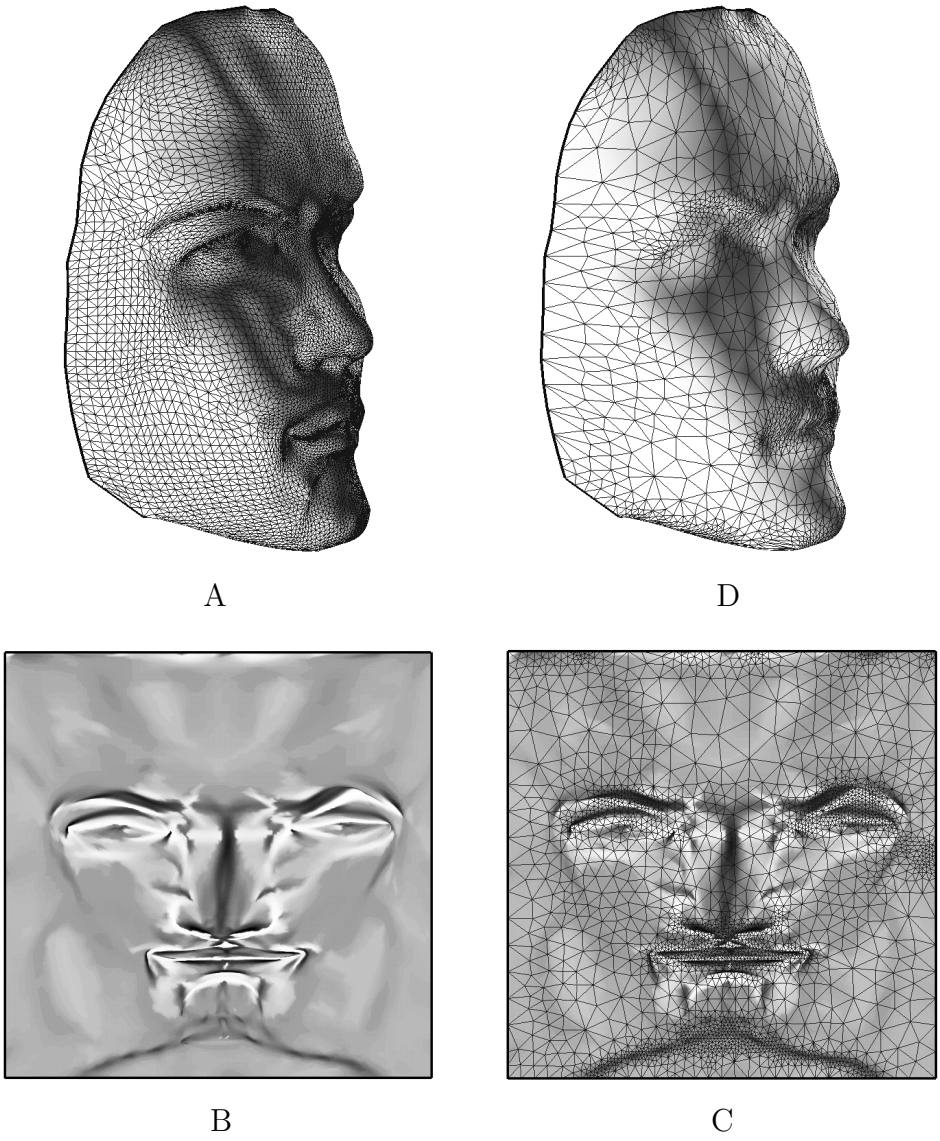


Figure 3.19: *Re-triangulation adaptative. A: Surface comportant un grand nombre de triangles; B: Carte des courbures dans le domaine paramétrique; C: Triangulation adaptative de la carte des courbures; D: Triangulation transformée dans* $\mathbb{R}^3$ *en appliquant la paramétrisation* $\mathbf{x}$.

le problème de re-triangulation d'une surface $\mathbf{S}$ de $\mathbb{R}^3$, difficile à résoudre dans $\mathbb{R}^3$, comme un problème bidimensionnel de triangulation dans $\mathbb{R}^2$. Ainsi, nous transformons le problème depuis l'espace $\mathbb{R}^3$ où il est posé dans un espace - le domaine paramétrique - où il devient plus simple à résoudre. Il est ensuite facile de revenir dans $\mathbb{R}^3$ en appliquant la paramétrisation $\mathbf{x}$ à la triangulation du domaine (u, v).

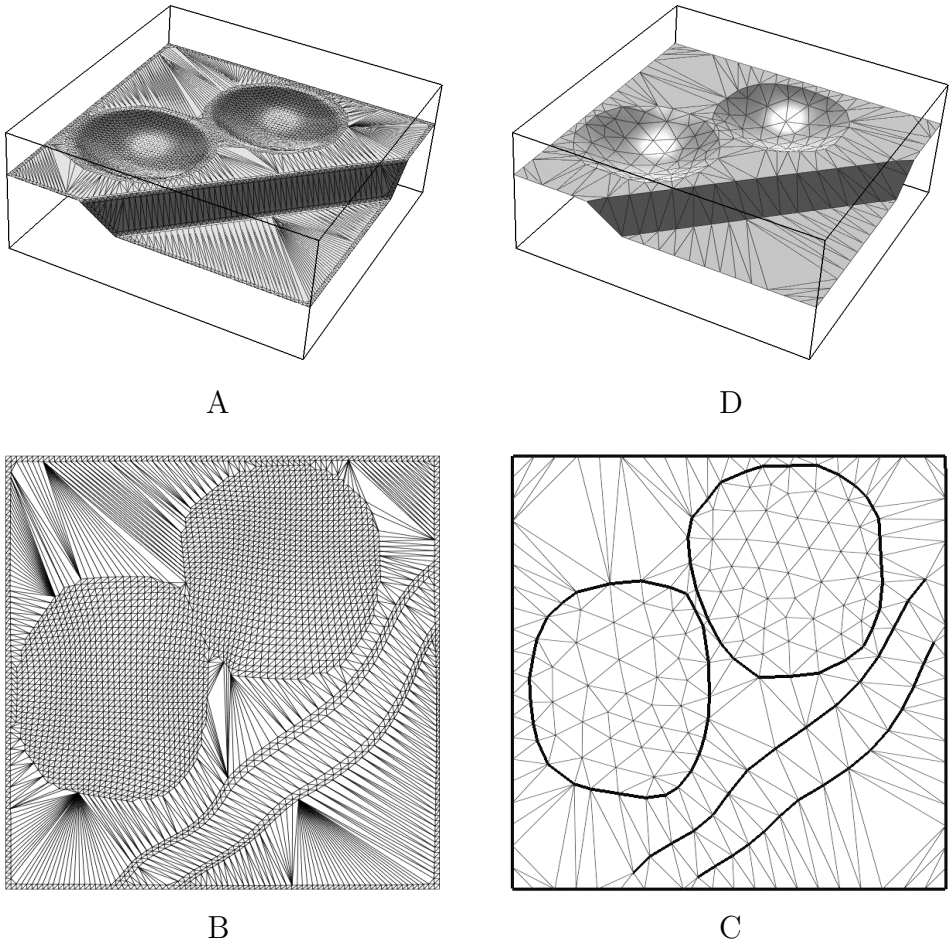


Figure 3.20: *Re-triangulation contrainte. A: Objet type C.A.O., comportant des angles et arêtes vives (6010 triangles); B: Objet transformé dans l'espace paramétrique; C: Triangulation contrainte de l'espace paramétrique, respectant les arêtes vives (en traits gras); D: Triangulation transformée dans $\mathbb{R}^3$ en appliquant la paramétrisation* $\mathbf{x}$ *(388 triangles).*

Une surface triangulée munie d'une paramétrisation $\mathbf{x}$ peut être considérée comme n'importe quelle autre surface paramétrique. Nous pouvons donc penser lui appliquer des méthodes de triangulation initialement prévues pour des surfaces polynomiales (voir par exemple [BG96]). Ici, nous avons choisi de densifier la triangulation en fonction de la courbure maximale de la surface. Nous montrons sur la Figure 3.19-B cette courbure maximale cartographiée dans le domaine paramétrique. Une estimation de la courbure d'une surface triangulée est donnée dans [Ham] et dans [Sam96].

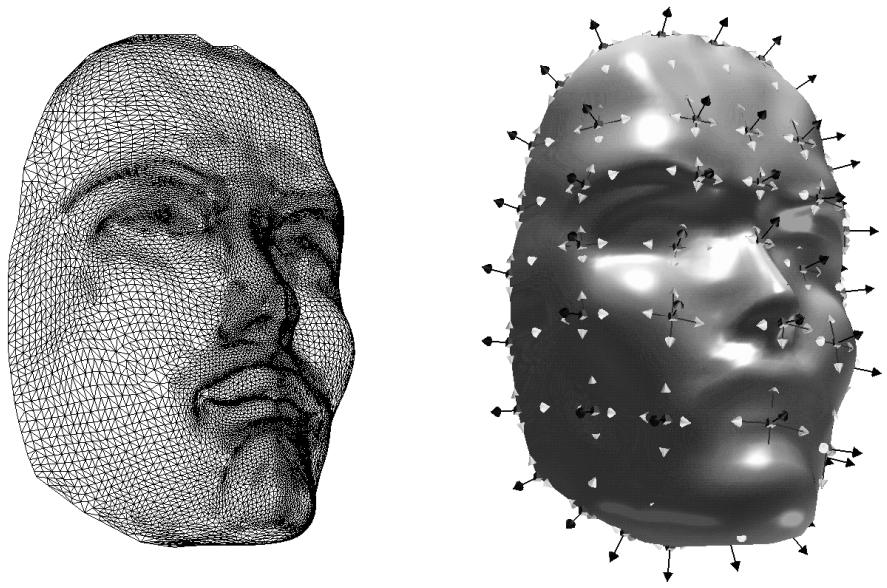
Nous pouvons ensuite trianguler le domaine paramétrique en créant plus de triangles dans les zones de forte courbure, en appliquant une méthode de triangulation anisotrope. Ce dernier point dépasse le cadre de ce travail, et le lecteur

pourra se référer par exemple à [BGH+95a, BGH+95b, CDHM95, VHM91, Val92] pour plus de détails concernant ce point précis. Pour générer la triangulation de la Figure 3.19-C, nous avons réalisé un échantillonage adaptatif de type arbre quaternaire, triangulé ensuite par Delaunay. Cette triangulation a été lissée par D.S.I., utilisée ici comme un lisseur Laplacien [Han95] (voir le deuxième chapitre). Certaines des méthodes de triangulation anisotrope devraient permettre d'obtenir un meilleur résultat, en prenant en compte une *carte de métrique*. Autrement dit, il est possible de prendre en compte une définition locale de la forme et de la taille des triangles souhaitées. En prenant pour métrique le tenseur métrique fondamental de la paramétrisation $\mathbf{x}$ (voir Section 3), il serait possible d'annuler totalement les distortions qui apparaissent lorsque l'on repasse du domaine paramétrique à $I\!R^3$ (nous pouvons voir par exemple sur la Figure 3.19-D que les triangles sont légèrement allongés près du bord de la surface).

Comme indiqué dans [LSS+98], certaines formes présentent des détails particuliers, comme des angles, qui doivent être préservés lors du processus de décimation. C'est le cas des deux arêtes vives et des deux zones circulaires de la surface que nous montrons Figure 3.20-A. C'est également le cas des bords internes de la surface, si celle-ci comporte des trous (par exemple les yeux d'un visage). Lee *et. al.* [LSS+98] proposent une décomposition en sous-domaines qui respecte certains côtés spécifiés par l'utilisateur. Dans notre approche, les détails à préserver sont spécifiés comme des lignes polygonales, liant un ensemble de sommets de la surface. Ces lignes sont transformées dans le domaine (u, v), et après ré-échantillonage, positionnées comme contraintes pour la triangulation. Nous pouvons voir l'espace paramétrique re-triangulé Figure 3.20-C, où les lignes épaisses représentent les contraintes. Le résultat final (Figure 3.20-D) compte bien moins de triangles que la surface initiale (388 au lieu de 6010), tout en respectant les arêtes vives du modèle. Il est clairement possible de combiner cette approche avec la triangulation adaptative mentionnée au paragraphe précédent, permettant de respecter à la fois des éléments caractéristiques et une carte de métrique.

3.5.3 Conversion de surfaces triangulées en surfaces polynomiales

Comme indiqué dans [KL96], une fois qu'une paramétrisation d'une surface triangulée a été définie, il est facile de la convertir en une surface polynomiale. Nous montrons Figure 3.21 une Spline bi-cubique (Figure 3.21-B) créée à partir d'une surface triangulée (Figure 3.21-A). La paramétrisation permet de découper facilement la surface en carreaux bi-cubiques, et d'ajuster ces carreaux à la surface tout en préservant la continuité C^1 entre les carreaux. D'autre types de surfaces polynomiales de degré plus élevé peuvent être utilisées, si une continuité



A B

Figure 3.21: *Une paramétrisation permet de convertir une surface triangulée (A) en une spline bi-cubique (B)*

d'ordre supérieur à C^1 est souhaitée. Dans le cadre de la géologie numérique, des méthodes telles que le lancer de rayons sismiques. De telles méthodes requièrent de grands ordres de continuité pour les surfaces qui représentent les interfaces entre couches géologiques. La possibilité de convertir une surface triangulée en surface polynomiale permet de bénéficier de la souplesse des surfaces triangulées pendant le processus de modélisation, et des grands ordres de continuité des surfaces polynomiales lors des calculs numériques effectués sur le modèle.

3.5.4 Création de grilles pour calculs numériques

Dans le cadre de la géologie numérique, notre méthode de paramétrisation a été utilisée pour définir un processus de modélisation, permettant de construire des grilles 3D. Ces grilles 3D serviront ensuite de support pour la modélisation de propriétés physiques définies sur le domaine d'étude. Ici, nous nous intéressons à un cas d'étude fournit par Gaz de France, dans l'objectif d'étudier les possibilités de stockage de gaz souterrain dans un réservoir naturel. Un tel réservoir est une couche de roche poreuse (telle que du grès), surmontée d'une couche de roche imperméable (de l'argile, le plus souvent). La Figure 3.22 montre une carte du toit du réservoir (autrement dit, les courbes de niveau de la surface définissant la limite supérieure du réservoir), ainsi que deux coupes verticales. Afin de construire un modèle de ce réservoir, nous disposons des données suivantes :

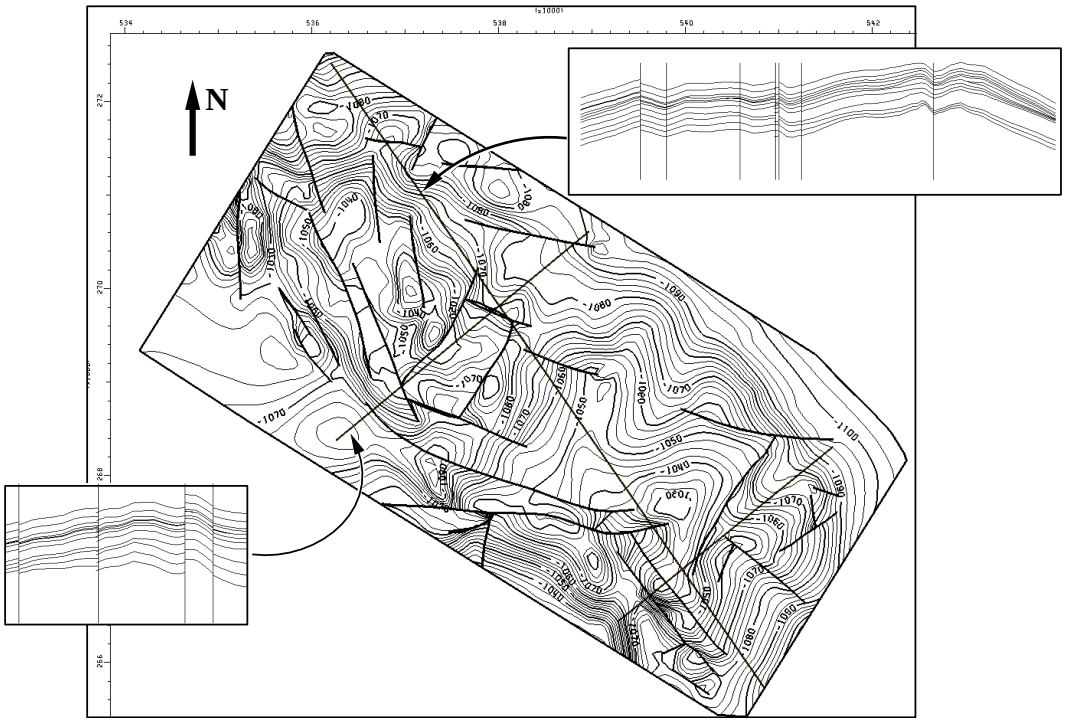


Figure 3.22: *Carte du réservoir modélisée à partir de données fournies par Gaz de France*

- Des données de puits, autrement dit, une information très fiable sur l'ensemble des couches géologiques à un endroit précis. Sur les puits sont identifiés un certain nombre de marqueurs, représentant non seulement la position exacte des couches à l'intersection du puits, mais également son pendage (le vecteur normal à l'interface);
- Une carte de courbes iso-bathes (iso-profondeur), définissant le toit du réservoir. Cette carte est définie comme un ensemble de courbes polygonales, que la surface devra respecter (voir Figure 3.23-A). Cette information est bien moins fiable que les marqueurs de puits, car elle est le résultat d'un processus d'interprétation;
- Pour chacune des couches à modéliser, une carte des courbes iso-paques (iso-épaisseur) a également été fournie;
- Un ensemble de courbes polygonales qui correspondent aux failles.
- Pour chacune des couches, nous avons également accès à une carte de porosité, également représentée sous forme de courbes d'iso-valeur.

Notre objectif est de construire à partir de ces données une grille représentant de manière précise la variation de la porosité. Les failles seront modélisées dans

cette grille par des faces décollées, dont la topologie est gérée par la méthode décrite en première partie. Pour ce modèle particulier, dont l'épaisseur est relativement faible, les failles peuvent être approximées par des surfaces verticales, obtenues par extrusion des courbes polygonales fournies (Figure 3.23-B). Ce dernier point n'est pas une limitation de notre méthode, nous montrerons plus loin un autre modèle pour lequel les failles n'ont pas été approximées par des surfaces verticales. Notre premier objectif est de construire un modèle structural, à savoir un ensemble de surfaces triangulées qui respectent la géométrie du domaine d'étude.

Construction du modèle structural

Comme nous l'avons déjà mentionné, plusieurs types de données sont disponibles pour construire le modèle. Les limites entre couches géologiques (aussi appelées *horizons*) sont représentées par des nuages de points de données, échantillonés sur des cartes d'iso-bathes (iso-profondeur), et la position des failles est indiquée par des lignes polygonales. Comme le réservoir concerné est relativement fin, nous considérons que les failles peuvent être ici approximées par des surfaces verticales. Cette considération simplifie le processus de construction du modèle, mais ne constitue en aucun cas une limitation de la méthode proposée. Comme nous disposons d'une carte de profondeur pour l'horizon correspondant au toit du réservoir (à savoir, la surface limite supérieure de la couche géologique qui nous intéresse), nous utiliserons cette surface comme référence dans le reste du processus de modélisation. Il est ensuite facile de définir tous les autres horizons du modèle à partir des cartes d'iso-paques, décrivant l'épaisseur des différentes couches. Ainsi, une fois que l'horizon de référence est construit, nous pouvons facilement obtenir le reste du modèle structural.

La surface triangulée représentant le toit du réservoir est construite à partir des marqueurs de puits, de la carte des iso-bathes et des traces de failles. La méthode est décrite dans la suite d'étapes suivantes, celles-ci sont détaillées Figure 3.23.

1. Une première surface triangulée qui approxime globalement la forme de l'horizon à modéliser est construite en triangulant l'enveloppe convexe de l'ensemble des points de données et des marqueurs de puits. La surface obtenue après cette première étape ne respecte pas encore les données.

2. Les marqueurs de puits devant être respectés exactement, ils sont insérés dans la triangulation, comme nouveaux sommets;

3. la surface est alors "attirée" par les points de données, en appliquant D.S.I. et en utilisant la contrainte d'ajustement aux données que nous avons présentée dans la deuxième partie. Cette méthode fait agir les points de

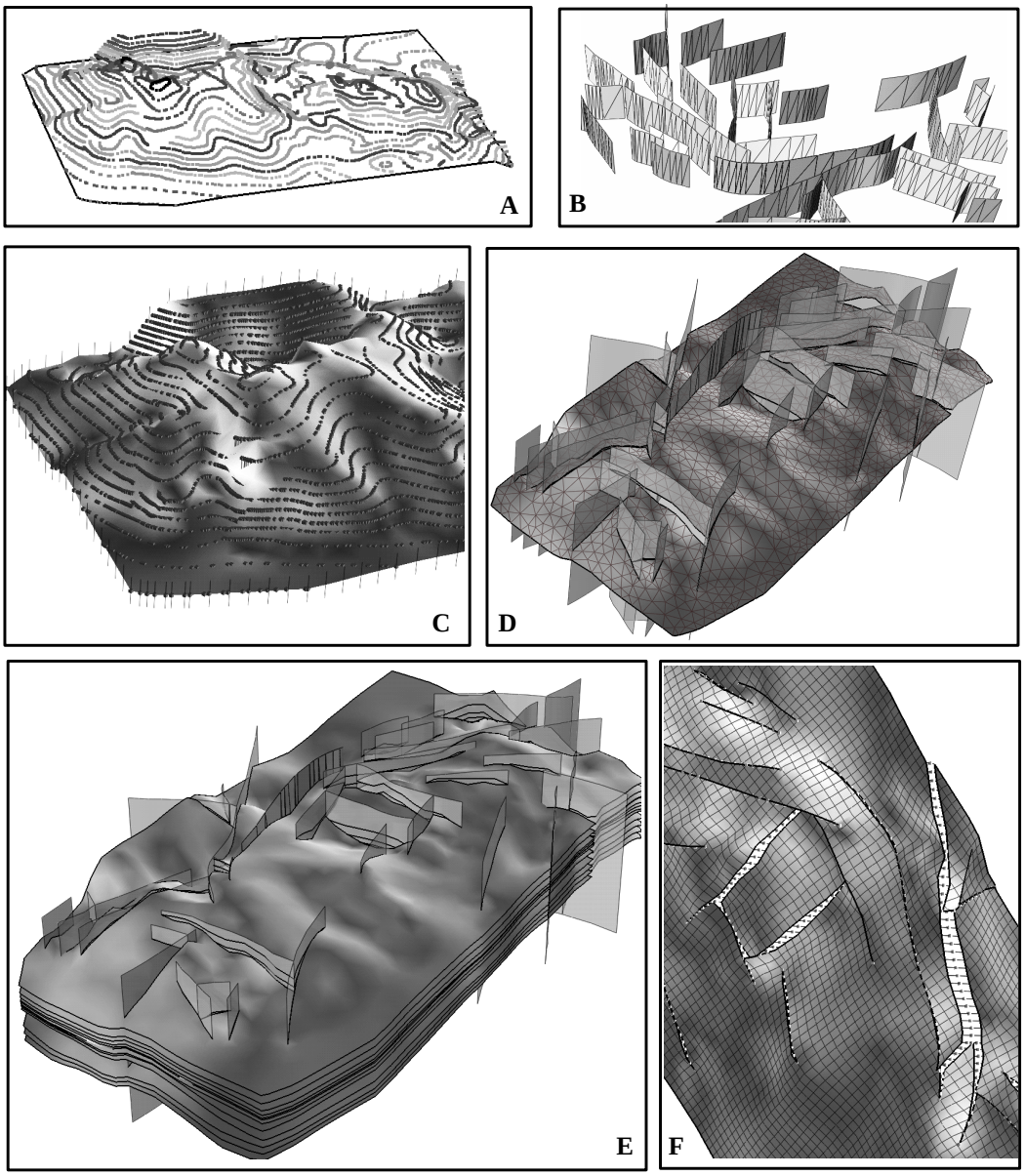


Figure 3.23: *Processus de modélisation du réservoir*

données comme des "aimants" qui attirent la surface. Des contraintes sont positionnées sur le bord de la surface, n'autorisant que des déplacements verticaux (sans quoi la surface rétrécierait). À la fin de cette étape, la surface respecte les données de puits ainsi que la carte iso-bathe, mais elle est toujours continue, et ne prend pas en compte les failles (voir Figure 3.23-C);

4. étant donné que les failles sont supposées verticales, les surfaces correspondantes peuvent être facilement construites par extrusion verticale. Les sur-

faces résultantes (Figure3.23-B) sont utilisées pour découper l'horizon de référence, à la manière d'un "emporte-pièce". Cette opération ne modifie que le maillage de la surface, mais pas sa géométrie, qui reste continue. Des opérations supplémentaires sont nécessaires pour modéliser les *rejets de failles*, à savoir les déplacements autour des failles;

5. avant de créer les rejets de failles, la surface doit subir certains traitements. Étant donné que cette surface est supposée servir de support à des calculs numériques, les triangles doivent être le plus équilatéraux possible, ce qui assure un bon comportement des méthodes numériques. Comme les calculs d'intersection de l'étape précédente génèrent parfois des triangles aplatis le long des bords des failles, ces bords doivent être filtrés, afin d'éliminer ces triangles aplatis à l'aide de re-triangulations locales. Cette opération coûte peu en termes de temps de calculs, puisque peu de triangles sont concernés;

6. nous pouvons alors modéliser les rejets de faille, en ré-interpolant la surface. Les sommets de l'horizon de la surface correspondant aux bords des failles sont contraints de manière à n'autoriser que des déplacement le long des surfaces de faille. Le résultat est une surface triangulée respectant à la fois les marqueurs de puits, les points de données et les traces de failles (Figure 3.23-D).

7. Une fois que l'horizon de référence, qui représente le toit du réservoir, a été modélisé, les autres horizons représentant les limites entre les différentes couches du modèle peuvent être facilement construits. Ces horizons sont simplement obtenus en recopiant l'horizon de référence et en lui appliquant la somme des cartes iso-paques qui indiquent l'épaisseur des différentes couches situées en dessous de la couche considérée.

Construction d'une grille 3D

Afin de permettre une représentation de grandeurs physiques définies sur l'ensemble du volume d'étude, nous allons à présent construire une grille curviligne 3D, respectant la géométrie du modèle. Cette grille pourra servir de support pour la représentation et la simulation de propriétés physiques, telles que la porosité, la perméabilité, des amplitudes sismiques ... La grille construite devra également correspondre de manière précise aux discontinuités des couches, rencontrées dans les régions de failles. Pour atteindre cet objectif précis, nous utilisons une structure de grilles faillées, qui permettent d'étendre le domaine de modélisation des grilles régulières. Ainsi, au niveau de la grille, il sera possible de modéliser les failles par un ensemble de cellules déconnect'ees.

Pour construire la grille à partir du modèle structural, nous commençons par définir une paramétrisation de l'horizon de référence, dont nous montrons

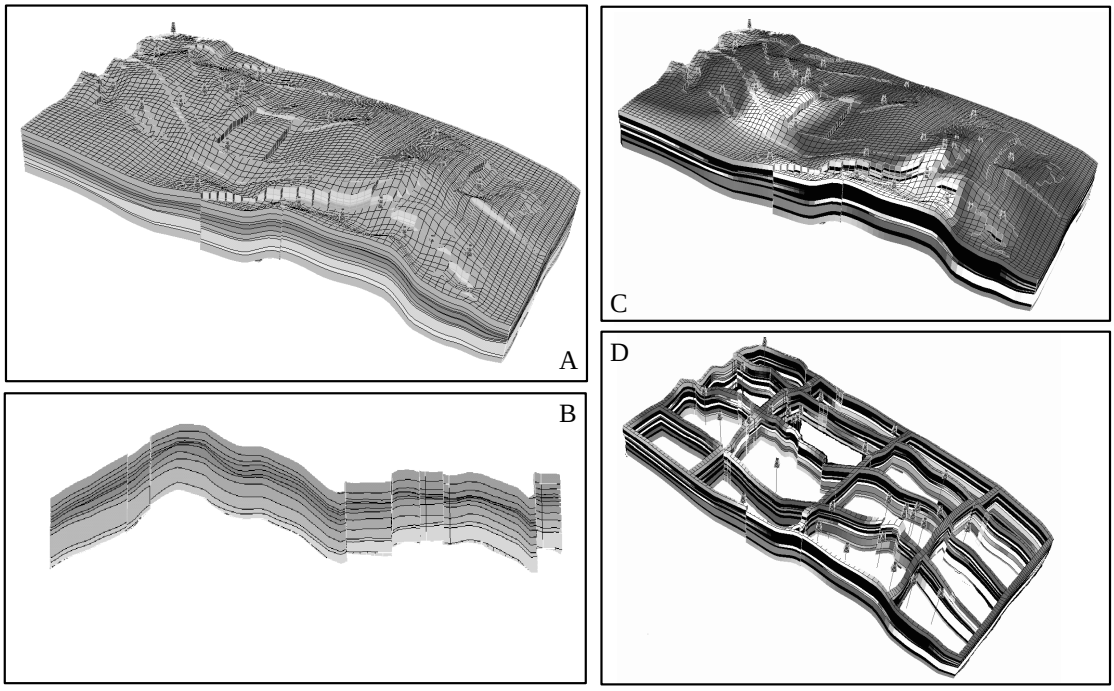


Figure 3.24: *Grille 3D faillée modélisée à partir de données fournies par Gaz de France*

les courbes iso-paramétriques sur la Figure 3.23-F. Ensuite, par échantillonage régulier dans le domaine (u, v), il est facile de construire la grille 3D que nous montrons sur la Figure 3.24-A. Les faces à décoller dans les régions des failles sont déterminées en appliquant l'algorithme de tracé de lignes de Bresenham [FvFH90] dans le domaine (u, v). La Figure 3.24-B montre une coupe de la grille, qui correspond de manière précise à la stratigraphie du réservoir. En "peignant" chaque couche de la grille avec les cartes de porosité, nous obtenons le résultat de la Figure 3.24-C. Sur cette figure, nous pouvons voir l'alternance entre les couches d'argile imperméables, en noir, et les couches de grès poreux, en niveaux de gris. La Figure 3.24-D montre un ensemble de coupes de la grille, où nous pouvons également distinguer cette structure en "mille-feuille".

Cette grille pourra ensuite être utilisée dans le cadre de simulations numériques, permettant d'évaluer les possiblités de stockage de gaz dans cette structure géologique. Les simulateurs associés ont certaines exigences en ce qui concerne la géométrie des cellules, qui peuvent s'exprimer en terme de contraintes sur le respect des distances et des angles par la paramétrisation. Notre méthode est donc bien adaptée à la construction de telles grilles, respectant ces critères.

Sur la Figure 3.25, nous montrons un autre exemple de grille modélisée en utilisant cette méthode, appliquée à un modèle plus complexe fourni par ELF-GRC. Pour cette grille, qui comporte de grandes failles, nous avons positionné

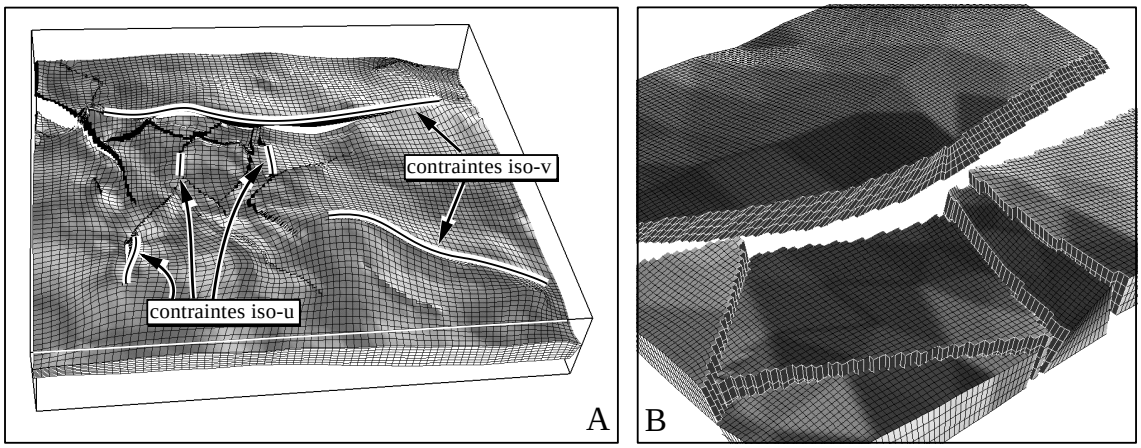


Figure 3.25: *Grille 3D faillée modélisée à partir de données fournies par ELF-GRC*

les contraintes iso-paramétriques sur certaines de ces failles (voir Figure 3.25-A). Ceci permet d'éliminer les effets de marche d'escalier près de ces failles. La Figure 3.25-B montre une grille générée à partir d'une surface plus petite extraite du modèle. Sur cette Figure, nous pouvons voir que les failles peuvent être obliques, et que des relations complexes entre failles peuvent être prises en compte.

3.5.5 Dépliage de surfaces géologiques

La modélisation géologique est un processus *itératif* , au cours duquel plusieurs modèles du sous-sol peuvent être construits et testés dans un cycle de validation-modification. Dans ce contexte, il est très utile de pouvoir définir des critères permettant de tester la validité d'un modèle géologique. Le dépliage de surfaces, encore appelé *restoration*, constitue l'un de ces tests de validation que les géologues utilisent pour rejeter les modèles invalides. Ainsi, la validité d'un modèle géologique peut être testée en analysant la répartition des déformations qui apparaissent lorsque les différentes surfaces qui composent le modèle sont dépliées.

Une revue des différentes méthodes de dépliages existant dans le domaine de la géologie a été réalisée par Samson dans [Sam96] (dont ce paragraphe est inspiré) et par Moretti dans [Mor91]. Historiquement, le problème du dépliage de modèles géologiques a été formalisé en 1969 dans [Dah69], où des règles empiriques de préservation du volume ont été remplacées par des critères plus précis, correspondant à la préservation du volume de couches sédimentaires et à la préservation de l'aire de surfaces sédimentaires. Depuis, un certain nombre de logiciels permettant de réaliser cette opération en combinant certaines transformations géométriques ont été écrits, comme par exemple le logiciel RESTORE [SED91], utilisable par un géologue généraliste sur un ordinateur de petite capacité, ou encore le logiciel LOCACE [Mor91], destiné à des utilisateurs

spécialistes du domaine, et offrant une plus grande puissance. D'autres approches ont été mises en œuvre dans [GGD91, GG93, RCea93, RSBC96, BCR+97], en discrétisant les surfaces à déplier en surfaces élémentaires, considérées comme des blocs rigides. Ces blocs sont ensuite déposés sur une surface plane, et ajustés les uns par rapport aux autres, en minimisant les interstices entre les blocs au sens des moindres carrés. Le même type d'approches a été appliqué à des coupes, dans [HS97]. La répartition des interstices résiduels permet de caractériser les surfaces, et de valider les modèles géologiques correspondant. Il est également possible d'utiliser la méthode de placage de texture décrite par Bennis *et. al.* [BVI91], évoquée dans l'introduction du chapitre. Cette méthode a été adaptée par son auteur pour résoudre des problèmes de dépliage de surfaces géologiques, ce qui montre le caractère connexe des deux disciplines (placage de textures et dépliage de surfaces).

Comme nous l'avons dit dans la Section 3.3, seules les surfaces d'une classe particulière, dites *développables* , peuvent être mises à plat sans déformations. Les méthodes précédement citées gèrent ces déformations en "déchirant" la surface dans les zones de fortes courbures [BVI91], ou encore en autorisant l'apparition d'interstices et de zones de recouvrement entre des éléments discrets autonomes les uns par rapport aux autres [RCea93]. Ici, nous allons appliquer une méthode différente, fondée sur une paramétrisation des surfaces, ce qui revient à répartir les déformation sur la surface, en résolvant un problème d'optimisation globale. Nous montrerons alors comment caractériser ces déformations résiduelles, afin d'utiliser notre méthode dans un cadre de validation de modèles géologiques.

Le dépliage d'une surface va alors être défini par les propriétés suivantes :

- Le domaine (u, v) correspond à la surface dépliée, inconnue au début du processus.
- Les failles sont fermées lors du dépliage.
- Les aires doivent être préservées.

En utilisant notre méthode, il est possible d'extrapoler une paramétrisation à partir d'une base curviligne spécifiée par l'utilisateur (voir Section 3.4.1), ce qui va nous permettre de satisfaire ces différents critères. Laisser la paramétrisation libre le long du bord de la surface laisse suffisamment de degrés de liberté pour minimiser les déformations. Afin d'améliorer la vitesse de convergence de l'algorithme itératif, il est possible d'effectuer un pré-traitement, "propageant" la base curviligne définie par l'utilisateur. Ce pré-traitement se réalise facilement, à l'aide d'un algorithme similaire aux algorithmes de remplissage 2D à partir d'un germe, en utilisant une pile (voir par exemple [FvFH90]).

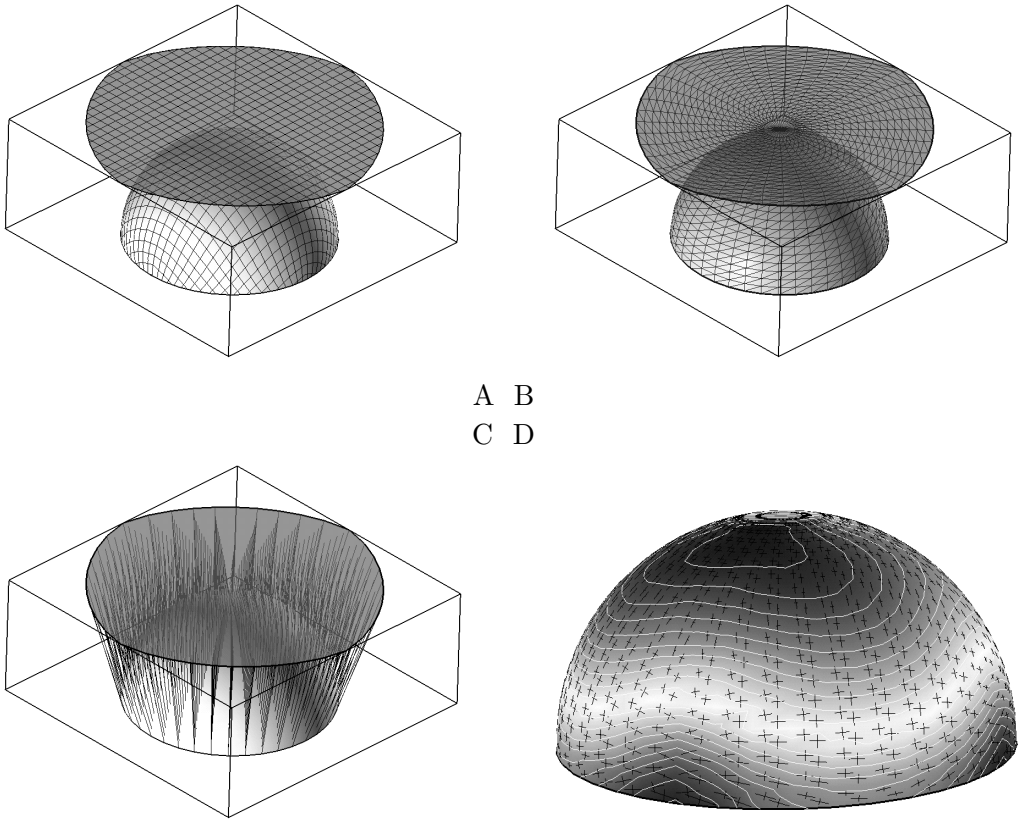


Figure 3.26: *Demi-sphère dépliée à l'aide d'une paramétrisation; A: demi-sphère et surface dépliée, les iso-paramétriques sont affichées; B: les mêmes surfaces, avec leur maillage affiché; C: champ de déplacement connectant les deux surfaces; D: module du tenseur de déformation de Green-Lagrange (niveaux de gris) et vecteurs propres du tenseur de déformation (croix noires); ici, les déformations sont plus fortes près du pôle et près de l'équateur.*

Nous montrons sur la Figure 3.26 le résultat obtenu avec une demi-sphère, surface non développable. Les Figures 3.26-A,B montrent la demi-sphère et la surface dépliée. La Figure 3.26-C montre le champ de déplacement, défini comme le champ de vecteurs liant la surface initiale à la surface dépliée. Ce champ de déplacement est utilisé par les géologues structuralistes pour visualiser les différents mouvements de blocs lors d'un dépliage. Les informations présentées par la Figure 3.26-D permettent de mettre en évidence les déformations liées au dépliage, comme nous le montrons plus loin.

Une fois qu'une surface est munie d'une paramétrisation, il est alors facile de la déplier (voir par exemple [BVI91]). Les géologues structuralistes qui réalisent ce dépliage souhaitent utiliser le résultat pour détecter des modèles 3D invalides du sous-sol, à savoir des modèles qui ne sont pas réalistes du point de vue géologique.

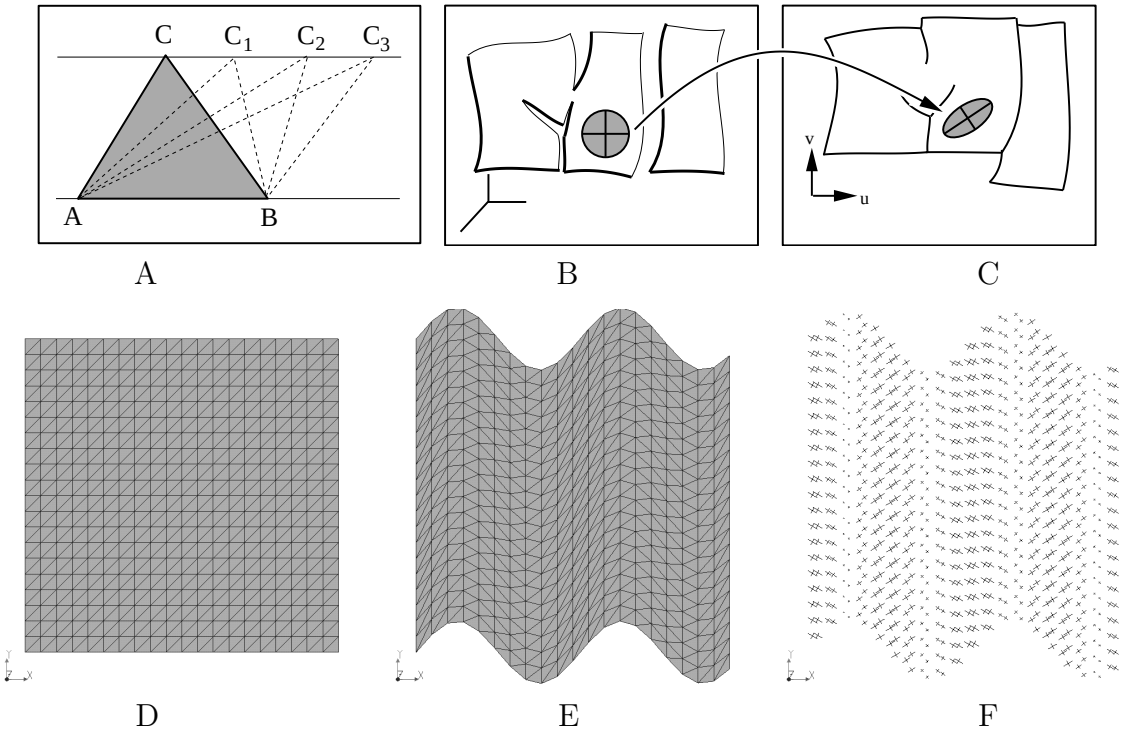


Figure 3.27: *Certaines transformations géométriques n'affectent pas les aires, bien que causant des déformations; A: ces quatre triangles ont la même aire; B: le type de déformation correspondant peut être détecté en analysant comment un cercle élémentaire est transformé en une ellipse à travers la transformation, en chaque point de la surface; les deux axes de cette ellipse sont donnés par le tenseur des déformations; D-E: un carré, déformé en 2D en préservant les aires; F: le tenseur des déformations permet de détecter et de visualiser les déformations.*

Par exemple, en utilisant les approches de type *Block Packing* [GGD91, GG93, RCea93], consistant à déplacer des blocs rigides, il est facile de détecter les zones présentant des incohérences en analysant le recouvrement ou les interstices entre les blocs. La méthode de Bennis *et. al.* [BVI91] va fournir le même type d'informations, en "déchirant" les horizons (représentés par des surfaces paramétriques) là ou leur courbure géodésique dépasse un certain seuil.

Notre méthode se différencie de ces deux types d'approches, car elle se place dans un contexte **continu**. Toutefois, il serait possible de définir un faible poids pour la contrainte de continuité à travers les failles, et d'analyser les recouvrements et interstices éventuels pour détecter les zones de forte tension. Nous pensons qu'il est préferable de caractériser le dépliage à l'aide de différentes fonctions d'énergie, permettant de cartographier la répartition des tensions sur la surface.

Comme cela a été mentionné dans [HG98], plusieurs fonctions d'énergie peuvent être utilisées pour caractériser une paramétrisation :

- L'énergie de *Dirichlet* $E_D(\Phi) = \frac{1}{2}\int |||[\frac{\partial\varphi}{x}\frac{\partial\varphi}{y}\frac{\partial\varphi}{z}]|||^2$, utilisée par Pinkall *et. al.* [PP93] et Eck *et. al.* [EDD+95].

- La norme du tenseur de déformation de *Green-Lagrange* $||G - Id||^2$, où G dénote le tenseur fondamental de la paramétrisation (voir Section 3.3) et où toute norme matricielle peut être utilisée. Ce critère a été utilisé par Maillet *et. al.* dans [MYV93], dans le cadre de la déformation et de l'animation d'objets texturés.

- Hormann *et. al.* proposent un autre critère dans [HG98], correspondant à l'altération de la forme des triangles, dont nous donnons ici une idée générale.

Intuitivement, la norme du tenseur de déformation de Green-Lagrange correspond au rapport de l'aire d'une surface élémentaire et de son image à travers la fonction de dépliage, ce qui permet de cartographier la répartition de ce type de déformations sur la surface. Toutefois, certaines transformations ne modifient pas les aires, tout en déformant les surfaces. Par exemple, sur la Figure 3.27, si nous déplaçons le sommet **C** le long d'une parallèle au côté [**AB**], l'aire du triangle (**A**, **B**, **C**) reste constante. Pour cette raison, cette information doit être complétée par un autre critère, permettant de détecter ces "cisaillements". Nous proposons de considérer de quelle manière un cercle élémentaire tracé sur la surface (Figure3.27-B) se transforme en une ellipse (Figure 3.27-C) à travers la paramétrisation inverse. Les directions et longueurs de ces axes s'obtiennent facilement, comme vecteurs propres et valeurs propres d'une matrice 2×2, appelé le *tenseur des déformations*. En affichant la répartition de ces axes sur la surface, il est possible de donner une idée des déformations sous-jacentes (Figure 3.27-D,E,F). C'est donc l'ensemble de ces deux informations qui va permettre à l'utilisateur de comprendre comment les aires et les formes sont localement affectées par le dépliage.

Pour tester notre méthode, nous avons utilisé deux modèles géologiques différents, présentés sur la Figure 3.28. La surface de gauche correspond à un horizon extrait du modèle EAEG-SEG, et présente une géométrie que les méthodes classiques ne pourraient pas prendre en compte, à cause des zones multi-Z-valuées. La surface de droite, fournie par ELF-GRC, a une géométrie plus simple, mais comporte un réseau de failles complexe. Notre méthode a permis de déplier ces deux surfaces, tout en respectant leur aire avec une erreur relative inférieure à 1/1000. La Figure 3.29 montre deux champs de déplacement correspondant au même dépliage, mais calculés relativement à deux origines différentes, symbolisées par des étoiles. Les deux champs ont un aspect totalement différent, bien que correspondant au même dépliage, et une bonne pratique est nécessaire pour être capable d'analyser l'évolution de la surface au niveau des failles. Pour

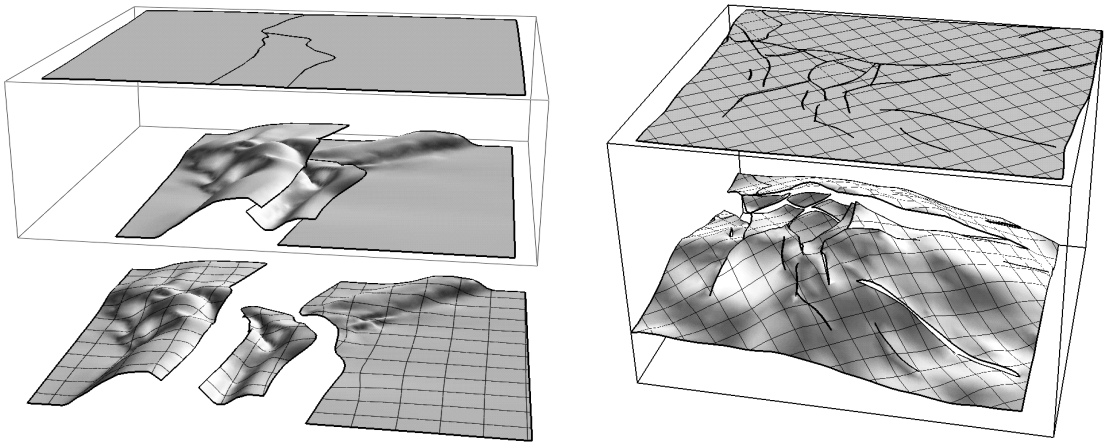


Figure 3.28: *À gauche: Dépliage d'un horizon (interface entre deux couches géologiques) du modèle EAEG-SEG; de haut en bas: horizon déplié, horizon original et vue éclatée de l'horizon avec les courbes iso-paramétriques associées; à droite: dépliage d'un horizon fortement faillé (modèle fourni par ELF-GRC); les courbes iso-paramétriques sont affichées sur les deux surfaces.*

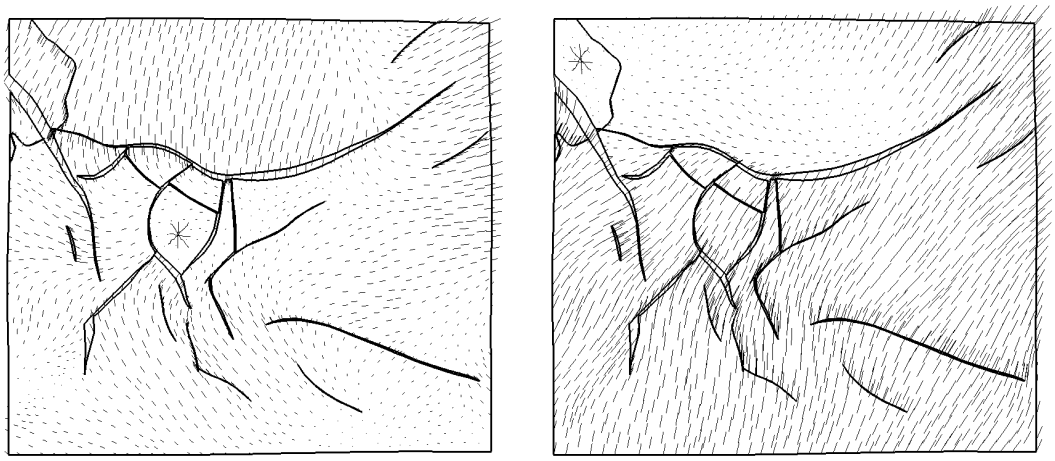


Figure 3.29: *Les géologues structuralistes sont habitués à caractériser un dépliage en termes de champ de déplacement, défini comme le champ des vecteurs connectant la surface originale à la surface dépliée. Un tel champ de déplacement est une propriété* **extrinsèque***, dépendante du choix d'une origine. Deux champs de déplacement correspondant au même dépliage sont montrés ici, l'origine étant symbolisée par une étoile.*

cette raison, nous pensons que la norme du tenseur de Green-Lagrange (correspondant à l'altération des aires) et le tenseur des déformations fournissent plus d'information, car ils caractérisent de manière intrinsèque la paramétrisation (voir Figure 3.30).

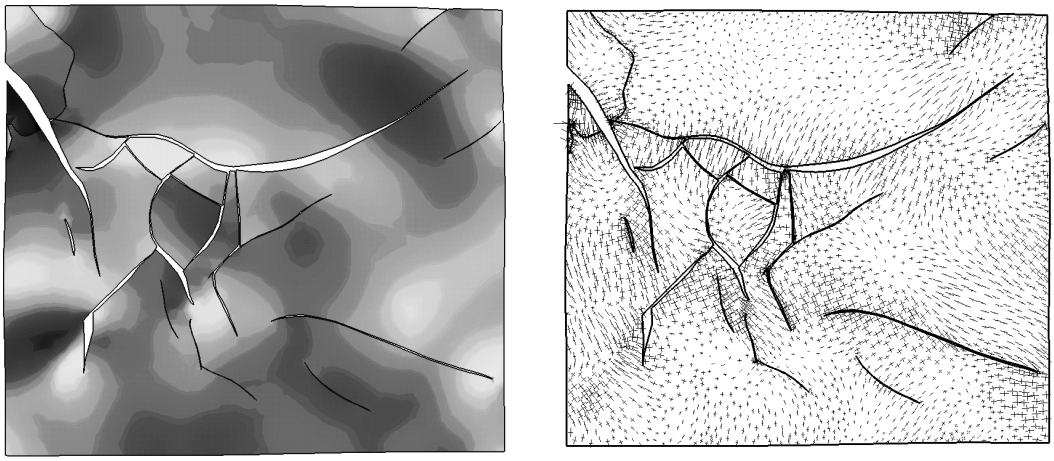


Figure 3.30: *À gauche: norme du tenseur de Green-Lagrange, caractérisant la manière dont les aires sont affectées par le dépliage; à droite: vecteurs et valeurs propres du tenseur des déformations; contrairement au champ de déplacement, ces propriétés sont définies de manière* **intrinsèque** *par rapport à une paramétrisation donnée, et ne nécessitent pas le choix d'une origine.*

Notre méthode de dépliage n'a pas encore été validée par des spécialistes de la géologie structurale, nous pensons qu'une analyse plus fine du comportement aux alentours des failles sera nécessaire pour permettre de l'utiliser comme outil de validation de modèles géologiques.

Remarque : Pour construire une paramétrisation sans distortions, Hormann et Greiner ont utilisé dans [HG98] la *norme de Frobenius* du tenseur des déformations, à savoir le rapport de ses valeurs propres (voir également [PP93] et [Bra92b, Bra92a]). Autrement dit, ce critère revient à dire qu'un cercle élémentaire doit être transformé en un autre cercle à travers la paramétrisation. Tous calculs effectués (voir [HG98]), ce critère $\mathcal{K}_\Phi(\mathbf{T})$ est constant sur chaque triangle $\mathbf{T}$:

$$\mathcal{K}_\Phi(\mathbf{T}) = \frac{\cot\alpha|a|^2 + \cot\beta|b|^2 + \cot\gamma|c|^2}{|\mathcal{D}_\mathbf{T}|} \tag{3.31}$$

où (a,b,c) dénotent les longueurs des trois côtés du triangle dans $\mathbb{R}^3$, α, β, γ correspondent aux trois angles aux sommets dans le domaine paramétrique (u, v) (avec la convention usuelle définissant γ comme le sommet opposé au côté a). Le scalaire $|\mathcal{D}_\mathbf{T}|$ correspond à l'aire de $\mathbf{T}$ dans le domaine paramétrique. Dans le Chapitre 2, nous avons vu comment approximer le Laplacien d'une fonction discrétisée aux sommets d'une surface triangulée (voir Section 2.3.1). Si nous effectuons un regroupement par triangles, nous obtenons à un facteur multiplicatif près la même formule pour le Laplacien discret et pour le critère de Hormann.

Ceci explique les propriétés des générateurs de maillage par lissage Laplacien [Han95], qui rendent ainsi chaque triangle le plus équilatéral possible.

3.6 Bilan et perspectives

Nous avons présenté dans cette partie un ensemble de nouvelles techniques pour la paramétrisation sans distortion des surfaces triangulées. Par rapport aux approches classiques, fondées sur une minimisation des distortions, notre méthode est plus souple et peut prendre en compte des sources diverses d'information. Ainsi, l'utilisateur peut spécifier les zones où les distortions doivent être minimisées par ordre de préférence. Il est également possible de faire passer un ensemble de courbes iso-paramétriques le long de courbes fournies par l'utilisateur, et aussi de rendre la paramétrisation continue à travers les déchirures de la surface. De plus, il est facile d'étendre la méthode en définissant de nouvelles contraintes, dès lors que ces contraintes peuvent être exprimées par des relations linéaires (ou linéarisables).

En rendant la formulation du problème la plus générale possible, nous nous sommes rendus compte que notre formalisme permettait d'exprimer les méthodes de paramétrisation usuelles, telles que les cartes harmoniques [EDD+95] ou la paramétrisation de Floater [Flo97] (voir Section 3.3.3). Des méthodes plus récentes, telles que celle décrite par Hormann dans [HG98], peuvent également être décrite par notre formalisme. Ceci permet d'enrichir de telles méthodes avec la possibilité de les combiner avec les autres méthodes existantes, et surtout de prendre en compte des contraintes linéaires.

La méthode peut être facilement implantée, puisqu'elle ne nécessite qu'une représentation efficace des surfaces triangulées, permettant un accès rapide aux satellites d'un sommet donné. La plupart des logiciels de C.A.O. proposent une telle représentation, ce qui rend la méthode facile à intégrer dans ce type de logiciels. Ainsi, cette méthode a été intégrée dans le noyau du logiciel Gocad, largement utilisé dans le domaine des géosciences. Dans ce cadre, un autre bénéfice d'une formalisation très générale du problème a été la possibilité d'appliquer la méthode à un large spectre de problèmes. Ainsi, plusieurs outils ont été développés, permettant le dépliage à aire constante des interfaces entre couches géologiques, ou encore facilitant certains calculs en les effectuant dans l'espace de textures (u, v) (voir Section 3.5).

La principale limitation de notre méthode est qu'elle ne s'applique qu'à des graphes planaires, c'est à dire à des surfaces triangulées topologiquement équivalentes à un sous-ensemble d'un disque. Ainsi, telle qu'elle est présentée ici, notre méthode ne peut pas s'appliquer à une surface fermée (par exemple une

sphère ou un tore). Nous pouvons imaginer facilement une généralisation de notre méthode à des surfaces de topologie arbitraire, qui consisterait à subdiviser de la surface à traiter en un ensemble de sous-surfaces, équivalentes à des disques topologiques. Dans [EDD+95], une telle approche est proposée, fondée sur la construction d'un diagramme de Voronoï curviligne inclus dans la surface. La méthode décrite dans [Tur91] pourrait également être utilisée pour choisir les sites du diagramme de Voronoï. Afin d'assurer une continuité du gradient d'un sous-domaine à l'autre, il serait envisageable d'utiliser une contrainte telle que celle décrite dans l'Équation 3.30, qui permettrait de gommer les limites des sous-domaines qui apparaissent clairement lorsque l'on applique la méthode proposée dans [EDD+95] (Lee *et. al.* proposent également dans [LSS+98] d'utiliser des fonctions de subdivision pour régler ce problème). Des méthodes d'analyse multi-résolution, comme celles décrites dans [EDD+95, KCVS98], permettent de décomposer la surface en éléments topologiquement équivalent à des disques. De plus, ce type de représentation devrait permettre d'appliquer des méthodes de résolution multi-grille [Hac85] à des maillages non-structurés, en définissant un certain nombre de niveaux intermédiaires, sous la forme de surfaces triangulées plus grossières. Ceci devrait permettre d'améliorer sensiblement la vitesse de convergence de l'algorithme D.S.I. itératif. Des premiers résultats encourageants ont été obtenus par O. Grosse dans le cadre de sa thèse [Gro01], qui a ainsi amélioré la vitesse de convergence de D.S.I. pour les grilles réguilières 3D.

Ces améliorations de la vitesse de convergence permettent d'envisager une généralisation de la méthode à la dimension supérieure, permettant de construire une paramétrisation pour des volumes tétraédrisés, comportant un très grand nombre d'éléments. Autrement dit, le problème consiste à assigner des coordonnées (u, v, w) aux sommets de la tétraédrisation. La méthode décrite pour les surfaces dans ce chapitre se fonde sur des relations entre les gradients des composantes de la paramétrisation, permettant de définir les critères à satisfaire (préservation des distances et des angles à travers la paramétrisation). Dans le cas 3D, il est également facile d'estimer le gradient d'une fonction φ, interpolée linéairement sur un tétraèdre. Les sommets α_i du tétraèdre ont pour coordonnées (x_i, y_i, z_i). Les trois composantes du gradient $[a, b, c]$ sont alors définies par l'équation de l'interpolation linéaire Φ de φ :

$$\begin{cases} \Phi(x,y,z) = a.x + b.y + c.z + d \\ \\ \begin{bmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \varphi(\alpha_0) \\ \varphi(\alpha_1) \\ \varphi(\alpha_2) \\ \varphi(\alpha_3) \end{bmatrix} \\ \\ \text{soit : } A.X = B \end{cases} \tag{3.32}$$

Nous pouvons alors trouver les composantes du gradient $[a, b, c]$ en résolvant un système linéaire à quatre inconnues $AX = B$. Comme dans le cas des surfaces triangulées, les composantes du gradient sont alors des combinaisons **linéaires** des valeurs de φ aux sommets du tétraèdre, dont les coefficients ne dépendent que de la géométrie du tétraèdre. Il est alors facile d'exprimer les contraintes de préservation des distances et des angles, comme nous l'avons vu Section 3.3.

Dans le Chapitre 1, nous avons vu comment représenter des maillages 3D composés de polyèdres arbitraires, et S. Conreaux est en train de définir différentes manières de construire de tels maillages [Cnr00]. Pour cette raison, il nous parait intéressant de ne pas nous limiter au cas simplicial pour ces problèmes de paramétrisation. Ainsi, dans le cas d'un polyèdre quelconque, la matrice A de l'Équation 3.32 peut avoir un nombre de lignes différent du nombre de colonnes. Si nous résolvons le système moindres carrés $A^t A.X = A^t B$, alors le gradient sera donné par :

$$\begin{bmatrix} \sum x_i^2 & \sum x_i.y_i & \sum x_i.z_i & \sum x_i \\ \sum x_i.y_i & \sum y_i^2 & \sum y_i.z_i & \sum y_i \\ \sum x_i.z_i & \sum y_i.z_i & \sum z_i^2 & \sum z_i \\ \sum x_i & \sum y_i & \sum z_i & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \sum x_i.\varphi(\alpha_i) \\ \sum y_i.\varphi(\alpha_i) \\ \sum z_i.\varphi(\alpha_i) \\ \sum \varphi(\alpha_i) \end{bmatrix} \tag{3.33}$$

En utilisant cette équation, les composantes du gradient "moindres carrés" sont également des combinaisons linéaires des valeurs de φ aux sommets du polyèdre, dont les coefficients ne dépendent que de la géométrie du polyèdre. Ceci devrait nous permettre de construire une paramétrisation pour de tels objets, ce qui pourrait s'appliquer à des problèmes de géostatistiques ou à d'autres problèmes numériques. Une équation similaire peut également être écrite pour des surfaces composées de polygones arbitraires. Nous pensons que les méthodes de déformation décrites comme une fonction de $\mathbb{R}^3$ dans lui-même, telles que par exemple FFD[9] [SP86, CJ91], Wires [SF98] ou DOGME [Bec92], pourraient être exprimées dans un contexte de minimisation d'une fonction objective (ce qui est déjà le cas de DOGME), et ainsi permettre la prise en compte de contraintes diverses. La thèse d'O. Grosse [Gro01] traite également de ce type d'approches, dans le cadre de la géologie numérique.

[9]Free-Form Deformation

Conclusion

Dans les trois chapitres de cette thèse, nous avons abordé trois problèmes de modélisation 3D qui se posent lorsque les objets sont représentés de manière discrète, à savoir :

1. Définir une structure de données combinatoire permettant de représenter la discrétisation des objets en primitives, ainsi que les connections entre ces primitives;

2. construire des surfaces ayant des propriétés géométriques utiles, et satisfaisant des contraintes définies par l'utilisateur;

3. construire une paramétrisation contrainte des objets, à savoir les mettre en correspondance avec un espace plus simple, où certains calculs et certaines opérations sont facilités.

Le premier point peut être résolu en utilisant des résultats de la topologie combinatoire, comme nous l'avons montré dans le premier chapitre. La structure hiérarchique que nous avons décrite permet une représentation efficace en termes d'espace et de complexité algorithmique. De plus, la possibilité de considérer une surface comme la séparation entre deux volumes nous a permis de considérer des surfaces non-variétés comme un ensemble de volumes variétés, et donc d'utiliser des structures de données plus simples. Ceci fournit un ensemble de fonctionalités compatibles avec le processus de modélisation en géologie numérique. Dans ce cadre, les utilisateurs construisent des surfaces avant de les relier ensemble pour délimiter des volumes.

D'un point de vue théorique, nous pensons que la structure de G-Cartes pourrait également servir de structure de données à une classe de problèmes liés à la géométrie algorithmique.

Les deux autres points concernent la *forme* et la *paramétrisation* des objets. Dans la plupart des cas, ce sont des problèmes "mal posés", n'ayant pas de solution unique. Afin de déterminer une solution unique, nous avons vu comment exprimer les propriétés souhaitées pour les objets sous la forme d'un critère global devant être optimisé. Dans les cas que nous avons étudiés, ce critère peut être

exprimé par une équation différentielle. Ainsi, dans le Chapitre 2, nous avons vu comment construire des surfaces ayant des formes lisses, en minimisant une approximation du Laplacien de la surface:

$$\mathcal{F}_{\Delta^2}(\varphi) \quad = \quad \sum_{\nu \in \{x,y,z\}} \int_{\mathcal{D}} \Delta\varphi^{\nu}(u,v)^2 . du dv$$

Dans le troisième Chapitre, une méthode permettant de construire une paramétrisation d'une surface a été introduite, en satisfaisant certains critères de non-distorsions. Ces critères correspondent au premier tenseur fondamental de la paramétrisation, qui caractérise la manière dont une fonction altère les distances et les angles:

$$\forall (u,v) \in \mathcal{D}, \quad G(u,v) = \begin{bmatrix} \frac{\partial \mathbf{x}}{\partial u}^2 & \frac{\partial \mathbf{x}}{\partial u}\frac{\partial \mathbf{x}}{\partial v} \\ \\ \frac{\partial \mathbf{x}}{\partial u}\frac{\partial \mathbf{x}}{\partial v} & \frac{\partial \mathbf{x}}{\partial v}^2 \end{bmatrix} \simeq \begin{bmatrix} 1 & 0 \\ \\ 0 & 1 \end{bmatrix}$$

Pour ces deux problèmes (lissage variationnel et paramétrisation contrainte), notre approche à consisté en trois étapes:

1. **discrétiser** les objets en cellules élémentaires, et représenter la combinatoire de cette discrétisation;
2. **approximer** l'équation différentielle sous forme d'une somme de carrés de combinaisons linéaires des valeurs aux sommets de la discrétisation;
3. **minimiser** ce critère à l'aide de l'algorithme D.S.I. itératif.

Ainsi, dans le deuxième chapitre, le Laplacien est approximé en un noeud de la surface par une combinaison linéaire des valeurs de la fonction aux voisins du noeud. Dans le troisième chapitre, les coefficients du premier tenseur fondamental sont exprimés en considérant une surface triangulée comme une fonction linéaire par morceaux. Ainsi, la méthode D.S.I. peut être considérée comme un solveur pour équations différentielles exprimées sur un support discrêt, les contraintes apparaissant alors comme des conditions aux limites. Notre approche peut également être assimilée à une méthode d'ajustement aux données. Comme nous l'avons vu dans la Section 2.5, le critère minimisé joue alors le rôle d'une matrice de régularisation, et l'utilisateur peut règler l'importance relative accordée à ce critère par rapport à celle accordée aux contraintes. Cette comparaison s'applique non seulement au lissage variationnel discret présenté dans le Chapitre 2, mais également à la paramétrisation contrainte que nous avons abordée dans le Chapitre 3.

Cette notion de paramétrisation est apparue comme un formalisme commun, permettant de traiter un certain nombre de problèmes posés en modélisation 3D. Comme nous l'avons montré dans la Section 3.5, en utilisant cette notion, il nous a été possible de mettre au point des outils pour :

- Texturer des surfaces tout en satisfaisant un ensemble de contraintes, les contraintes pouvant être éditées de manière interactive;
- construire des grilles structurées s'appuyant sur des surfaces triangulées;
- convertir des surfaces triangulées en surfaces polynomiales;
- re-trianguler des surfaces de manière optimale, suivant différents critères.

Une formulation générale de ce type de problèmes nous a permis d'exprimer des approches plus classiques dans notre formalisme, et de les combiner les unes avec les autres. Ainsi, des méthodes connues tels que les cartes harmoniques [EDD+95] et la paramétrisation de Floater [Flo97] apparaissent comme un cas particulier de notre approche. Des approches plus récentes, telles que celle de Hormann *et. al.* [HG98] peuvent également être exprimées en utilisant notre formalisme.

Comme nous l'avons déjà évoqué dans les sections *Bilan et perspectives* des trois chapitres, ces méthodes peuvent se généraliser à des dimensions supérieures, permettant de manipuler des maillages volumiques. Le grand nombre d'éléments qui constituent habituellement de tels maillages va nécessiter de mettre au point des techniques de multi-résolution et de multi-grille pour accélérer la convergence de l'algorithme D.S.I. itératif (voir Section 2.5). Les différents problèmes liés à l'extension en 3D des méthodes décrites ici vont donner lieu à plusieurs travaux de recherches :

- La construction et l'édition de maillages polyèdraux généraux seront abordées dans la thèse de Stéphane Conreaux [Cnr00] (voir Section 1.7);
- Mathieux Dazy va étudier dans le cadre de sa thèse une structure à la fois topologique et géométrique pour gérer de manière consistante les problèmes d'intersections (voir Section 1.7);
- nous étudierons également les problèmes liés à la *visualisation* de tels objets, à l'aide de techniques de rendu volumiques, généralisées à des cellules de formes arbitraires;
- l'approximation et la minimisation d'autres formes différentielles peuvent donner lieu à différentes extensions de la méthode. En considérant deux couronnes de voisins au lieu d'une seule, il nous semble possible de considérer des grandeurs géométriques d'ordre supérieur à celui du Laplacien. Par exemple, en minimisant certains coefficients du Hessien, caractérisant la courbure d'une surface, il serait possible d'obtenir des formes plus rondes (voir Section 2.5);

- une extension des méthodes de paramétrisations en 3D sera étudiée au cours de la thèse d'Olivier Grosse [Gro01] (voir Section 3.6). Ceci permettra de mettre au point des outils d'édition de type *déformations forme libre*;

Nous pensons que les généralisations des méthodes décrites ici, qui permettront de manipuler des maillages arbitraires en dimension supérieures, pourraient avoir un grand nombre d'applications possibles dans le domaine de la modélisation 3D. Par exemples, les méthodes de génération de grilles, ou encore la simplification de maillages volumiques pourraient bénéficier de ce type d'approches. En effet, la mise en correspondance de l'objet à modéliser avec un espace paramétrique (u, v, w), plus simple géométriquement que $\mathbb{R}^3$, devrait permettre de simplifier les algorithmes liés à la manipulation de ces objets volumiques.

Bibliographie

[AF86] S. Ansaldi and B. Falcidieno. Form Feature Representation and Recognition in a Hierarchical Boundary Model. In *Proc. of the IFIG WG 5.2 Working Conference*, may 1986.

[AFF85a] S. Ansaldi, L. De Floriani, and B. Falcidieno. An Edge-Face Relational Scheme for Boundary Representations. *Computer Graphics Forum*, 4:319–332, 1985.

[AFF85b] S. Ansaldi, L. De Floriani, and B. Falcidieno. Geometric Modeling of Solid Objects by Using a Face Adjacency Graph Representation. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 19. ACM, July 1985.

[Ago76] M. Agoston. *Algebraic Topology : A First Course - Pure and Applied Mathematics*. Marcel Dekker, 1976.

[Alf84] P. Alfed. A bivariate c^2 Clough-Tocher scheme. *Computer Aided Geometric Design*, 1:257–267, 1984.

[Alf87] P. Alfed. *A case study for multivariate piecewise polynomials*, pages 149–159. SIAM, 1987.

[Bar81] B.A. Barsky. *The β-Spline : A Local Representation Based on Shape Parameters and Fundamental Geoemtric Measures*. PhD thesis, University of Utah, 1981.

[Bar88] B.A. Barsky. *Computer Graphics and Geometric Modeling using β-Splines*. Springer Verlag, 1988.

[Bau75] B. Baumgart. A Polyhedron Representation for Computer Vision. In *AFIPS Nat. Conf. Proc.*, volume 44, pages 589–596, June 1975.

[BBB87] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[BBG74] R.E. Barnhill, G. Birkhoff, and Gordon. Smooth interpolation in triangles. *J. Approx. Theory*, 8:114–291, 1974.

[BBT97] D. Bechmann, Y. Bertrand, and S. Thery. N-dimensional gregory-bézier for multidimensional cellular complexes. In *Proceedings of Mathematical Methods for Curves and Surfaces IV*, 1997.

[BCR+97] O. Bourgeois, P.R. Cobbold, D. Rouby, J.C. Thomas, and V. Shein. Least-squares restoration of Tertiary thrust sheets in map view, Tajik depression, central Asia. *J. Geophys. Res.*, 102:27553–27573, 1997.

[BDFL93] Y. Bertrand, J.F. Dufourd, J. Françon, and P. Lienhardt. Algebraic Specification and Development in Geometric Modeling. In *Proc. of TAPSOFT'93*, April 1993.

[Bec92] Dominique S. Bechmann. Animation through space and time based on a space deformation model (DOGME), September 1992.

[Ber96] Y. Bertrand. Etude des performances en temps et en espace des 2-cartes et de leurs extensions. Technical Report 96/10, Université Louis Pasteur, LSIIT, URA CNRS 1871, 1996.

[Ber97] Y. Bertrand. Mémoire d'Habilitation à Diriger les Recherches: Modeleurs géométriques à base topologique: spécification, performances et applications, 1997.

[Bez70] P. Bezier. *Emploi des machines à commande numérique*. masson, 1970.

[Bez86] P. Bezier. *Courbes et Surfaces*. Hermès, 1986.

[BFH95] H. Bendels, D.W. Fellner, and S. Havemann. *Erweiterbare datenstrukturen zur modellierung und visualisierung polygonaler welten*, pages 149–157. D.W. Fellner, November 1995.

[BG96] Houman Borouchaki and Paul Louis George. Maillage de surfaces paramétriques. partie I: Aspects théoriques. Technical Report RR 2928, INRIA, Rocquencourt, France, July 1996. http://www.inria.fr/RRRT/RR-2928.html.

[BGH+95a] Houman Borouchaki, Paul Louis George, Frederic Hecht, Patrick Laug, Bijan Mohammadi, and Eric Saltel. Mailleur bidimensionnel de delaunay gouverné par une carte de métriques. partie I: Algorithmes. Technical Report RR 2741, INRIA, Rocquencourt, France, Dec. 1995. http://www.inria.fr/RRRT/RR-2741.html.

[BGH+95b] Houman Borouchaki, Paul Louis George, Frederic Hecht, Patrick Laug, Bijan Mohammadi, and Eric Saltel. Mailleur bidimensionnel de delaunay gouverné par une carte de métriques. partie II: Applications. Technical Report RR 2760, INRIA, Rocquencourt, France, Dec. 1995. http://www.inria.fr/RRRT/RR-2760.html.

[Blo85] J. Bloomenthal. Modeling the mighty maple. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 19, pages 305–311. ACM, July 1985.

[Boe88] W. Boehm. *Differential Geometry II - Curves and surfaces for CAGD*. 1988.

[Bra92a] K. Brakke. Minimal surfaces, corners and wires. *Journal of Geometric Analysis*, 2:11–36, 1992.

[Bra92b] K. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[Bri89] E. Brisson. Representing Geometric Structures in D Dimensions: Topology and Order. In *Symposium on Computational Geometry*, pages 218–227. ACM, June 1989.

[BS86] E. Bier and K. Sloan. Two-part texture mapping. *IEEE Computer Graphics and Applications*, pages 40–53, September 1986.

[BS95] J. Braun and M. Sambridge. A numerical method for solving partial derivative equations on highly irregular evolving grids. *Nature*, 376:655–660, 1995.

[BVI91] C. Bennis, J.M. Vézien, and G. Iglésias. Piecewise surface flattening for non-distorted texture mapping. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 25, pages 237–246. ACM, July 1991.

[BY95] J.D. Boissonat and M. Yvinec. *Géométrie Algorithmique*. Ediscience international, 1995.

[Car76] M.F. Do Carmo. *Differential geometry of curves and surfaces*. Prentice Hall, Englewood Cliffs, Inc., 1976.

[Cat74] E. Catmull. *A subdivision algorithm for computer display of curved surfaces*. PhD thesis, Dept. of Computer Sciences, University of Utah, December 1974.

[Caz97] D. Cazier. *Construction de systèmes de réécriture pour les opérations booléennes en modélisation géométrique*. Thèse de Doctorat, Université Louis Pasteur de Strasbourg - Département d'Informatique, 1997.

[CC78] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[CDHM95] M.J. Castro-Díaz, F. Hecht, and B. Mohammadi. New progress in anisotropic grid adaptation for inviscid and viscous flows simulations. In *4th Annual Intl. Meshing Roundtable*, Oct. 1995. http://www.ce.cmu.edu/ sowen/Roundtable.agenda.html.

[CEM97] R. Cognot, T. Aït Ettajer, and J.L. Mallet. Modeling Discontinuities on Faulted Geological Surfaces. In *SEG technical program*, pages 1711–1718, 1997.

[CG91] G. Celniker and D. Gossard. Deformable Curve and Surface Finite Elements for Free-Form Shape Design. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 257–265. ACM, 1991.

[CGA] CGAL. Computational Geometry Algorithms Library. In *http://www.cs.ruu.nl/CGAL/index.html*.

[CJ91] Sabine Coquillart and Pierre Jancéne. Animated free-form deformation: An interactive animation technique. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 23–26, July 1991.

[Cnr97] J. Conraud. *Génération de maillages de simplexes pour la modélisation d'objets naturels*. Thèse de Doctorat, INPL, 1997.

[Cnr00] S. Conreaux. *Modélisation de 3-Variétés - structures et algorithmes (titre provisoire)*. Thèse de Doctorat, INPL, 2000.

[Dah69] C. Dahlstrom. Balanced cross section. *Canadian Journal of Earth Sciences*, 6:743,757, 1969.

[Daz01] M. Dazy. *Modéles simpliciaux hiérarchiques et problèmes d'intersections (titre provisoire)*. Thèse de Doctorat, 2001.

[dB78] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[dB87] C. de Boor. *B-Form basics*, pages 131–148. SIAM, 1987.

[dC59] P.F. de Casteljau. Outillage méthodes calculs. Technical report, André Citroën Automobiles SA, 1959.

[Dee95] M. Deering. Geometric Compression. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 13–20. ACM, August 1995.

[DEG96] T.K. Dey, H. Edelsbrunner, and S. Guha. Computational Topology. In *Computational Geometry - Ten Years After (http://www.ics.uci.edu/ eppstein/gina/DeyEdelsbrunnerGuha.ps.Z)*, 1996.

[DKT98] T. DeRose, M. Kass, and T. Truong. Subdivision Surfaces in Character Animation. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 85–94. ACM, July 1998.

[DLG90] N. Dyn, D. Levin, and J.A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[DS78] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[Du88] W.H. Du. *Etude sur la représentation de surfaces complexes: Application à la reconstruction de surfaces échantillonées*. Thèse de Doctorat, Ecole Nationale Supérieure des Télécommunications, 1988.

[EDD+95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 173–182. ACM, August 1995.

[Edm60] J. Edmonds. A Combinatorial Representation for Polyhedral Surfaces. *Notices Amer. Math. Soc.*, 7, 1960.

[EH96] Matthias Eck and Hugues Hoppe. Automatic reconstruction of B-Spline surfaces of arbitrary topological type. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 325–334. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[EL92] H. Elter and P. Lienhardt. Extensions of the Notion of Map for the Representation of the Topology of Cellular Complexes. In *Proc. of 4^{th} Canadian Conference on Computational Geometry*, 1992.

[Evo] Evolver. (the surface evolver - version 2.11) k. a. brakke. In *http://www.susqu.edu/facstaff/b/brakke/evolver/evolver.html*.

[EW97] H. Edelsbrunner and R. Waupotitsch. A combinatorial approach to cartograms. *Journal of Computational Geometry*, 1997.

[Far86] G. Farin. Triangular Bernstein Bézier Patches. *Computer Aided Geometric Design*, 3:83–130, 1986.

[Far87] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*, pages 235–245. SIAM, 1987.

[Far91] G. Farin, editor. *NURBS for Curve and Surface Design*. SIAM, 1991.

[Far92] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design (3rd Edition)*. Academic Press, 1992.

[FB95] David Forsey and Richard H. Bartels. Surface fitting with hierarchical splines. *ACM Transactions on Graphics*, 14(2):134–161, April 1995.

[FF88] L. De Floriani and B. Falcidieno. A Hierarchical Boundary Model for Solid Object Representation. *ACM-Transactions on Graphics*, 7(1):42–60, January 1988.

[FG96] C. Fiorio and J. Gustedt. Two linear time union-find strategies for image processing. *Theoritical Computer Science*, 154:165–181, 1996.

[Fio95] C. Fiorio. *Approche interpixel en analyse d'images, une topologie et des algorithmes de segmentation*. Thèse de Doctorat, Université Montpellier II, 1995.

[FLN+90] Thomas A. Foley, David A. Lane, Gregory M. Nielson, Richard Franke, and Hans Hagen. Interpolation of scattered data on closed surfaces. *Computer Aided Geometric Design*, 7(1-4):303–312, June 1990.

[Flo97] M.S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, April 1997.

[FM79] Francis and Morin. Arnold Shapiro's eversion of the sphere. *Math. Intelligencer*, 2:200–203, 1979.

[FP95] L. De Floriani and E. Puppo. Hierarchical Triangulation for Multiresolution Surface Description. *ACM-Transactions on Graphics*, 14(4):363–411, October 1995.

[Fra87] G. K. Francis. *A Topological Picturebook*. Springer, 1987.

[Fuc97] L. Fuchs. *Une Spécification Formelle des Modèles de Courbes et de Surfaces de Forme Libre*. Thèse de Doctorat, Université Louis Pasteur /UPRES-A CNRS 7005, November 1997.

[FvFH90] Foley, vanDam, Feiner, and Hughes. *Computer Graphics, principles and practice*. Addisson Welsey, 1990.

[GG93] J.P. Gratier and B. Guillier. Compatibility constraints on folded and faulted strata and calculation of total displacement using computational restoration (unfold program). *Journal of Structural Geology*, 15(3-5):391–402, 1993.

[GGD91] J.P. Gratier, B. Guillier, and A. Delorme. Restoration and balance of a folded and faulted surface by best-fitting of finite elements: principles and applications. *Journal of Structural Geology*, 13(1):111–1115, 1991.

[GLH93] J.A. Gregory, V.K.H. Lau, and J.M. Hahn. *High order continuous polynomial patches*, pages 117–132. Springer Verlag, 1993.

[Gre80] J.A. Gregory. Triangular Interpolation Patch for CAGD. *Computer Graphics and Image Processing*, 13:80–87, 1980.

[Gre94] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13:143–154, 1994.

[Gre96] G. Greiner. *Curvature approximation with application to surface modeling*, pages 241–252. Teubner, 1996.

[Gro01] O. Grosse. *Mise en cohérence automatique d'un modèle géologique 3D légèrement perturbé*. Thèse de Doctorat, INPL, 2001.

[GS85] L. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoï Diagrams. *ACM-Transactions on Graphics*, 4(2):74–123, 1985.

[Gun93] C. Gunn. Discrete Groups and Visualization of Three Dimensional Manifolds. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, 1993.

[Hac85] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer Verlag, 1985.

[Ham] B. Hamann. *Curvature approximation for triangulated surfaces*, pages 139–153. Springer Verlag.

[Han95] Peter Hansbo. Generalized Laplacian smoothing of unstructured grids. *Comm. Numer. Meth. Eng.*, 11:455–464, 1995.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 71–78, July 1992.

[HDD+93] H. Hoppe, T. DeRose, T. DuChamp, J. McDonald, and W. Stuetzle. Mesh Optimization. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 19–26. ACM, August 1993.

[HDD+94] H. Hoppe, T. DeRose, T. DuChamp, M. Halstead, H. Jin, and J. McDonald. Piecewise Smooth Surface Reconstruction. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 28, pages 295–302, July 1994.

[Her87] G. Herron. *Techniques for visual continuity*, pages 163–176. SIAM, 1987.

[HG98] K. Hormann and G. Greiner. MIPS: An Efficient Global Parameterization Method. Technical Report 28/1998, University of Erlangen, Computer Graphics Group, 1998.

[HS97] S. Hardy and J. Suppe. Progress towards advanced restoration and forward modelling of structure in cross section. In *Geol. Soc. Am. Abs. with Prog.*, volume 29, page A44, 1997.

[Jen87] T. Jensen. *Assembling Triangular and rectangular patches and multivariate splines*, pages 203–220. SIAM, 1987.

[Kal93] M. Kallay. *Constrained optimization in surface design*. Springer-Verlag, 1993.

[KCVS98] L. Kobbelt, S. Campagna, J. Voratz, and H.P. Seidel. Interactive Multi-Resolution Modeling on Arbitrary Meshes. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 105–114. ACM, July 1998.

[Ket98] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proc 14th Annu. ACM Sympos. Comput. Geom.*, pages 146–154, 1998.

[KHD93] M. Kass, M. Halstead, and T. DeRose. Efficient, Fair Interpolation Using Catmull-Clark Surfaces. *Computer Graphics (SIGGRAPH Conf. Proc.)*, 27:35–44, 1993.

[KL96] V. Krishnamurthy and M. Levoy. Fitting Smooth Surfaces to Dense Polygon Meshes. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, August 1996.

[Kob96a] L. Kobbelt. A Variational Approach to Subdivision. *CAGD*, 13:743–761, 1996.

[Kob96b] L. Kobbelt. Interpolatory Subdivision on Open Quadrilateral Nets. In *Computer Graphics Forum*, 1996.

[Kob97] L. Kobbelt. Discrete Fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1997.

[Kob98] L. Kobbelt. *Variational Design with Parametric Meshes of Arbitrary Topology*. Teubner, 1998.

[Lan96] V. Lang. *Une étude de l'utilisation des ensembles simpliciaux en modélisation géométrique interactive*. Thèse de Doctorat, Université Louis Pasteur - Strasbourg, 1996.

[LC89] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 21, pages 163–169. ACM, 89.

[LDW97] M. Lounsbery, T. DeRose, and J. Warren. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics*, 16(1):34–73, January 1997.

[Lev95a] B. Levy. Ccontinuité G^1-G^2 des surfaces triangulées. *Mémoire de DEA*, 1995.

[Lev95b] B. Levy. Reconstruction de surfaces complexes - ajustement aux données. Technical Report 42518, Institut Franąis du Pétrole, 1995.

[Lie88] P. Lienhardt. Extension of the notion of map and subdivisions of a three-dimensional space. In *Proc. of* 5^th *Symposium on Theoritical Aspects in Computer Science*, february 1988.

[Lie94] P. Lienhardt. N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds. *Journal on Coumputational Geometry and Applications*, 4(3):275–324, 1994.

[LL95] V. Lang and P. Lienhardt. Geometric modeling with simplicial sets. Technical Report 95/06, 1995.

[LM98] B. Levy and J.L. Mallet. Non-Distorted Texture Mapping for Sheared Triangulated Meshes. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, July 1998.

[LMD92] M. Lounsbery, St. Mann, and T. DeRose. Parametric surface interpolation. *IEEE Comp. Graph. Appl.*, 12:45–52, 1992.

[LMM95] S. Levy, D. Maxwell, and T. Munzner. *Outside In (video).* The Geometry Center, University of Minnesota, 1995.

[LMTR94] M. Leger, J.M. Morvan, M. Thibaut, and H. Rakotoarisoa. *Minimization of geometrical criteria defined on a surface*, pages 110–135. World Scientific, 1994.

[Loo87] C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis. *Department of Mathematics, University of Utah*, August 1987.

[LP95] S. Levy and A.K. Peters. *Making Waves: a guide to the ideas behind outside in.* 1995.

[LS86] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting.* Academic Press, 1986.

[LSS+98] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 95–104. ACM, July 1998.

[LT95] S. Levy and W. Thurson. Evert (sphere eversion program, source code in C++). In *ftp://ftp.geom.umn.edu/pub/software/evert.tar.Z*, 1995.

[Mal89] J.L. Mallet. Discrete Smooth Interpolation in Geometric Modeling. *ACM-Transactions on Graphics*, 8(2):121–144, 1989.

[Mal92] J.L. Mallet. Discrete Smooth Interpolation. *Computer Aided Design Journal*, 24(4):263–270, 1992.

[Mal99] J.L. Mallet. *An Introduction to Geomodeling (preprint).* 1999.

[Män88] N. Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, 1988.

[Mas67] W.S. Massey. *Algebraic Topology, an Introduction.* Harcourt, Brace & World, Inc., 1967.

[Mau96] C.R.F. Maunder. *Algebraic Topology.* Dover, 1996.

[Max77] N. Max. Turning a sphere inside out (video). *International Film Bureau*, 1977.

[Mea82] D. Meagher. Geometric modeling using octree encoding. *Comput. Graph. Image Process.*, 19:129–147, 1982.

[ML88] S.D. Ma and H. Lin. Optimal texture mapping. In *EUROGRAPHICS'88*, pages 421–428, September 1988.

[Mor91] I. Moretti. Locace/Baliss un test de la cohérence géologique d'une interprétation sismique. *Revue de l'I.F.P.*, 46(5):563–580, 1991.

[MP77] Millman and Parker. *Elements of Differential Geometry*. Prentice-Hall, 1977.

[MP78] D.E. Muller and F.P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7:217–236, 1978.

[MS92] H. Moreton and C. Séquin. Functional Optimization for Fair Surface Design. *Computer Graphics (SIGGRAPH Conf. Proc.)*, 26:167–176, 1992.

[MS96] D.R. Musser and A. Saini. *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*. Addisson Wesley, 1996.

[MYV93] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 27. ACM, 1993.

[Nll98] S. Nullans. *Reconstruction géométrique de formes - Application à la géologie*. Thèse de Doctorat, Université de Nice-Sophia Antipolis, 1998.

[PBCF93] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-Independant Modeling with Simplicial Complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.

[PD85] Peachey and R. Darwyn. Solid texturing of complex surfaces. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 19, pages 287–296. ACM, July 1985.

[Ped95] H.K. Pedersen. Decorating implicit surfaces. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 291–300. ACM, 1995.

[Pet78] J.P. Petit. Retournement non trivial du tore. In *Comptes rendus de l'Academie des sciences de Paris*, 1978.

[Pet79] J.P. Petit. Retournement non trivial du tore. *Pour la Science*, Janvier 1979.

[Pet85a] J.P. Petit. *Le Geometricon - Les Aventures d'Anselme Lanturlu (bande dessinée)*. Belin, 1985.

[Pet85b] J.P. Petit. *Le Topologicon - Les Aventures d'Anselme Lanturlu (bande dessinée)*. Belin, 1985.

[Pet85c] J.P. Petit. *Le Trou Noir - Les Aventures d'Anselme Lanturlu (bande dessinée)*. Belin, 1985.

[PH97] J. Popović and H. Hoppe. Mesh Optimization. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, 1997.

[Phi66] A. Phillips. Turning a surface inside out. *Scientific American*, pages 112–120, 1966.

[Pip87] B.R. Piper. *Visually smooth interpolation with triangular Bézier patches*, pages 221–233. SIAM, 1987.

[PM94] Litwinowicz P and G. Miller. Efficient Techniques for Interactive Texture Placement. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 119–122. ACM, July 1994.

[PP93] U. Pinkall and K. Polthier. *Computing discrete minimal surfaces and their conjugates*, volume 2, pages 15–36. 1993.

[PR97a] J. Peters and U. Reif. Analysis of generalized B-spline subdivision algorithms. *SIAM Journal of Numerical Analysis*, 1997.

[PR97b] J. Peters and U. Reif. The Simplest Subdivision Scheme for Smoothing Polyhedra. *ACM Transactions on Graphics*, 16(4), October 1997.

[Prz95] K. Przemyskaw. Construction of g^1 continuous joins of rational bézier patches. *Computer Aided Geometric Design*, 12:283–303, 1995.

[PS95] Ron Pfeifle and Hans-Peter Seidel. Fitting triangular B-splines to functional scattered data. In Wayne A. Davis and Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 26–33. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1995. ISBN 0-9695338-4-5.

[QS90] E. Quak and L.L. Schumaker. Cubic spline fitting using data dependant triangulation. *Computer Aided Geometric Design*, 7:293–301, 1990.

[RCea93] D. Rouby, P.R. Cobbold, and et. al. Least-squares palinspastic restoration of regions of normal faulting = application to the Campos basin (Brazil). *Tectono-physics*, 221:439–452, 1993.

[Rei95] U. Reif. A Unified Approach to Subdivision Near Extraordinary Vertices. *Computer Aided Geometric Design*, 12:153–174, 1995.

[Req82] A.A.G. Requicha. Representation of Rigid Solids - Theory, Methods and Systems. *ACM Computing Surveys*, 12(4):437–464, 1982.

[RSBC96] D. Rouby, T. Souriot, J.P. Brun, and P.R. Cobbold. Displacements, strains and rotations within the Afar depression (Djibouti) from restoration in map view. *Tectonics*, 15:952–965, 1996.

[Sam96] P. Samson. *Équilibrage de structures géologiques 3D dans le cadre du projet Gocad*. Thèse de Doctorat, INPL, 1996.

[SBD86] F.J.M Schmitt, B.A. Barsky, and Wen-Hui Du. An Adaptive Subdivision Method for Surface Fitting from Sampled Data. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 179–188. ACM, 1986.

[Sch81] L.L. Schumaker. *Spline Function: Basic Theory*. Wiley, 1981.

[Sch93] W. Schwartz. *C^1 smoothing of multi-patch Bézier surfaces*. Springer Verlag, 1993.

[Sch96] J. E. Schweitzer. *Analysis and Applications of Subdivision Surfaces*. PhD thesis, University of Washington, 1996.

[SED91] D. Schultz-Ela and K. Duncan. Practical restoration of extensional cross sections. *Geobyte*, pages 14–23, December 1991.

[Seg98] D. Segonds. *Intégration de l'approche paramétrique dans le géomodeleur Gocad*. Thèse de Doctorat, INPL, 1998.

[Sei98] H. P. Seidel. 3D Geometry Compression with Splines. In *SIGGRAPH course notes: 3D Geometry Compression (course 21)*. ACM, July 1998.

[SF98] K. Singh and E. Fiume. Wires, A Geometric Deformation Technique. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 405–414. ACM, July 1998.

[SFL98] J. Sullivan, G. Francis, and S. Levy. The Optiverse (video). In *H.C. Hege and K. Polthier eds, VideoMath Festival at ICM'98*. Springer, 1998.

[Sib80] R. Sibson. A vector identity for the Dirichlet tesselation. In *Mathematicel Proceedings of the Cambridge Philosiphical Society*, volume 87, pages 151–155, 1980.

[Sib81] R. Sibson. *A brief description of natural neighbors interpolations*, pages 21–36. Wiley, 1981.

[SK97] L. Shirman and Y. Kamen. Fast and Accurate Texture Placement. *IEEE Computer Graphics and Applications*, pages 61–66, January 1997.

[SL95] A. Stepanov and M. Lee. The standard template library. In *http://www.cs.rpi.edu/ musser/doc.ps*, 1995.

[SM99] N. Sukumar and B. Moran. c^1 natural neighbors interpolant for partial derivative equations. *Numerical Methods for Partial Differential Equations*, 1999.

[Sma58] S. Smale. A classification of immersions of the two-sphere. *Trans. Amer. Math. Soc.*, 90, 1958.

[SMB98] N. Sukumar, B. Moran, and T. Belytschko. The natural element method in solid mechanics. *Intl. J. for Numerical Methods in Engineering*, 43(5):839–887, 1998.

[SML98] D. Segonds, J.L. Mallet, and B. Lévy. Smooth triangulated surfaces: g^1 continuity for ray-tracing. In *SEG'98 Technical Program*, 1998.

[SMSW86] Samek, Marcel, C. Slean, and H. Weghorst. Texture mapping and distortions in digital graphics. *The Visual Computer*, 2(5):313–320, September 1986.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, August 1986.

[Sta98] J. Stam. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 395–404. ACM, July 1998.

[Sto69] J.J. Stocker. *Differential Geometry.* Wiley, 1969.

[Str97] B. Stroustrup. *The C++ Programming Language, 3rd. ed.* Addison Wesley, 1997.

[Suk98] N. Sukumar. *The natural element method in solid mechanics.* PhD thesis, Northwestern University - Theoretical and Applied Mechanics, IL, U.S.A., 1998.

[SZL92] P. Schröder, J. Zarge, and W. Lorensen. Decimation of Triangle Meshes. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 65–70. ACM, August 1992.

[SZSS98] T.W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-Uniform Recursive Subdivision Surfaces. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 387–394. ACM, July 1998.

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[TGHL98] G. Taubin, A. Guéziec, W.P. Horn, and F. Lazarus. Progressive Forest Split Compression. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, 1998.

[Toc79] F. E. Tocher. The computer contouring of fabric diagrams. *Comput. Geosci.*, pages 73–126, 1979.

[Tra94] L. Traversoni. Natural neighbor finite element. In *Intl. Conf. on Hydraulic Engineering Software, Hydrosoft Proc.*, volume 2, pages 291–297. Computational Mechanics Publications, 1994.

[Tur91] G. Turk. Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 289–298. ACM, 1991.

[Tut60] W.T. Tutte. Convex representation of graphs. In *Proc. London Math. Soc.*, volume 10, 1960.

[Val92] Marie-Gabrielle Vallet. *Génération de Maillages Éléments Finis Anisotropes et Adaptatifs*. Thèse de Doctorat, L'Université de Paris VI, Sept. 1992. Abstract at http://www.inria.fr/RRRT/TU-0197.html.

[Ver96] Santosh Verma. *Flexible grids for reservoir simulation*. PhD thesis, Stanford University - School of Earth Sciences, 1996.

[VHM91] Marie-Gabrielle Vallet, Frederic Hecht, and B. Mantel. Anisotropic control of mesh generation based upon a Voronoi type method. In *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, 1991.

[Wat94] D.F. Watson. *nngridr: An implementation of natural neighbors interpolation*. 1994.

[Wei84] K. Weiler. Topology as a Framework for Solid Modeling. In *Proc. of Graphics Interface*, pages 53–56, may 1984.

[Wei85] K. Weiler. Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, 1985.

[Wei86] K. Weiler. The Radial Edge Structure: a Topological Representation for Non-Manifold Geometric Boundary Modeling. In *Proc. of the IFIG WG 5.2*, may 1986.

[WW92] W. Welch and A. Witkin. Variational Surface Modeling. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 157–166. ACM, 1992.

[WW94] W. Welch and A. Witkin. Free-Form Modeling Using Triangulated Surfaces. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 247–256. ACM, 1994.

[WW98] H. Weimer and J. Warren. Subdivision Schemes for Thin Plate Splines. In *EUROGRAPHICS'98*, 1998.

[Zor97] D. Zorin. *Stationary Subdivision and Multiresolution Surfaces Representations*. PhD thesis, Caltech, Pasadena, 1997.

[Zor98] D. Zorin. Subdivision Surfaces. In *SIGGRAPH course notes: Subdivision for Modeling and Animation (course 36)*, pages 45–81. ACM, July 1998.

[ZS98] D. Zorin and P. Schröder. Foundations I: Basic Ideas. In *SIGGRAPH course notes: Subdivision for Modeling and Animation (course 36)*, pages 17–44. ACM, July 1998.

[ZSS96] D. Zorin, P. Schröder, and W. Sweldens. Interpolating Subdivision for Meshes with Arbitrary Topology. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 189–192. ACM, 1996.

[ZSS97] D. Zorin, P. Schröder, and W. Sweldens. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, 1997.