N° attribué par la bibliothèque

Université d'Auvergne
École doctorale Sciences pour l'Ingénieur
Laboratoire LAIC

Thèse

présentée par

Thibault Marzais

et soutenue

le 04 décembre 2008

en vue de l'obtention du

Doctorat de l'Université d'Auvergne

Spécialité : Informatique

Titre :

Reconstruction de surfaces paramétrées par LP-Fitting

Directeurs de thèse : Rémy Malgouyres et Yan Gérard

Jury

M. Marc Daniel	Rapporteur
M. Pascal Lienhardt	Rapporteur
M. André Lieutier	Examinateur
M. Vincent Barra	Président
M. Yan Gérard	Directeur de thèse
M. Rémy Malgouyres	Directeur de thèse

N° attribué par la bibliothèque

Université d'Auvergne
École doctorale Sciences pour l'Ingénieur
Laboratoire LAIC

Thèse

présentée par

Thibault Marzais

et soutenue

le 04 décembre 2008

en vue de l'obtention du

Doctorat de l'Université d'Auvergne

Spécialité : Informatique

Titre :

Reconstruction de surfaces paramétrées par LP-Fitting

Directeurs de thèse : Rémy Malgouyres et Yan Gérard

Jury

M. Marc Daniel	Rapporteur
M. Pascal Lienhardt	Rapporteur
M. André Lieutier	Examinateur
M. Vincent Barra	Président
M. Yan Gérard	Directeur de thèse
M. Rémy Malgouyres	Directeur de thèse

Cette thèse a été effectuée au sein du LAIC, Laboratoire d'Algorithme et Image de Clermont-Ferrand, équipe d'accueil 2146.
L'adresse du laboratoire est la suivante :

LAIC, IUT departement Informatique
B.P.86
63172 AUBIERE cedex
FRANCE

Remerciements

Après de longs mois de travail et de rédaction, il convient de remercier tous ceux qui ont participé de près comme de loin à cet avènement.

Tout d'abord, je voudrais remercier les membres du Jury. Merci à Marc Daniel, Pascal Lienhardt, André Lieutier et Vincent Barra, d'avoir accepté de faire partie du jury et m'apporter les conseils et dernières modifications à cette thèse.

Ensuite, je souhaite remercier mes deux directeurs de thèse Yan Gerard et Remy Malgouyres, qui ont encadré mes travaux. Encore merci à Yan d'avoir passé de nombreuses heures à m'écouter, me corriger, me guider...

Merci également à tous les permanents du LAIC, du département Info et du département R&T : Jean-Pierre, Fabien, Malika, Alex, l'abbé Gaillard, Nadia, Jerzy, Marie-Alix, François, Guénal, Olivier, Jean-Paul, Marie-Françoise, Abdel, Mireille, Nathalie...

Je pense aussi aux autres doctorants, stagiaires et aux gens de passage en C4/C6 : Pierre, Rita, Fabien, Alexandre, Anastasia, Florent, Mustafa, Lukasz, Carine, Florent, Raja, Cédric, Frédéric, Olivier, Grégoire, Valentin, Philippe... Beaucoup de très bons moments avec vous les amis.

Comment ne pas mentionner ceux qui m'ont vu de plus loin : Anne, Olivier, Pedro, Olivier, Marianne, Jeremy, Nicolas, François... les longues discussions à refaire les maths, les parties de coinche, de 5000, et le nota bene !

Mes pensées sont aussi dirigées envers ma famille, et particulièrement Aude ma petite femme qui deviendra cet été ma femme.

Introduction

Dans les domaines de recherche actuels en vogue, on trouve l'informatique graphique. Même si ce terme est très vague, on sait qu'il est étroitement lié à l'informatique et à l'imagerie. Ce qu'un public non initié ne sait pas toujours, c'est qu'il y a deux type d'imagerie.

D'une part nous avons l'imagerie d'acquisition, celle qui est liée aux caméras et appareils photos numériques, ainsi que tout ce qui touche à cette imagerie d'acquisition, à savoir l'analyse d'une séquence d'images, l'extraction de données, la segmentation d'images, etc...

Il y a d'autre part la synthèse d'images, qui est une discipline un peu en marge car son objectif n'est pas l'acquisition ou l'étude d'images réelles mais la génération d'images virtuelles. Son principe repose le plus souvent sur la notion de scène, c'est à dire l'ensemble des éléments que l'on souhaite voir. La visualisation de ces éléments de nature géométrique est l'objet d'un ensemble de techniques de modélisation, d'illumination, de rendu qui visent le plus souvent à obtenir une image la plus réaliste possible. De nos jours, le matériel informatique est de plus en plus puissant, nous disposons de GPU (cartes graphiques) à grande mémoire, nous utilisons des API récentes telles qu'OpenGL 3 et DirectX 10. La principale application qui participe à cette effervescence est le jeu vidéo, guidée par la recherche du réalisme, le soin du détail, et la contrainte temps réel, le tout pour une résolution d'écran de plus en plus grande. Ces calculs restent pour autant très gourmands, et sont toujours étudiés afin de les optimiser.

Les GPU sont devenus tellement puissants qu'aujourd'hui nous sommes à l'aube d'un tournant majeur. De plus en plus de gens croient à la programmation par GPU, d'autant plus que par leurs architectures, ils se révèlent bien plus puissants dans certains types de calculs. Le langage CUDA développé par NVIDIA, et le langage Brook+ développé par ATI en sont la preuve. Certains projets de calculs distributifs adaptent leurs logiciel de calculs pour profiter de cette avancée.

À la limite entre ces deux domaines d'imagerie, il y a la modélisation. Le rôle de la modélisation est de décrire un objet ou une scène et de permettre un rendu de cet objet par l'ordinateur. Il existe différentes manières de représenter des objets. Ils peuvent être sous la forme d'un nuage de points, de maillages, ou de surfaces plus complexes. L'objet ou la scène peuvent être éventuellement issus de relevés d'acquisition, ou bien modélisés par un infographiste. De même, quand les ingénieurs font de la CAO (conception assistée par ordinateur), ils modélisent des objets qui seront usinés et deviendront par la suite des objets réels. Actuellement, pour concevoir une pièce mécanique, beaucoup se contentent de faire varier les points de contrôle qui régissent les surfaces paramétrées, sans avoir de réel contrôle sur l'objet. En effet, en règle générale, la position des points de contrôle ne permet pas de dire de manière efficace « la surface doit passer à cet endroit ». C'est là que

se situe mon travail de thèse. Il s'agit, à partir d'un relevé de points sur la surface d'un objet (réel ou non), ou bien à partir d'un maillage, de déduire une modélisation de cet objet en se basant sur des modèles plus complets et facilement représentables. Les surfaces paramétrées en font partie, et sont très utilisées en CAO. Cependant, tout n'est pas si simple. Passer de la représentation du nuage de points à la surface paramétrée ne se fait pas en une seule étape et nécessite de manipuler des objets et des méthodes très variés.

Ce problème de reverse engineering est appelé le plus souvent « problème de reconstruction de surfaces », et la donnée est le plus souvent un maillage. Son intérêt date du milieu des années 80 et la méthode que nous allons suivre pour le résoudre a été publiée en 1996 par M. Eck and H. Hoppe [EH96]. Son principe consiste à décomposer la résolution en trois grandes étapes :

- Quadrangulation du maillage, c'est à dire découper des morceaux sur la surface du maillage qui sont homéomorphes à des disques, et dont chaque morceau peut être identifié comme un quadrangle.
- Paramétrisation de chaque quadrangle. Cette étape consiste à définir un plongement de la partie du maillage vers un espace de dimension inférieure. Le résultat est une paramétrisation, donnée nécessaire à l'approximation qui suivra.
- Approximation par Least-Square fitting. Le fitting est un terme signifiant lissage ou bien approximation. Cette étape permet de déterminer les descripteurs de la surface à reconstruire, généralement par la méthode des moindres carrés.

La contribution de ce mémoire consiste à proposer une variante de la dernière étape en introduisant le LP-fitting (LP pour programmation linéaire) de façon à minimiser la norme uniforme de l'erreur au lieu de sa norme euclidienne. Nous résolvons par la suite le problème avec un solveur de programmes linéaires. Nous présenterons aussi une méthode originale de quadrangulation de maillage développée en collaboration avec Jean-Marie Favreau et Vincent Barra. Il faut aussi préciser qu'une partie importante de ce travail a consisté à éclaircir et compléter un certain nombre de points laissés dans l'ombre par ce que l'on connaît de la littérature sur le sujet.

Ce mémoire comporte deux parties. La première partie est consacrée aux différents outils et objets qui sont nécessaires à la compréhension, à la manipulation et qui seront utilisés tout au long de ce mémoire. Le chapitre 1 reprend les définitions des maillages, leur acquisition, et les techniques de génération de maillages à partir d'un nuage de points. Le second chapitre rassemble les définitions et les propriétés des surfaces paramétrées qui nous seront utiles pour la reconstruction. On y retrouve les contraintes de recollement qui vont être utilisées par la suite. Le chapitre 3 présente ce qu'est l'optimisation et notamment la programmation linéaire car cet outil est au cœur de notre algorithme de reconstruction. Le quatrième chapitre de cette partie présente la différence entre interpolation et fitting. La différence en

l'interpolation et le fitting réside dans l'attache aux données. L'interpolation va faire en sorte que la surface produite passe exactement par les données initiales, quitte à ne fournir aucune solution si c'est impossible, tandis que le fitting passera « au plus près ».

Après la présentation du contexte mathématique et informatique du sujet, la seconde partie est consacrée à la reconstruction de surfaces proprement dite. La décomposition du problème initiée par Eck-Hoppe est introduite dans le chapitre 5, tandis que les trois chapitres suivants sont chacun consacrés à l'une de ces étapes (quadrangulation-paramétrisation-fitting). On présente en particulier dans le chapitre 6 une méthode de quadrangulation originale et dans le chapitre 8 la variante du LP-fitting.

L'organisation de ce mémoire ne représente pas l'ordre chronologique de mon travail. Je dirais même qu'au contraire, c'est presque l'ordre inverse que j'ai suivi. En effet, même si l'objectif de la thèse était au départ de faire le processus entier de reconstruction, nous nous sommes vite heurté à la difficulté du problème considéré dans son ensemble. Naturellement, nous avons adhéré à la décomposition de la reconstruction en différentes étapes. Chacune d'entre elles met en oeuvre un raisonnement complexe que j'ai essayé de décrire aussi précisément que possible dans le chapitre correspondant. Ce travail a été guidé par l'idée de développer le LP-fitting, c'est-à-dire minimiser la norme uniforme de l'erreur au lieu de sa norme euclidienne. Nous nous sommes d'abord intéressés à la reconstruction d'une seule surface paramétrée (Bézier, B-Spline ou définie par l'utilisateur). Par la suite, nous avons tenté de reconstruire des modèles plus complexes, en définissant des cartes de recollements. La difficulté du recollement de surfaces B-splines nous a conduit à travailler avec des surfaces de Bézier. Nous avons par ailleurs introduit deux options pour les coins : le recollement régulier en étoile equiangulaire ou le recollement choisi, selon une carte de recollement donnant les angles formés par les différentes arêtes. Enfin, dans le cadre d'une collaboration entre les différents laboratoires de recherche de Clermont-Ferrand, nous avons entrepris de mettre bout à bout mes travaux et ceux de Jean-Marie Favreau, qui travaillait sur le dépliage de surfaces. Nous avons défini ensemble un cadre formel pour définir la notion de pavage d'une surface (sans jonction en T) et nous l'avons utilisé pour développer un algorithme de quadrangulation de maillages.

Table des matières

Remerciements **5**

Introduction **6**

Table des matières **9**

Table des figures **13**

Liste des tableaux **17**

I Outils et objets **19**

1 Maillages **21**
1.1 Acquisition 21
1.2 Marching cubes 23
1.3 Triangulations 24
1.3.1 Complexe simplicial 24
1.3.2 Définitions 25
1.3.3 Techniques de triangulations 25
1.4 Homologie et géométrie discrète différentielle 27

2 Surfaces paramétrées **29**
2.1 Surfaces de Bézier 30
2.1.1 Dérivée d'une surface de Bézier 31
2.1.2 Élevation du degré 31
2.2 Surfaces B-Splines 32
2.3 Recollement de patches 33
2.3.1 Recollement de deux surfaces de Bézier 34
2.3.1.1 Recollement C^0 34
2.3.1.2 Recollement C^k 35
2.3.1.3 Recollement G^1 38
2.3.2 Recollement de plusieurs surfaces de Bézier 42
2.3.2.1 Échec partiel du recollement C^1 42

2.3.2.2 Cas particulier de recollement C^1 qui fonctionne . . . 44
2.3.2.3 Recollement G^1 de plusieurs patches . . . 45

3 Optimisation 49
3.1 Optimisation sans contraintes . . . 50
3.2 Optimisation sous contraintes . . . 51
3.2.1 Contraintes égalité . . . 51
3.2.2 Contraintes inégalité . . . 51
3.2.3 Méthodes de pénalité . . . 52
3.2.4 Programmation linéaire . . . 52

4 Interpolation et Fitting 55
4.1 Principe . . . 55
4.1.1 Centre, médiane et moyenne . . . 55
4.1.1.1 La moyenne arithmétique . . . 56
4.1.1.2 La moyenne arithmétique pondérée . . . 56
4.1.1.3 La moyenne géométrique . . . 56
4.1.1.4 La médiane . . . 56
4.1.1.5 Le centre . . . 57
4.1.2 Interpolation polynomiale . . . 57
4.1.3 Régression linéaire . . . 57
4.2 Fitting : position du problème . . . 58
4.2.1 Terme d'erreur . . . 58
4.2.2 Reconstruction par moindres carrés (Least Square Fitting) . . . 59
4.2.3 Reconstruction par programmation linéaire (LP Fitting) 59

II Reconstruction de surfaces 63

5 Principe de la reconstruction de surfaces 65
5.1 Introduction . . . 65
5.2 Reconstruction de surfaces paramétrées . . . 66

6 Segmentation et quadrangulations 69
6.1 Segmentation de maillages . . . 69
6.2 Découpage d'un maillage en une surface homéomorphe à un disque . . . 70
6.3 Théorie pour le pavage de surface . . . 71
6.3.1 Cellules et pavages . . . 72
6.3.1.1 Maillage initial . . . 72
6.3.1.2 Cadre topologique : notion de 2-m-cellule . . . 72
6.3.1.3 Tuiles et pavages . . . 73

6.3.1.4 Pourquoi un tel cadre ? 75
6.3.2 Pavage avec une unique tuile 76
6.3.2.1 Découpage et pavage 76
6.3.2.2 Calculer un découpage avec une longueur minimale 77
6.3.2.3 Trouver un 1-lacet non séparant 79
6.3.2.4 Découper une M-tuile 79
6.3.2.5 Contrôler le nombre d'arêtes de la M-tuile . 80
6.3.2.6 Nombre minimal d'arêtes 80
6.3.2.7 Méthode de découpage intermédiaire 83
6.3.3 Pavage avec des tuiles multiples 83
6.4 Quadrangulation 84
6.4.1 Définition et algorithme 84
6.4.2 Décollage de bords 86
6.4.3 Pavage avec un nombre minimal de quadrangles . . . 88
6.5 Autres techniques de quadrangulation 89

7 Paramétrisation **91**
7.1 Propriétés des techniques de paramétrisation 91
7.2 Différentes catégories de paramétrisation 92
7.2.1 Paramétrisation à bords libres 92
7.2.2 Paramétrisation à bords fixes 93
7.3 Paramétrisation des quadrangles segmentés 95

8 Fitting et Reconstruction **97**
8.1 Reconstruction d'une surface paramétrée 97
8.1.1 Position du problème 97
8.1.2 Conclusions 98
8.2 Reconstruction globale 99
8.2.1 Déduction des contraintes de recollement 100
8.2.1.1 Graphe de recollement 100
8.2.1.2 Choix de la qualité des surfaces paramétrées 101
8.2.1.3 Type de recollement 101
8.2.1.4 Extraction des contraintes de recollement d'un M-pavage 101
8.2.2 Reconstruction de plusieurs surfaces paramétrées par LP-Fitting 102
8.2.2.1 Reconstruction globale sans recollement . . . 102
8.2.2.2 Contraintes de recollement 102
8.2.2.3 Quelle fonction objectif pour le recollement ? 102

9 Résultats expérimentaux **105**
9.1 Quadrangulation 105
9.2 Reconstruction d'une surface paramétrée 107
9.2.1 Protocole de test 107
9.2.2 Interprétation des résultats 108
9.2.3 Histogrammes 109
9.2.4 Discussion 112
9.3 Reconstruction globale 112
9.3.1 Reconstruction d'une sphère 112
9.3.1.1 Surfaces bicubiques 113
9.3.1.2 Surfaces de degrés supérieurs 118
9.3.2 Reconstruction de modèles complexes 122
9.3.2.1 Reconstruction avec un nombre minimal de quadrangles 122
9.3.2.2 Reconstruction avec l'algorithme simple de quadrangulation 122
9.3.2.3 Reconstruction avec l'algorithme intermédiaire de quadrangulation 127
9.3.2.4 Discussion 130
9.3.3 Difficultés de résolution 130
9.3.4 Améliorations possibles 130

Conclusion et Perspectives **132**

Bibliographie **134**

Résumé **144**

Abstract **144**

Table des figures

1.1 Classification des méthodes d'acquisition, figure tirée de la thèse de Brian Lee Curless 22
1.2 Les 15 configurations de l'algorithme des marching cubes . . 23
1.3 Les n-simplexes 24
1.4 Triangulation de Delaunay 26
1.5 Exemple de diagramme de Voronoï 26
1.6 Exemples d'alpha shape. 27
1.7 Un complexe simplicial orienté C 27

2.1 Exemple de surface de Bézier 31
2.2 Illustration des fonctions $N_{i,4}$, vecteurs nodaux $S = T = \{0, 0, 0, 0, 1, 1, 1, 1\}$ 33
2.3 Position des points de contrôle dans le cas $n = 4$ 40
2.4 Cas de figure de recollement de plusieurs patches ayant un sommet en commun 42
2.5 Cas de figure de recollement de trois patches ayant un somment en commun, le recollement C^1 deux à deux n'est pas possible 42
2.6 Jonction fermée de n patches 43
2.7 Cas particulier de recollement C^1 de plusieurs patches où les conditions C^1 deux à deux entre les patches suffisent 45
2.8 Choix d'une configuration de recollement. Détermination des paramètres α_i et β_i 47

3.1 Représentation graphique de la fonction (3.2) 50

4.1 Il n'existe pas de solution X telle que $A \times X = B$ 58
4.2 Erreur de reconstruction introduite par un vecteur X 59
4.3 Least square fitting 59

5.1 Les différents niveaux de reconstruction 65

6.1 Schéma polygonal canonique : A gauche, le tore découpé, à droite, le polygone plan le représentant. 71

6.2 Quelques exemples de 2-m-cellules. Leurs bords sont décrits par une structure de 1-complexes. 73
6.3 Une M-tuile composée de la 2-m-cellule 6.2(c) et d'un plongement F sur un 2-mesh de genre 2. 74
6.4 Un M-pavage $(C, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$ (figure 6.3) par un complexe m-cellulaire ayant trois 2-m-cellules 75
6.5 Un pantalon et le M-pavage correspondant avec trois cylindres. 76
6.6 Une M-tuile $(P, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$. P est le polygone construit lors de la preuve du lemme 4. La M-tuile $(P', \mathcal{M}, F')$ est obtenue en supprimant les sommets de P dont l'image prend part à exactement deux arêtes de $\Gamma \cup B$. 77
6.7 Une M-tuile $(C, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$. C est une 2-m-cellule avec deux 1-m-cellules b_1 et b_2. k est un lacet trivial séparant, l est un 1-chemin non valide, m est un 1-chemin valide entre les bords b_1 etb_2, n est un 1-lacet valide non trivial séparant, et o un 1-lacet non séparant. 78
6.8 Les deux étapes de calcul d'un 1-lacet non trivial non séparant. 79
6.9 Découpage de la M-tuile selon le chemin rouge. À gauche : la M-tuile (en gris) avec quatre arêtes, quatre sommets (v_a, v_b, v_c et v_d). À droite : le résultat du découpage de la M-tuile avec les sommets v_a, v_b, et les sommets dupliqués (v'_c, v''_c) et (v'_d, v''_d). 80
6.10 Diagramme du procédé itératif. $(C_0, \mathcal{M}, F_0)$ est la M-tuile initiale. $(C_i, \mathcal{M}, F_i)$ est la M-tuile à la $i^{\text{ème}}$ itération de l'algorithme 2, et π_i le mapping surjectif du découpage de la M-tuile. 81
6.11 Deux exemples de pavage du tore. Le polygone de la tuile de la figure de gauche a 4 côté, alors que celui de droite en a 6. . 81
6.12 Étapes formelles de réduction du nombre d'arêtes d'une tuile. 82
6.13 Le double tore découpé, déplié avec la méthode 7.2.1. En bleu, les chemins de découpe. En violet, les multipoints. 85
6.14 Découpage d'un polygone pair en deux. Le nombre de côtés est indiqué dans le polygone. 86
6.15 Exemple de jonction en T. 86
6.16 Décollage de bords : le chemin b-e découpe la surface sans longer le bord. 87
6.17 Procédure de décollage du bord. 87

7.1 Double tore déplié avec le circle packing. Schéma polygonal canonique associé. Coordonnées de texture. 93
7.2 Configuration d'une arête à l'intérieur du maillage initial . . . 94
7.3 Paramétrisation d'un tube 96

8.1 Les quatres coins du patch. Les coordonnées paramétriques des points à reconstruire. 99

8.2 Six patches et un graphe de recollement correspondant. . . . 100
8.3 Graphe de recollement d'un cylindre (à gauche) et d'un ruban de Möbius (à droite). 100

9.1 Découpages à une seule tuile d'un 2-mesh de genre 2 sans bords. (830 points, 1664 triangles). 105
9.2 Différentes quadrangulations d'un 2-mesh de genre 2 sans bords (830 points, 1664 triangles). 106
9.3 Un quadrangle, issu d'un objet en forme de bretzel, résultant de l'algorithme 1. Ce quadrangle présente deux coins très proches. Cette configuration est non souhaitable lors d'une future paramétrisation ou reconstruction 107
9.4 Reconstruction du nuage de points rouge par une surface paramétrée. Vecteurs représentant l'erreur de reconstruction. . . 108
9.5 Reconstruction d'une sphère avec Least Squares Fitting (LSQF) 109
9.6 Reconstruction d'une sphère avec LP Fitting (LPF) 109
9.7 Histogrames d'une sphère perturbée et reconstruite par une surface de Bézier 5*5 : LP-fitting (en haut) least squares fitting (en bas) 110
9.8 Histogrames d'une autre sphère perturbée et reconstruite par une surface de Bézier 5*5 : LP-fitting (en haut) least squares fitting (en bas) 110
9.9 Boite englobante autour du point M. La surface reconstruite par LP-Fitting passera au travers de la boite. 111
9.10 Deux histogrammes de reconstruction : LP-fitting (en haut) least squares fitting (en bas) 111
9.11 Deux histogrammes de reconstruction de points très fortement perturbés : LP-fitting (en haut) least squares fitting (en bas) 111
9.12 Graphe de recollement de la sphère découpée en six morceaux 113
9.13 Reconstruction de la sphère avec comme fonction objectif le maximum des h_i 114
9.14 Reconstruction de la sphère avec comme fonction objectif la somme des h_i 115
9.15 Trous laissé apparents par la reconstruction sans recollement. 116
9.16 Différences de régularité entre la reconstruction C^0 et la reconstruction G^1. 116
9.17 Reconstruction de la sphère avec comme fonction objectif le maximum des h_i 118
9.18 Reconstruction de la sphère avec comme fonction objectif la somme des h_i 119
9.19 Irrégularités dûes à la reconstruction C^1. 120

9.20 Reconstruction sans recollement du double tore quadrangulé à l'aide de la méthode 1 avec des surfaces de Bézier de degrés 5×5 . La fonction objectif est la somme des h_i 122
9.21 Reconstruction du double tore (6 surfaces de Bézier 5×5) avec comme fonction objectif la somme des h_i 123
9.22 Reconstruction du double tore (6 surfaces de Bézier 5×5) avec comme fonction objectif la somme des h_i (vue approchée) 124
9.23 Reconstruction G^1 du double tore (6 surfaces de Bézier 5×5) avec comme fonction objectif la somme des h_i (vue approchée) 125
9.24 Reconstruction du double tore (4 surfaces de Bézier 5×5) avec comme fonction objectif le maximum des h_i 127
9.25 Reconstruction du double tore (4 surfaces de Bézier 5×5) avec comme fonction objectif le maximum des h_i. Vue de près d'un sommet 128
9.26 Reconstruction du double tore (4 surfaces de Bézier 7×7) avec comme fonction objectif la somme des h_i. Le degré est trop fort, il y a des oscillations. 129

Liste des tableaux

9.1 Résultats de la reconstruction sans recollement d'une sphère avec des surfaces bicubiques. À gauche, fonction objectif : maximum des h_i. À droite, fonction objectif : somme des h_i . 117
9.2 Résultats de la reconstruction G^1 d'une sphère avec des surfaces de degré 6×6. À gauche, fonction objectif : maximum des h_i. À droite, fonction objectif : somme des h_i 121
9.3 Résultats de la reconstruction G^1 du double tore avec des surfaces de degré 5×5. À gauche, fonction objectif : somme des h_i. À droite, fonction objectif : maximum des h_i 126

Première partie

Outils et objets

Chapitre 1

Maillages

La représentation d'objets à l'aide de maillages a fait l'objet de beaucoup d'investigations ces dernières décennies. Les maillages sont une représentation discrète d'objets de l'espace et sont très largement utilisés en modélisation géométrique et en informatique graphique (jeux vidéo, animation, etc...). Un tel engouement pour les maillages s'explique par la simplicité de leur représentation. Un maillage est composé d'un ensemble de points de l'espace, et d'un ensemble de facettes à formes géométriques diverses s'appuyant sur ces points. Cette représentation permet de modéliser des objets [Del94, HDD+93], de les stocker [TG98], de les simplifier [CMS98], de faire de la déformation (morphing) [Ale02], etc...

Dans le cadre de la reconstruction de surfaces, il est important de connaître l'origine de ces maillages. Commençons par nous pencher sur les méthodes d'acquisition, afin de savoir d'où proviennent les données étudiées, avant d'aborder les techniques de génération de maillages.

1.1 Acquisition

Il existe de nombreux dispositifs permettant de scanner les objets et d'en retirer des maillages ou tout simplement des nuages de points. Ces dispositifs sont divers et variés, et permettent d'obtenir un nuage de points à la surface d'un objet. Une classification des outils d'acquisition a été faite dans la thèse de Brian Lee Curless [Cur97]. L'auteur y décrit plusieurs méthodes d'acquisition (figure 1.1). On y retrouve les acquisition avec ou sans contact, réflectives non optiques comme le sonar, réflectives optiques telles que la triangulation spatiale, etc...

Le recueil de données le plus connu dans la communauté de modélisation géométrie est certainement le Stanford 3D Scanning Repository [sta08] et son fameux Stanford bunny. L'Universtité de Stanford a obtenu ces maillages en utilisant un scanner 3D de la société Cyberware. Ce scanner fait le tour de l'objet et prend des images de profondeur tout autour de l'objet. Ces

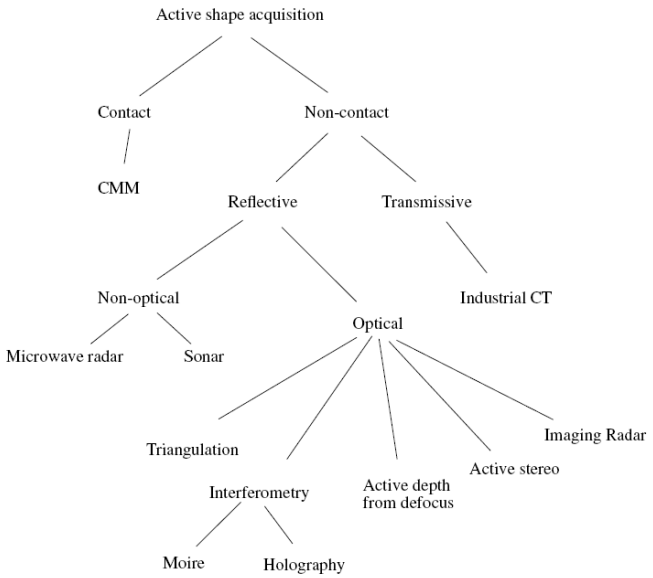


FIGURE 1.1 – Classification des méthodes d'acquisition, figure tirée de la thèse de Brian Lee Curless

images sont recoupées et alignées afin d'être mises dans le même repère, afin d'obtenir un maillage de l'objet scanné. Le procédé de reconstruction est décrit dans [TL94].

Dans l'ère du numérique que nous traversons actuellement, même les appareils photos numériques standards font partie des outils d'acquisition de nuages de points 3D [AKTL06]. Ainsi, comme le signalent les auteurs, il est possible aujourd'hui d'obtenir des données à l'aide d'appareils simples, vendu dans le commerce.

Comme dans tout processus de modélisation du réel, la fiabilité des données obtenues n'est pas parfaite. Il y a une phase de calculs entre les données d'acquisition, que sont les données brutes, et les données obtenues, que sont le nuage de points 3D et un éventuel maillage. On doit donc garder à l'esprit qu'il peut y avoir des erreurs d'imprécision à chaque étape du processus. En effet, les capteurs, aussi précis soient-ils, ne sont pas exempts d'erreurs. De même, les algorithmes d'alignement tiennent compte du fait que l'alignement peut ne pas être parfait, et peuvent donc introduire des erreurs inévitables sur la localisation des points 3D. Enfin, puisque tout le processus se fait par ordinateur, nous devons prendre en compte la limitation des calculs numérique à base de nombres flottants.

1.2 Marching cubes

Certains types de scanner permettent de scanner des objets et d'en donner une représentation voxellisée. L'espace est divisé en petits cubes. A l'instar du pixel dans une image noir & blanc, ces voxels ont une valeur binaire pour déterminer si oui ou non l'objet se trouve dans cette partie de l'espace. La représentation est correcte, mais lorsqu'on souhaite visualiser ce genre de données, on se retrouve confronté à plusieurs problèmes :

- La subdivision de l'espace peut être trop grossière. La résolution n'est pas un paramètre, et est souvent fixée par l'appareil d'acquisition.
- Conséquence indirecte de la subdivision, c'est l'éclairage de l'objet qui sera médiocre. En effet, si on choisit la représentation graphique des petits cubes, on se trouve confronté à au plus six normales aux faces.

C'est dans cette optique que la technique des marching cubes [LC87] a fait son apparition. Elle consiste à définir la surface de l'objet et produire une triangulation (partie 1.3) à partir de la scène voxellisée. Les triangles vont donc servir de frontière de l'objet, séparant ainsi les parties pleines (l'objet) et les parties creuses (le vide). Les voxels sont désormais considérés comme des vertex de $\mathbb{Z}^3$ disposé dans l'espace. En partant d'un groupe de huit vertex (formant les huit sommets d'un cube), les auteurs produisent un ensemble de triangles correspondant à la séparation des points du cube. Avec les différentes symétries et rotations, seules les 15 configurations de la figure 1.2 ont été retenues. La seconde partie de l'algorithme consiste à produire des normales pour chacun des vertices.

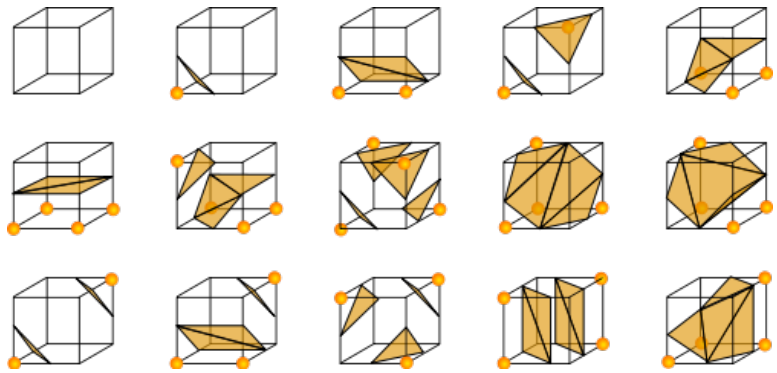


FIGURE 1.2 – Les 15 configurations de l'algorithme des marching cubes

La jonction de tous ces triangles forme une triangulation globale de l'objet. Cependant, il existe des cas où la jonction de deux configurations le long d'une face du cube se passe mal. Ces problèmes d'ambigüité sont traités dans l'algorithme de génération des triangles. Il s'agit de définir d'autres triangulations possible, afin de représenter la structure de l'objet et de respecter les jonctions [NH91].

1.3 Triangulations

Les types de maillages les plus utilisés sont les maillages triangulaires (appelés triangulation) et les maillages quadrangulaires (appelés quad-mesh).

Les maillages triangulaires entrent dans le cadre d'une structure mathématique appellé complexes simpliciaux.

1.3.1 Complexe simplicial

Soit n un entier naturel. Un n-simplexe est l'enveloppe convexe de $n+1$ points formant un repère affine de dimension n dans un espace de dimension au moins n. On retrouve une représentation de n-simplexes figure 1.3.

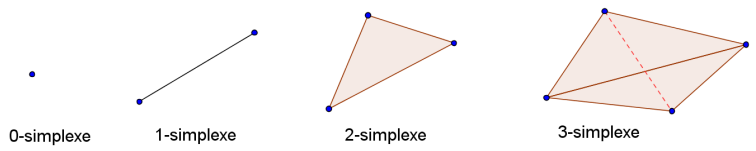


FIGURE 1.3 – Les n-simplexes

Un complexe simplicial est un ensemble de simplexes C qui doit vérifier les contraintes suivantes :

- Les faces d'un simplexe de C appartiennent à C. C'est à dire que si un triangle (un 2-simplexe) est dans le complexe simplicial, ses arêtes (1-simplexe) et ses sommets (0-simplexe) sont aussi dans le complexe simplicial.
- L'intersection de deux simplexes de C est soit vide s'il sont disjoints, soit un simplexe de C de dimension inférieure.

Définition La dimension d'un complexe simplicial est la dimension de son élément le plus grand.

Définition Un complexe simplicial est pur lorsque chaque simplexe est inclus dans un simplexe de la dimension du complexe.

Définition Un complexe simplicial de dimension d est connecté si la relation d'adjacence (deux complexes de dimension d partageant un simplexe de dimension $d-1$) forme un graphe connecté.

Définition Un point du complexe simplical est dit singulier si son voisinage n'est pas homéomorphe à un disque, à une boule ou à un demi disque.

1.3.2 Définitions

Soit M un ensemble de points. De façon informelle, une triangulation de cet ensemble de points est un maillage $\mathcal{M}$ tel que :

- Tous les polygones qui composent $\mathcal{M}$ sont des triangles.
- Tous les points initiaux sont sommets d'au moins un triangle.
- Les sommets des triangles de $\mathcal{M}$ sont uniquement les points initiaux.
- Il n'y a pas de recouvrement de triangles, l'intersection de deux triangles est soit vide, soit un sommet, soit une arête.

De plus, il arrive parfois qu'on impose à la triangulation d'être en une seule composante connexe (la connexité pouvant être portée sur les sommets ou les arêtes). Toutes ces propriétés empiriques nous conduisent vers une définition plus formelle [cga08] :

Définition Une triangulation est un 2-complexe simplicial pur, connecté et sans singularité.

1.3.3 Techniques de triangulations

La triangulation en dimension 2 la plus connue est certainement la triangulation de Delaunay [Del34, LS80]. Un triangle est de Delaunay si son cercle circonscrit n'englobe pas d'autres points. En d'autres termes, sur la figure 1.4(b), le triangle qui supporte le cercle n'est pas un triangle de Delaunay car son cercle circonscrit contient un point à l'intérieur. Par contre, la figure 1.4(c) présente un triangle de delaunay, puisqu'aucun autre point n'est à l'intérieur du cercle.

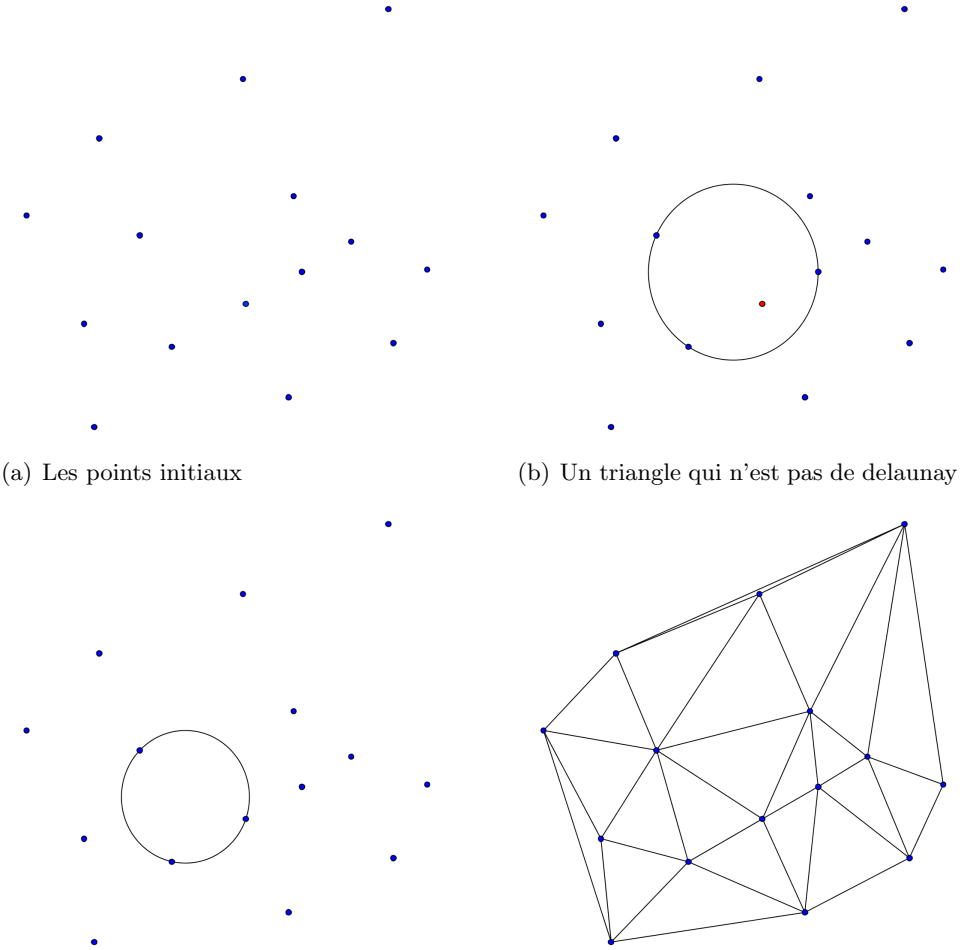
La triangulation de Delaunay est formée de tous les triangles de Delaunay.

Le dual de la triangulation de delaunay est le diagramme de Voronoï [Aur91]. Ce diagramme est formé des zones autour de chaque point. Chaque point de la zone est plus proche de son point d'ancrage que de tous les autres points du plan, comme illustré sur la figure 1.5.

Soit M un ensemble de points. Soit α le rayon d'un cercle C. Ce cercle se déplace dans le plan au milieu des points de M. On appelle Alpha-shapes [EM94] l'ensemble de segments formé par les paires de points où le cercle prend appui sans contenir d'autre point du plan. La figure 1.6(a) montre un exemple.

Cette technique des alpha shapes 2D produit des segments qui sont une partie de la triangulation de delaunay. C'est aussi une généralisation de l'enveloppe convexe. En effet, en prenant une boule de rayon infini, les seules paires de points qui satisfont la condition forment les arêtes de l'enveloppe convexe, comme sur la figure 1.6(b).

Il existe aussi des triangulations en 3 dimensions. La généralisation des alpha-shapes en 3D donne une triangulation du nuage de points 3D. De même, la triangulation de Delaunay est aussi généralisable en 3D [RT95,



(a) Les points initiaux

(b) Un triangle qui n'est pas de delaunay

(c) Un triangle de delaunay

(d) La triangulation de Delaunay

FIGURE 1.4 – Triangulation de Delaunay

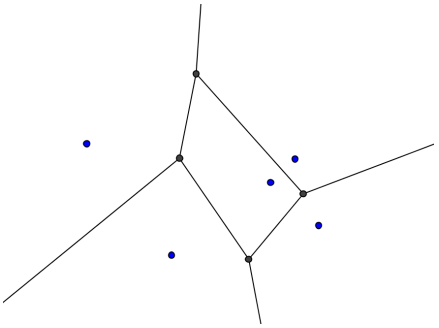
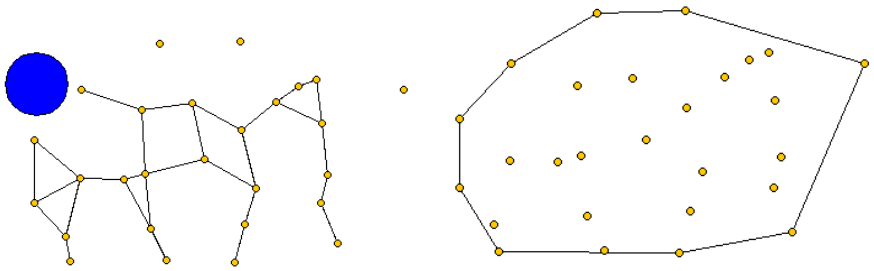


FIGURE 1.5 – Exemple de diagramme de Voronoï



(a) La boule bleue prend appui sur chaque paire de points pour former des segments

(b) Le rayon de la boule est infini, on retrouve l'enveloppe convexe du nuage de points

FIGURE 1.6 – Exemples d'alpha shape.

HI97] en prenant en compte la sphère portée par quatre points, et forme une tétraedrisation. Ainsi, pour trouver une triangulation, il faut un traitement supplémentaire.

1.4 Homologie et géométrie discrète différentielle

Rappelons brièvement dans cette partie la notion de bord utilisée pour définir les groupes d'homologie d'un complexe simplicial telle qu'elle apparaît en topologie algébrique ou dans le cadre de la géométrie discrète différentielle (DDG pour discrete differential geometry [GD06]).

Ces théories s'appuient sur une extension de la notion de complexe simplicial : le complexe simplicial orienté (figure 1.7) :

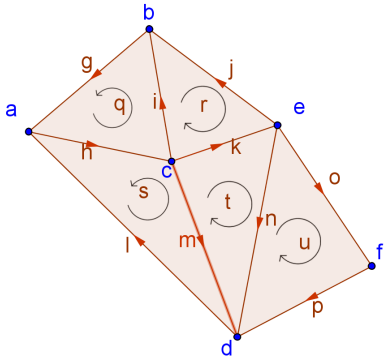


FIGURE 1.7 – Un complexe simplicial orienté C

Définition Le bord d'un k-simplexe est défini comme la somme signée de ses $(k-1)$-simplexes

Exemples : $\partial r = i - j - k$, $\partial h = c - a$.

Par convention, le bord d'un sommet est nul : $\partial e = 0$.

Définition Une chaîne est une combinaison linéaire de simplexes.

Définition Une k-chaîne est une combinaison linéaire de k-simplexes (avec des coefficients pris dans un groupe : $\mathbb{Z}$, $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{R}$ ...)

Définition C_k est le groupe des k-chaînes.

Par linéarité, on étend facilement la notion de bord de simplexes, à la notion de bord de chaîne. Par exemple :

$$\begin{aligned} \partial(2r+q) &= 2\partial r + \partial q \\ &= 2(i-j-k) + (g+h+i) \\ &= 3i - 2j - 2k + g + h \end{aligned}$$

Proposition 1 *Le bord d'une k-chaîne est une $(k-1)$-chaîne.*

Cette proposition nous donne donc l'enchainement suivant :

$$\ldots \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k+1}} \ldots \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0$$

Théorème 2 *L'opérateur bord appliqué deux fois successivement donne un résultat nul. Autrement dit, $\partial \circ \partial = 0$*

Ce théorème implique que $Im\ \partial_k \subset Ker\ \partial_{k-1}$. On peut alors construire un groupe quotient qui est le groupe d'homologie [Ser53] :

Définition Les groupes d'homologie H_{k-1} sont définis comme le quotient de $Ker\ \partial_{k-1}$ et de $Im\ \partial_k$

$$H_{k-1} = Ker\ \partial_{k-1} / Im\ \partial_k$$

Ces outils permettent de définir les nombres de Betti [Die78, Mil64] et permettent de distinguer des complexes simpliciaux de topologie différente.

Chapitre 2

Surfaces paramétrées

Toutes les définitions présentées le seront dans le cadre de $\mathbb{R}^3$, mais elles se généralisent en dimension n.

Une surface paramétrée est d'une façon générale l'ensemble image d'une application `S` :

$$\begin{array}{rccc} \texttt{S}: & X \subset \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & (x(s,t), y(s,t), z(s,t)) \end{array}$$

où la partie X de $\mathbb{R}^2$ est le plus souvent choisie homéomorphe à un disque (i.e. connexe et non trouée).

Dans le cas des surfaces de Bézier ou des surface B-Splines, les fonctions $x(s,t)$, $y(s,t)$ et $z(s,t)$ sont tout simplement des polynômes (polynômes par morceaux pour les surfaces B-Splines) des deux variables s et t.

Dans ce cadre, on peut écrire la fonction `S` :

$$\begin{array}{rccc} \texttt{S}: & X \subset \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & \left(\begin{array}{c} x(s,t) = \sum_{k=1}^{N} x_k * f_k(s,t) \\ y(s,t) = \sum_{k=1}^{N} y_k * f_k(s,t) \\ z(s,t) = \sum_{k=1}^{N} z_k * f_k(s,t) \end{array} \right) \end{array}$$

où $f_k(s,t)$ désigne une base de polynômes à deux variables.

Si on regroupe les coordonnées entre elles, cela nous donne :

$$\begin{array}{rccc} \texttt{S}: & X \subset \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & \sum_{k=1}^{N} P_k * f_k(s,t) \end{array}$$

Les point 3D $P_k = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix}$, qui représentent les coordonnées des fonc-

tions x, y et z dans la base $(f_k)_{1\leq k\leq N}$, sont appelés points de contrôle de la surface paramétrée. Ils contrôlent la déformation du morceau de plan X de $\mathbb{R}^2$ plongé dans $\mathbb{R}^3$. En général, ils n'appartiennent pas à la surface.

2.1 Surfaces de Bézier

Plutôt que de choisir comme base de polynômes le produit tensoriel $s^i t^j$ des monômes s^i et t^j, de nombreuses propriétés conduisent à choisir comme base le produit tensoriel des polynômes de Bernstein. On rappelle que les polynômes de Bernstein de degré n s'écrivent :

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \text{ pour } t \in [0,1]$$

Rappelons aussi brièvement quelques propriétés qui seront utilisées par la suite :

$$tB_{i,n}(t) = \frac{i+1}{n+1} B_{i+1,n+1}(t) \tag{2.1}$$

$$(n+1)B_{i,n}(t) = (i+1)B_{i+1,n+1}(t) + (n+1-i)B_{i,n+1}(t) \tag{2.2}$$

$$\sum_{i=0}^{n} B_{i,n}(t) = 1 \tag{2.3}$$

$$B_{i,n}(t) \geq 0 \qquad \forall t \in [0,1] \tag{2.4}$$

Soient m et n deux entiers fixés, on prend donc comme base de fonction f_k le produit tensoriel $B_{i,n}(s)B_{j,m}(t)$ avec des indices i allant de 0 à n et j de 0 à m.

Dans ce cadre, une surface de Bézier de points de contrôle $P_{i,j}$ est l'image de l'application :

$$\begin{array}{rccc} \mathtt{S}: & [0,1]\times[0,1] & \to & \mathbb{R}^3 \\ & (s,t) & \mapsto & \displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j} B_{i,m}(s) B_{j,n}(t) \end{array} \tag{2.5}$$

La figure 2.1 représente une surface de Bézier.

Il s'ensuit de (2.4) et (2.3) que $\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(s)B_{j,n}(t) = 1$, avec comme conséquence le fait que la surface de Bézier est dans l'enveloppe convexe de ses points de contrôle.

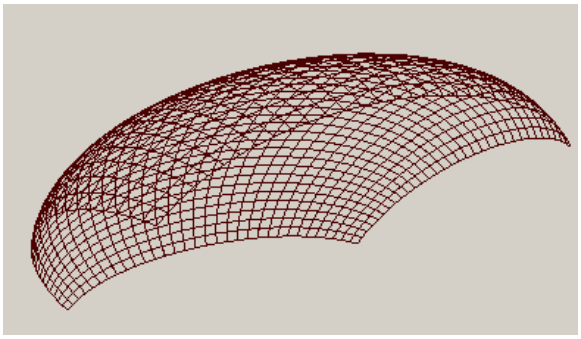


FIGURE 2.1 – Exemple de surface de Bézier

2.1.1 Dérivée d'une surface de Bézier

La dérivée partielle d'une surface de Bézier s'écrit :

$$\begin{aligned}\left(\frac{\partial}{\partial s}\mathtt{S}\right)(s,t) &= \frac{\partial}{\partial s}\left(\sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j}B_{i,m}(s)B_{j,n}(t)\right)\\ &= m\sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j}\left(B_{i-1,m-1}(s) - B_{i,m-1}(s)\right)B_{j,n}(t) \qquad (2.6)\end{aligned}$$

car $B'_{i,m}(s) = m(B_{i-1,m-1}(s) - B_{i,m-1}(s))$ et avec comme convention $B_{-1,m}(s) = 0$ et $B_{m+1,m}(s) = 0$. Avec un décalage d'indices, cette égalité conduit à :

$$\left(\frac{\partial}{\partial s}\mathtt{S}\right)(s,t) = m\sum_{i=0}^{m-1}\sum_{j=0}^{n}\left(P_{i+1,j} - P_{i,j}\right)B_{i,m-1}(s)B_{j,n}(t) \qquad (2.7)$$

2.1.2 Élevation du degré

L'élévation du degré consiste à considérer une courbe de Bézier de degré n comme une courbe de Bézier de degré $n+1$. En tant que telle, ses points de contrôle (au nombre de $n+2$) s'obtiennent à partir des points de contrôle initiaux (au nombre de $n+1$). On peut les obtenir simplement en utilisant l'application linéaire H_{n+1} qui plonge les polynômes de $\mathbb{R}_n[X]$ dans $\mathbb{R}_{n+1}[X]$.

$$\begin{aligned}H_{n+1} :&\mathbb{R}_n[X]\to\mathbb{R}_{n+1}[X]\\ &p \qquad\mapsto p\end{aligned}$$

Selon que l'on considère la base canonique des polynômes, ou bien la base des polynômes de Bernstein (en utilisant l'équation (2.2)) la matrice de ce plongement s'écrit :

$$H_{n+1}^{\text{can}} = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ 0 & \cdots & 0 \end{pmatrix}$$

$$H_{n+1}^{\text{Bern}} = \frac{1}{n+1} \begin{pmatrix} n+1 & & & & \\ 1 & n & & & \\ & 2 & \ddots & & \\ & & \ddots & 1 & \\ & & & n+1 \end{pmatrix}$$

L'opérateur d'élévation du degré n au degré p se nomme $H_{n\to p}$. Il est obtenu par composition de plusieurs élévations successives du degré $n+1$ jusqu'au degré p.

$$H_{n\to p} = H_p \times H_{p-1} \times \cdots \times H_{n+1}$$

2.2 Surfaces B-Splines

Les surfaces B-Spline utilisent comme base de fonction le produit tensoriel des fonctions définies par la formule de Cox-de Boor.

Soient deux entiers m et n. Soient deux entiers k et l et des vecteurs $S = \{s_0, \ldots, s_{m+k-1}\}$ et $T = \{t_0, \ldots, t_{n+l-1}\}$, avec $s_0 \leq s_1 \leq \ldots \leq s_{m+k-1}$ et $t_0 \leq t_1 \leq \ldots \leq t_{n+l-1}$.

S et T sont appelés vecteur nodaux. $k-1$ (resp. $l-1$) représente le degré de la surface B-Spline selon l'abscisse s (resp. t). $m+1$ (resp. $n+1$) représente le nombre de points de contrôle selon l'abscisse s (resp. t).

La formule de Cox-de Boor est la suivante [1] :

$$\begin{cases} N_{i,r}(t) = \dfrac{t - t_i}{t_{i+r-1} - t_i} N_{i,r-1}(t) + \dfrac{t_{i+r} - t}{t_{i+r} - t_{i+1}} N_{i+1,r-1}(t) \\ \\ N_{i,1}(t) = 1 \text{ si } t_i \leq t \leq t_{i+1} \text{ et } (i = 0 \text{ ou } t \neq t_i) \\ \\ N_{i,1}(t) = 0 \text{ sinon} \end{cases}$$

Une illustration des fonctions $N_{i,4}$ est présente figure 2.2.

Ainsi, on écrit une surface B-Spline avec $x(s,t)$, $y(s,t)$ et $z(s,t)$ de la forme :

1. Il est possible d'obtenir une division par zéro. Dans ce cas, nous fixerons $\frac{0}{0} = 0$

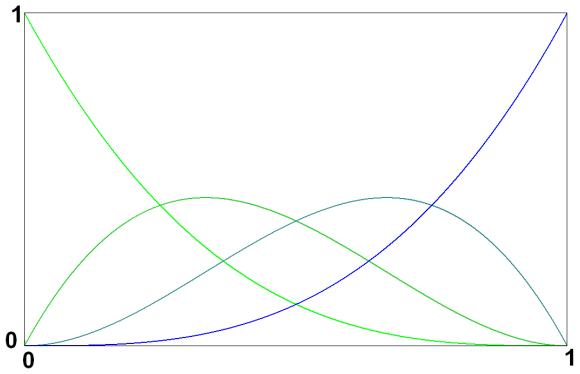


FIGURE 2.2 – Illustration des fonctions $N_{i,4}$, vecteurs nodaux $S = T = \{0, 0, 0, 0, 1, 1, 1, 1\}$

$$\begin{array}{rccl} x: & [s_{k-1}, s_m] \times [t_{l-1}, t_n] & \rightarrow & \mathbb{R} \\ & (s,t) & \mapsto & x(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} x_{i,j} N_{i,k}(s) N_{j,l}(t) \end{array}$$

La forme générale d'une surface B-Spline est donc :

$$\begin{array}{rccl} \mathtt{S}: & [s_{k-1}, s_m] \times [t_{l-1}, t_n] & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & \sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j} N_{i,k}(s) N_{j,l}(t) \end{array} \tag{2.8}$$

2.3 Recollement de patches

Les surfaces de Bézier et B-Splines constituent des *carreaux* ou *patches*. En assemblant ces patches, on peut générer des surfaces plus complexes (non nécessairement homéomorphes à un disque), mais cet assemblage n'est pas sans poser des problèmes de régularité au niveau des recollements. Nous n'étudierons que le recollement de surfaces de Bézier, à noter que le recollement C^0 de deux surfaces B-Splines uniformes se fait de la même manière que les surfaces de Bézier. Certains recollements de surfaces B-Splines sont étudiés, dans certains cas particuliers, dans les références suivantes [MBVW95, EH96, Pet94, Pet95, Pet02].

Nous étudierons dans un premier temps le recollement entre deux patches de Bézier et par la suite le recollement global de tous les patches de Bézier d'une même surface.

2.3.1 Recollement de deux surfaces de Bézier

Soient 2 patches $\mathbb{A}$ et $\mathbb{B}$, leurs surfaces de Bézier respectives $\mathtt{S}^{\mathbb{A}}$ et $\mathtt{S}^{\mathbb{B}}$ de points de controles $P^{\mathbb{A}}_{i,j}$, $P^{\mathbb{B}}_{k,l}$ et de degrés $m^{\mathbb{A}}$, $n^{\mathbb{A}}$, $m^{\mathbb{B}}$, $n^{\mathbb{B}}$. Nous allons étudier les conditions de leur recollement, c'est-à-dire plus précisément les contraintes sur les points de contrôle qui impliquent la régularité du recollement de $\mathtt{S}^{\mathbb{A}}$ et $\mathtt{S}^{\mathbb{B}}$.

2.3.1.1 Recollement C^0

Le recollement C^0 consiste à ce que les deux bords des deux patches soient contigus. Quitte à échanger les deux surfaces et à reparamétrer, on peut considérer que les deux bords concernés sont les courbes de Bézier $\left(\mathtt{S}^{\mathbb{A}}\right)_{s=1}$ et $\left(\mathtt{S}^{\mathbb{B}}\right)_{s=0}$. La condition de recollement C^0 devient alors l'égalité pour tout t de ces deux courbes.

$$\sum_{i=0}^{m^{\mathbb{A}}}\sum_{j=0}^{n^{\mathbb{A}}} P^{\mathbb{A}}_{i,j} B_{i,m^{\mathbb{A}}}(1) B_{j,n^{\mathbb{A}}}(t) = \sum_{k=0}^{m^{\mathbb{B}}}\sum_{l=0}^{n^{\mathbb{B}}} P^{\mathbb{B}}_{k,l} B_{k,m^{\mathbb{B}}}(0) B_{l,n^{\mathbb{B}}}(t) \quad \forall t \in [0,1]$$

soit

$$\sum_{j=0}^{n^{\mathbb{A}}} P^{\mathbb{A}}_{m^{\mathbb{A}},j} B_{j,n^{\mathbb{A}}}(t) = \sum_{l=0}^{n^{\mathbb{B}}} P^{\mathbb{B}}_{0,l} B_{l,n^{\mathbb{B}}}(t) \quad \forall t \in [0,1] \tag{2.9}$$

Dans le cas ou $n^{\mathbb{A}}$ est égal à $n^{\mathbb{B}}$, cette contrainte se traduit simplement en une égalité des points de contrôle le long de la frontière commune :

$$P^{\mathbb{A}}_{m^{\mathbb{A}},j} = P^{\mathbb{B}}_{0,j} \ \forall j \in [\![0..n]\!]$$

Dans le cas ou $n^{\mathbb{A}}$ est différent de $n^{\mathbb{B}}$ (sans restreindre la généralité du propos, on suppose $n^{\mathbb{A}} < n^{\mathbb{B}}$), on peut simplifier l'équation (2.9) afin de n'avoir qu'un nombre fini de contraintes linéaires. On élève le degré de la courbe $\left(\mathtt{S}^{\mathbb{A}}\right)_{s=1}$ afin d'atteindre le degré $n^{\mathbb{B}}$.

Soient $P'^{\mathbb{A}}_{m^{\mathbb{A}},.}$ les points de contrôle de la courbe $\left(\mathtt{S}^{\mathbb{A}}\right)_{s=1}$ élevée au degré $n^{\mathbb{B}}$. La condition C^0 s'écrit alors :

$$\sum_{j=0}^{n^{\mathbb{B}}} P'^{\mathbb{A}}_{m^{\mathbb{A}},j} B_{j,n^{\mathbb{B}}}(t) = \sum_{l=0}^{n^{\mathbb{B}}} P^{\mathbb{B}}_{0,l} B_{l,n^{\mathbb{B}}}(t) \ \ \forall t \in [0,1]$$

Comme précédemment, on obtient une égalité de points de contrôle :

$$P'^{\mathbb{A}}_{m^{\mathbb{A}},j} = P^{\mathbb{B}}_{0,j} \ \forall j \in [\![0..n^{\mathbb{B}}]\!]$$

Élever au degré $n^{\mathbb{B}}$ une courbe de degré $n^{\mathbb{A}}$ revient à appliquer $n^{\mathbb{B}} - n^{\mathbb{A}}$ élévations de degré. La condition C^0 ($P'^{\mathbb{A}}_{m^{\mathbb{A}},j} = P^{\mathbb{B}}_{0,j}\ \forall j \in [\![0..n^{\mathbb{B}}]\!]$) entre les deux patches $\mathbb{A}$ et $\mathbb{B}$ s'écrit donc :

$$H_{n^{\mathbb{B}} \to n^{\mathbb{A}}} \times P^{\mathbb{A}}_{m^{\mathbb{A}},\cdot} = P^{\mathbb{B}}_{0,j}\ \forall j \in [\![0..n^{\mathbb{B}}]\!] \tag{2.10}$$

C'est une relation linéaire entre les points de contrôle de la surface $\mathtt{S}^{\mathbb{A}}$ et les points de contrôle de la surface $\mathtt{S}^{\mathbb{B}}$.

2.3.1.2 Recollement C^k

Le recollement C^1 consiste à avoir une condition de continuité C^0 ainsi qu'une condition sur les dérivées : toutes les dérivées partielles d'ordre 1 doivent être les mêmes le long de la courbe commune. Le recollement C^k consiste à ce que toutes les dérivées partielles *jusqu'à* l'ordre k soient égales le long de la courbe commune.

Première étape, on a d'après le théorème de Schwartz [1] :

$$\left(\frac{\partial^2}{\partial s \partial t}\mathtt{S}^{\mathbb{A}}\right)(s,t) = \left(\frac{\partial^2}{\partial t \partial s}\mathtt{S}^{\mathbb{A}}\right)(s,t) \tag{2.11}$$

Deuxième étape, comme la dérivée d'une surface de Bézier est une surface de Bézier, et que $\left(\frac{\partial^2}{\partial s \partial t}\mathtt{S}^{\mathbb{A}}\right)(s,t) = \left(\frac{\partial^2}{\partial t \partial s}\mathtt{S}^{\mathbb{A}}\right)(s,t)$, nous pouvons affirmer que toutes les dérivées partielles d'ordre k de la surface de Bézier $\mathtt{S}^{\mathbb{A}}$ s'écrivent ainsi :

$$\left(\frac{\partial^k}{\partial s^{k1} \partial t^{k2}}\mathtt{S}^{\mathbb{A}}\right)(s,t) \text{ avec } k1 + k2 = k$$

Troisième étape, supposons désormais que nous avons un recollement C^{k-1}. En particulier, nous avons :

$$\left(\frac{\partial^k}{\partial s^{k1}}\mathtt{S}^{\mathbb{A}}\right)_{s=1}(t) = \left(\frac{\partial^k}{\partial s^{k1}}\mathtt{S}^{\mathbb{B}}\right)_{s=0}(t)$$
$$\forall t \in [0,1] \text{ et pour } k1 \in [\![0..k-1]\!]$$

et donc

$$\left(\frac{\partial^k}{\partial s^{k1} \partial t^{k2}}\mathtt{S}^{\mathbb{A}}\right)_{s=1}(t) = \left(\frac{\partial^k}{\partial s^{k1} \partial t^{k2}}\mathtt{S}^{\mathbb{B}}\right)_{s=0}(t)$$
$$\forall t \in [0,1]\ \text{, pour } k1 + k2 = k \text{ et pour } k1 \leq k-1$$

1. La fonction $\mathtt{S}^{\mathbb{A}}$ est deux fois différentiable au point (s,t) et vérifie les conditions d'application du théorème de Schwartz

Pour pouvoir décider du recollement C^k, il faut que toutes les dérivées d'ordre k soient égales sur les deux courbes. Ainsi, le recollement C^k sachant le recollement C^{k-1}, le long des courbes $s=0$ et $s=1$, se résume à l'égalité des dérivées $\left(\frac{\partial^k}{\partial s^k}\mathtt{S}^{\mathbb{A}}\right)_{s=1}(t)$ et $\left(\frac{\partial^k}{\partial s^k}\mathtt{S}^{\mathbb{B}}\right)_{s=0}(t)$

Dernière étape, on peut déduire des calculs précédents (équation 2.7) :

$$\left(\frac{\partial^2}{\partial s^2}\mathtt{S}^{\mathbb{A}}\right)(s,t) = m^{\mathbb{A}}\left(m^{\mathbb{A}}-1\right) \sum_{i=0}^{m^{\mathbb{A}}-2}\sum_{j=0}^{n^{\mathbb{A}}}\left(P^{\mathbb{A}}_{i+2,j} - 2P^{\mathbb{A}}_{i+1,j} + P^{\mathbb{A}}_{i,j}\right) B_{i,m^{\mathbb{A}}-2}(s)B_{j,n^{\mathbb{A}}}(t) \tag{2.12}$$

En généralisant, on obtient :

$$\left(\frac{\partial^k}{\partial s^k}\mathtt{S}^{\mathbb{A}}\right)(s,t) = m^{\mathbb{A}}\left(m^{\mathbb{A}}-1\right)\dots\left(m^{\mathbb{A}}-k+1\right) \sum_{i=0}^{m^{\mathbb{A}}-k}\sum_{j=0}^{n^{\mathbb{A}}} {P^{\mathbb{A}}}'_{i,j,k} B_{i,m^{\mathbb{A}}-k}(s)B_{j,n^{\mathbb{A}}}(t) \tag{2.13}$$

avec ${P^{\mathbb{A}}}'_{i,j,k} = \sum_{p=i}^{i+k}(-1)^{k-(p-i)}\binom{k}{p-i}P^{\mathbb{A}}_{p,j}$

Ainsi, l'égalité entre les deux dérivées $k^{\text{ième}}$ en s s'écrit donc :

$$\begin{aligned}
&\left(\frac{\partial^k}{\partial s^k}\mathtt{S}^{\mathbb{A}}\right)_{s=1}(t) = \left(\frac{\partial^k}{\partial s^k}\mathtt{S}^{\mathbb{B}}\right)_{s=0}(t) \quad \forall t\in[0,1] \\
\iff \quad & m^{\mathbb{A}}\left(m^{\mathbb{A}}-1\right)\dots\left(m^{\mathbb{A}}-k+1\right)\sum_{i=0}^{m^{\mathbb{A}}-k}\sum_{j=0}^{n^{\mathbb{A}}} {P^{\mathbb{A}}}'_{i,j,k} B_{i,m^{\mathbb{A}}-k}(1)B_{j,n^{\mathbb{A}}}(t) \\
= \quad & m^{\mathbb{B}}\left(m^{\mathbb{B}}-1\right)\dots\left(m^{\mathbb{B}}-k+1\right)\sum_{r=0}^{m^{\mathbb{B}}-k}\sum_{l=0}^{n^{\mathbb{B}}} {P^{\mathbb{B}}}'_{r,l,k} B_{r,m^{\mathbb{B}}-k}(0)B_{l,n^{\mathbb{B}}}(t) \\
& \forall t\in[0,1] \\
\iff \quad & m^{\mathbb{A}}\left(m^{\mathbb{A}}-1\right)\dots\left(m^{\mathbb{A}}-k+1\right) \\
& \sum_{j=0}^{n^{\mathbb{A}}}\left(\sum_{\substack{p=\\ m^{\mathbb{A}}-k}}^{m^{\mathbb{A}}}(-1)^{m^{\mathbb{A}}-p}\binom{k}{p-m^{\mathbb{A}}+k}P^{\mathbb{A}}_{p,j}\right)B_{j,n^{\mathbb{A}}}(t) \\
= \quad & m^{\mathbb{B}}\left(m^{\mathbb{B}}-1\right)\dots\left(m^{\mathbb{B}}-k+1\right)\sum_{l=0}^{n^{\mathbb{B}}}\left(\sum_{q=0}^{k}(-1)^{k-q}\binom{k}{q}P^{\mathbb{B}}_{q,l}\right)B_{l,n^{\mathbb{B}}}(t) \\
& \forall t\in[0,1]
\end{aligned}$$

$$
\begin{aligned}
\iff \quad & m^{\mathbb{A}}\left(m^{\mathbb{A}}-1\right)\dots\left(m^{\mathbb{A}}-k+1\right) \\
& \sum_{j=0}^{n^{\mathbb{A}}}\left(\sum_{p=0}^{k}(-1)^{k-p}\binom{k}{p}P^{\mathbb{A}}_{p-k+m^{\mathbb{A}},j}\right)B_{j,n^{\mathbb{A}}}(t) \qquad (2.14)\\
= \quad & m^{\mathbb{B}}\left(m^{\mathbb{B}}-1\right)\dots\left(m^{\mathbb{B}}-k+1\right)\sum_{l=0}^{n^{\mathbb{B}}}\left(\sum_{q=0}^{k}(-1)^{k-q}\binom{k}{q}P^{\mathbb{B}}_{q,l}\right)B_{l,n^{\mathbb{B}}}(t) \\
& \forall t\in[0,1]
\end{aligned}
$$

Cette condition d'égalité entre deux dérivées $k^{\text{ième}}$ peut se simplifier si l'on a $n^{\mathbb{A}}=n^{\mathbb{B}}=n$ (sinon, il est nécessaire de procéder à une élévation de degré comme nous le ferons par la suite)

$$
\begin{aligned}
& \forall i\in[\![0..n]\!] \\
& m^{\mathbb{A}}\dots\left(m^{\mathbb{A}}-k+1\right)\sum_{p=0}^{k}(-1)^{k-p}\binom{k}{p}P^{\mathbb{A}}_{p-k+m^{\mathbb{A}},i} \qquad (2.15)\\
= \quad & m^{\mathbb{B}}\dots\left(m^{\mathbb{B}}-k+1\right)\sum_{q=0}^{k}(-1)^{k-q}\binom{k}{q}P^{\mathbb{B}}_{q,i}
\end{aligned}
$$

Si de plus $m^{\mathbb{A}}=m^{\mathbb{B}}=m$, la condition d'égalité entre deux dérivées $k^{\text{ième}}$ devient :

$$
\left(\sum_{p=0}^{k}(-1)^{k-p}\binom{k}{p}P^{\mathbb{A}}_{p-k+m,i}\right)=\left(\sum_{q=0}^{k}(-1)^{k-q}\binom{k}{q}P^{\mathbb{B}}_{q,i}\right)\quad\forall i\in[\![0..n]\!] \quad (2.16)
$$

Dans le cas ou $n^{\mathbb{A}}$ n'est pas égal à $n^{\mathbb{B}}$, nous allons procéder de la même manière que pour la condition C^0. Ainsi, l'équation (2.14) peut être mise sous la forme suivante :

$$
\sum_{j=0}^{n^{\mathbb{A}}}P^{k\,\mathbb{A}}_{j}B_{j,n^{\mathbb{A}}}(t)=\sum_{l=0}^{n^{\mathbb{B}}}P^{k\,\mathbb{B}}_{l}B_{l,n^{\mathbb{B}}}(t)\quad\forall t\in[0,1] \qquad (2.17)
$$

avec

$$
P^{k\,\mathbb{A}}_{j}=m^{\mathbb{A}}\left(m^{\mathbb{A}}-1\right)\dots\left(m^{\mathbb{A}}-k+1\right)\sum_{p=0}^{k}(-1)^{k-p}\binom{k}{p}P^{\mathbb{A}}_{p-k+m^{\mathbb{A}},j}
$$

et

$$
P^{k\,\mathbb{B}}_{l}=m^{\mathbb{B}}\left(m^{\mathbb{B}}-1\right)\dots\left(m^{\mathbb{B}}-k+1\right)\sum_{q=0}^{k}(-1)^{k-q}\binom{k}{q}P^{\mathbb{B}}_{q,l}
$$

En supposant $n^{\mathbb{A}} < n^{\mathbb{B}}$, et vu le rôle symétrique des deux patches, les contraintes C^k sachant le recollement C^{k-1} peuvent donc s'écrire ainsi (d'après l'équation (2.10)) :

$$H_{n^{\mathbb{A}} \to n^{\mathbb{B}}} \times {P^k}_{\cdot}^{\mathbb{A}} = {P^k}_{\cdot}^{\mathbb{B}} \tag{2.18}$$

Nous allons maintenant écrire plus simplement cette condition, en introduisant les deux opérateurs L, R et I. L'opérateur $R_{n^{\mathbb{A}},m^{\mathbb{A}}}$, R pour *right*, dépend de deux paramètres et agit sur un vecteur de points de contrôle : il transforme les points de contrôle $P^{\mathbb{A}}_{i,j}$ en $\left(R_{n^{\mathbb{A}},m^{\mathbb{A}}}\left(P^{\mathbb{A}}\right)\right)_{i,j} = P^{\mathbb{A}}_{i-1,j}$ et insère des 0 là où les indices de $P^{\mathbb{A}}$ sont non définis. Si l'on considère la matrice des points de contrôle, c'est un décalage vers la droite. L'opérateur L est le même en décalant les points de contrôle à gauche, et l'opérateur I est l'identité. Pour des raison de lourdeur de notation, nous les noterons par la suite L, R et I.

Les formulations de ${P^k}_{\cdot}^{\mathbb{A}}$ et ${P^k}_{\cdot}^{\mathbb{B}}$ montrent que ce sont des sommes des $k+1$ rangées de points de contrôle les plus proches de la bordure, et pondérées par les coefficients du triangle de Pascal. En remarquant que les opérateurs $(I-R)^k$ et $(L-I)^k$ produisent les coefficients binomiaux, nous pouvons déduire la formulation suivante utilisant les opérateurs L, R et I :

$${P^k}_{\cdot}^{\mathbb{A}} = \begin{pmatrix} 0 & I_{n^{\mathbb{A}}+1} \end{pmatrix} \times (I-R)^k \times P^{\mathbb{A}} \tag{2.19}$$

$${P^k}_{\cdot}^{\mathbb{B}} = \begin{pmatrix} I_{n^{\mathbb{B}}+1} & 0 \end{pmatrix} \times (L-I)^k \times P^{\mathbb{B}} \tag{2.20}$$

Les résultats sont aussi valables pour $k = 0$. Ainsi, le recollement C^k se résume en $k+1$ conditions :

$$\begin{aligned} & H_{n^{\mathbb{A}} \to n^{\mathbb{B}}} \times \begin{pmatrix} 0 & I_{n^{\mathbb{A}}+1} \end{pmatrix} \times (I-R)^r \times P^{\mathbb{A}} \\ = \quad & \begin{pmatrix} I_{n^{\mathbb{B}}+1} & 0 \end{pmatrix} \times (L-I)^r \times P^{\mathbb{B}} \qquad \forall r \in [\![0..k]\!] \end{aligned} \tag{2.21}$$

Cette condition est bien une condition linéaire liant les points de contrôle initiaux des surfaces à recoller.

2.3.1.3 Recollement G^1

La condition G^1, comme la condition C^1, s'ajoute à la condition C^0. Elle consiste à ce que les plans tangents à chaque surface le long de la courbe commune soient identiques.

Nous allons reprendre les conditions de recollement G^1 telles qu'elle sont décrites dans [Far02]. Quitte à faire une élévation de degrés sur la facette ayant le degré le plus faible, nous pouvons supposer que $n^{\mathbb{A}} = n^{\mathbb{B}} = n$. Nous

nous intéresserons à la rangée de points de contrôle commune (puisque la condition C^0 est valide) et aux deux rangées de points de contrôle situées de chaque côté.

Toutes les explications qui suivent sont illustrées sur la figure 2.3.

Nous élevons le degré de la courbe commune. Nous avons donc trois rangées de points p, q et r. Les points p et r sont au nombre de $n+1$ tandis que la rangée q contient $n+2$ points. Nous pouvons introduire les points $p(t)$ et $r(t)$ qui sont la position à t fixé de la courbe de Bézier des rangées de point correspondantes, et $qb(t)$ et $qt(t)$ qui sont les deux derniers points de l'algorithme de Casteljau de la construction de la courbe commune. Les lieux des points $qb(t)$ et $qt(t)$ peuvent aussi être vus comme des courbes de Bézier de degré n, si on ne prend pas en compte le dernier point q_{n+1} (resp. le premier point q_0).

On peut donc écrire explicitement les expression des points $p(t)$, $qb(t)$, $qt(t)$ et $r(t)$:

$$p(t) = \sum_{i=0}^{n} p_i B_{i,n}(t) \qquad qb(t) = \sum_{i=0}^{n} q_i B_{i,n}(t)$$
$$qt(t) = \sum_{i=0}^{n} q_{i+1} B_{i,n}(t) \qquad r(t) = \sum_{i=0}^{n} r_i B_{i,n}(t)$$

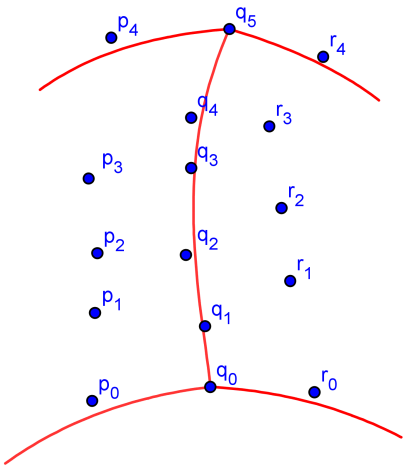
Le point $q(t)$ de la courbe commune est sur le segment $[qb(t), qt(t)]$ dont la direction fournit la tangente. De plus, les vecteurs dérivés selon la seconde coordonnée s sont les vecteurs $\overrightarrow{q(t)p(t)}$ et $\overrightarrow{q(t)r(t)}$. Cela signifie que le plan tangent au point $q(t)$ est porté par le triangle $qt(t)qb(t)p(t)$ pour le patch $\mathbb{A}$ et $qt(t)qb(t)r(t)$ pour le patch $\mathbb{B}$.

La condition G^1, qui consiste à ce que les plans tangents à chacune des facettes soient les mêmes, peut se résumer à la coplanarité des quatre points $p(t)$, $qb(t)$, $qt(t)$ et $r(t)$. Ils doivent donc être coplanaires pour tout t. Pour être plus précis, il faut que $r(t)$ et $p(t)$ soient de part et d'autre de $qb(t)$ et $qt(t)$, afin que la surface ne se replie pas sur elle-même (on aurait un point de rebroussement). Cela se traduit géométriquement par le fait que le segment $[p(t)r(t)]$ coupe la droite $(qb(t)qt(t))$ ce qui signifie, pour chaque valeur de $t \in [0,1]$, qu'il existe un coefficient $\lambda(t) \in [0,1]$ et un coefficient $\mu(t) \in \mathbb{R}$ tels que :

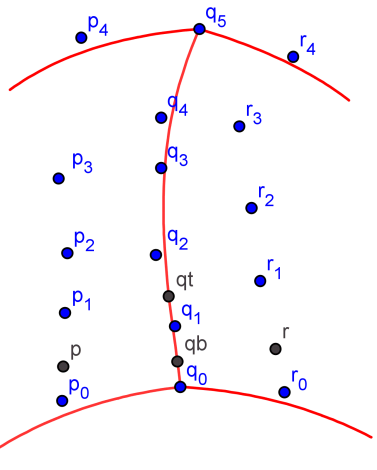
$$\lambda(t)p(t) + (1-\lambda(t))r(t) = \mu(t)qb(t) + (1-\mu(t))qt(t) \tag{2.22}$$

Si nous faisons l'hypothèse que les fonctions $\lambda(t)$ et $\mu(t)$ sont affines, c'est à dire $\lambda(t) = \lambda_0 + \lambda_1 t$ et $\mu(t) = \mu_0 + \mu_1 t$ [1], alors l'équation (2.22) devient :

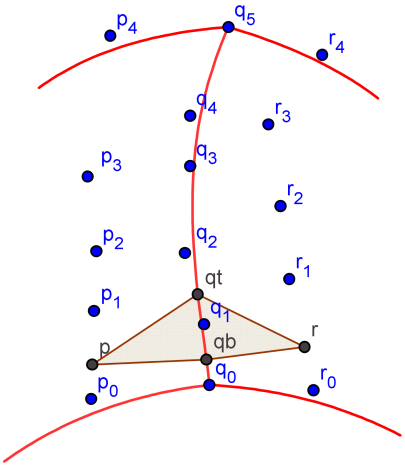
1. la condition $\lambda(t) \in [0,1]$ impose $0 \leq \lambda_0 \leq 1$ et $0 \leq \lambda_0 + \lambda_1 \leq 1$



(a) Les points de contrôle après élévation de degré de la courbe commune



(b) Les points p, qb, qt et r



(c) Le polygone « tangent »

FIGURE 2.3 – Position des points de contrôle dans le cas $n = 4$

$$\begin{aligned} & (\lambda_0 + \lambda_1 t) \sum_{i=0}^{n} p_i B_{i,n}(t) + (1 - \lambda_0 - \lambda_1 t) \sum_{i=0}^{n} r_i B_{i,n}(t) \\ = \quad & (\mu_0 + \mu_1 t) \sum_{i=0}^{n} q_i B_{i,n}(t) + (1 - \mu_0 + \mu_1 t) \sum_{i=0}^{n} q_{i+1} B_{i,n}(t) \end{aligned} \tag{2.23}$$

Ce qui donne, en arrangeant grâce à l'équation (2.1)

$$\begin{aligned} & \sum_{i=0}^{n} (\lambda_0 p_i + (1-\lambda_0) r_i) B_{i,n}(t) && + \lambda_1 \sum_{i=0}^{n} (p_i - r_i) \frac{i+1}{n+1} B_{i+1,n+1}(t) \\ = & \sum_{i=0}^{n} (\mu_0 q_i + (1-\mu_0) q_{i+1}) B_{i,n}(t) && + \mu_1 \sum_{i=0}^{n} (q_i - q_{i+1}) \frac{i+1}{n+1} B_{i+1,n+1}(t) \end{aligned} \tag{2.24}$$

En élevant le degré des termes $B_{i,n}(t)$ et en relevant tous les termes en $B_{i,n+1}(t)$, cela donne [1] :

$$\begin{aligned} & \forall i \in [\![0..n+1]\!] \\ & \frac{i}{n+1} \left[(\lambda_0 + \lambda_1) p_{i-1} + (1 - \lambda_0 - \lambda_1) r_{i-1} \right] + \left(1 - \frac{i}{n+1} \right) \left[\lambda_0 p_i + (1-\lambda_0) r_i \right] \\ = & \frac{i}{n+1} \left[(\mu_0 + \mu_1) q_{i-1} + (1 - \mu_0 - \mu_1) q_i \right] + \left(1 - \frac{i}{n+1} \right) \left[\mu_0 q_i + (1-\mu_0) q_{i+1} \right] \end{aligned} \tag{2.25}$$

Enfin, en posant [2] $\alpha_1 = \lambda_0 + \lambda_1$, $\beta_1 = \mu_0 + \mu_1$, $\alpha_0 = \lambda_0$ et $\beta_0 = \mu_0$, on obtient la condition G^1 suivante :

$$\begin{aligned} & \forall i \in [\![0..n+1]\!] \\ & \frac{i}{n+1} \left[\alpha_1 p_{i-1} + (1 - \alpha_1) r_{i-1} \right] + \left(1 - \frac{i}{n+1} \right) \left[\alpha_0 p_i + (1-\alpha_0) r_i \right] \\ = & \frac{i}{n+1} \left[\beta_1 q_{i-1} + (1 - \beta_1) q_i \right] + \left(1 - \frac{i}{n+1} \right) \left[\beta_0 q_i + (1-\beta_0) q_{i+1} \right] \end{aligned} \tag{2.26}$$

Ceci est la condition G^1 de recollement entre deux patches telle qu'elle est citée dans [Far02] et reprise dans [Du88, Hah06]. Remarquons que si nous fixons α_0, α_1, β_0 et β_1, cette condition est linéaire. En particulier, si nous prenons cette équation lorsque i vaut 0 et $n+1$, nous avons alors les équations aux sommets :

$$\begin{aligned} \alpha_0 p_0 + (1-\alpha_0) r_0 &= \beta_0 q_0 + (1-\beta_0) q_1 \\ \alpha_1 p_n + (1-\alpha_1) r_n &= \beta_1 q_n + (1-\beta_1) q_{n+1} \end{aligned} \tag{2.27}$$

1. Les termes q_{-1}, r_{-1}, p_{-1} et q_{n+2} n'existent pas mais sont considérés comme valant 0 et sont annulés quoi qu'il en soit par un facteur nul.

2. on a donc $0 \leq \alpha_0 \leq 1$ et $0 \leq \alpha_1 \leq 1$

2.3.2 Recollement de plusieurs surfaces de Bézier

Dans cette section, nous discuterons de la jonction fermée de plusieurs patches de Bézier partageant un sommet commun. Associé à la jonction deux à deux étudié dans la partie précédente, nous couvrons tous les cas de recollement d'objets complexes.

Dans un premier temps, on considère le recollement C^1 et les problèmes qu'il engendre (2.3.2.1 et 2.3.2.2). On étudie enfin le recollement G^1 de plusieurs patches (2.3.2.3).

2.3.2.1 Échec partiel du recollement C^1

Soient plusieurs patches $\mathbb{A}_i$ et leurs surfaces respectives $\mathtt{S}^{\mathbb{A}_i}$ ayant un sommet en commun, comme illustré sur le figure 2.4. La figure 2.5 représente schématiquement la position des points de contrôle de chacun des patches. Nous allons étudier le recollement de ces patches ainsi que les différentes conditions de continuité.

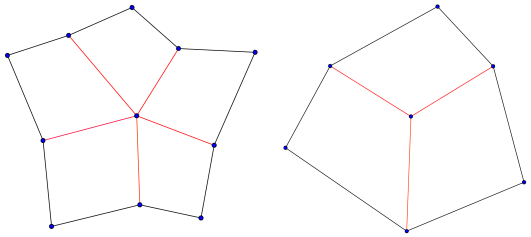


FIGURE 2.4 – Cas de figure de recollement de plusieurs patches ayant un sommet en commun

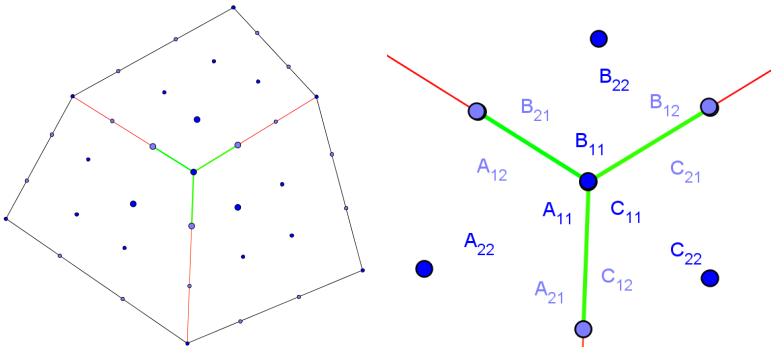


FIGURE 2.5 – Cas de figure de recollement de trois patches ayant un somment en commun, le recollement C^1 deux à deux n'est pas possible

Prenons par exemple le cas de la figure 2.5 où nous avons trois patches à recoller, de mêmes degrés. Le recollement C^1 des patches deux à deux (issu

des équations (2.21)) conduit à écrire les égalités suivantes [1] :

$$\begin{array}{ll} A_{12} = B_{21} & A_{22} - A_{12} = B_{21} - B_{22} \\ A_{11} = B_{11} & A_{21} - A_{11} = B_{11} - B_{12} \\ A_{21} = C_{12} & A_{22} - A_{21} = C_{12} - C_{22} \\ A_{11} = C_{11} & A_{12} - A_{11} = C_{11} - C_{21} \\ B_{12} = C_{21} & B_{22} - B_{12} = C_{21} - C_{22} \\ B_{11} = C_{11} & B_{21} - B_{11} = C_{11} - C_{12} \end{array} \tag{2.28}$$

On remarque en analysant la première ligne que ces contraintes impliquent une relation entre les points de contrôle. En effet, si on tient compte de l'équation $A_{12} = B_{21}$ dans la seconde équation $A_{22} - A_{12} = B_{21} - B_{22}$, cela nous donne la relation linéaire suivante :

$$A_{22} + B_{22} = 2B_{21}$$

La demi somme de A_{22} et B_{22} est égale à B_{21} (ou A_{12}). Autrement dit, B_{21} est le milieu du segment $[A_{22}B_{22}]$.

Ces équations de recollement C^1 imposent l'égalité des douze points A_{ij}, B_{kl} et C_{mn} cités précédemment. On a en particulier $A_{12} = A_{11}$, $B_{12} = B_{11}$ et $C_{12} = C_{11}$. Les dérivées au point de jonction des courbes communes étant proportionnelles aux différences entre ces trois paires de points, la continuité C^1 conduit à point stationnaire et potentiellement à un point de rebroussement. C'est un résultat cohérent d'un point de vue théorique, mais peu satisfaisant pour les applications visées.

Le recollement C^1 de trois patches soulève un problème dont nous allons maintenant préciser le cadre. Considérons donc désormais le problème général de la jonction fermée de n patches ($n > 2$) illustré sur la figure 2.6.

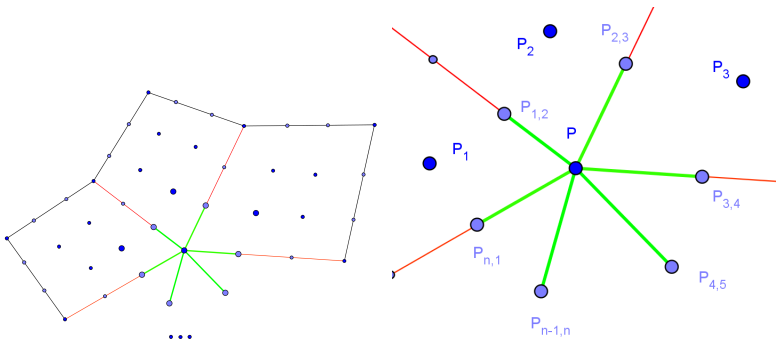


FIGURE 2.6 – Jonction fermée de n patches

1. La première colonne représente les conditions C^0 : chaque point de contrôle de la bordure est égal à son homologue dans l'autre bordure commune. La seconde colonne représente la condition C^1 qui s'ajoute à la condition C^0, les différences des points de contrôles sont égales le long de la bordure commune.

Les équations de continuité C^0 étant prises en compte, les équations de recollement C^1 (2.21) permettent d'écrire les égalités suivantes (voir la remarque précédente sur les demis sommes de points de contrôle) :

$$\begin{aligned} &P_{1,2} + P_{3,4} = 2P \\ &P_{2,3} + P_{4,5} = 2P \\ &P_{3,4} + P_{5,6} = 2P \\ &\dots \\ &P_{n-1,n} + P_{1,2} = 2P \\ &P_{n,1} + P_{2,3} = 2P \end{aligned} \tag{2.29}$$

Selon les valeurs de n, nous pouvons regrouper les équations (2.29) [1] :

– Si n n'est pas un multiple de 4, ce système donne :

$$P_{i,i+1} \;=\; P \quad \forall i \in [\![1..n]\!]$$

Ceci induit une dérivée nulle sur chacune des n courbes de liaison entre deux patches consécutifs (comme dans le cas à trois patches de la partie 2.3.2.1), et donc un point stationnaire.

– Si $n = 4$, on ne peut rien ajouter, les points sont « libres ».

– Si $n = 4p$ avec $p > 1$, nous avons $\forall i \in [\![1..4]\!]$:

$$P_{i,i+1} \;=\; P_{i+4,i+5} \;=\; \dots \;=\; P_{i+4(p-1),i+4(p-1)+1}$$

Les points égaux entre eux sont réunis en quatre groupes cycliques, répartis autour de P.
Deux cas sont possibles, le cas dégénéré où tous ces points de contrôle sont confondus avec P, ce qui ramène au cas précédent (point stationnaire). La deuxième possibilité est d'avoir une surface qui se replie sur elle-même, solution peu satisfaisante d'un point de vue applicatif. Dans la pratique, nous avons observé que le système de contraintes de reconstruction privilégiait le cas dégénéré.

2.3.2.2 Cas particulier de recollement C^1 qui fonctionne

Nous venons de voir qu'il n'y a qu'une seule configuration permettant d'obtenir un recollement C^1 convenable. Il s'agit du recollement C^1 de quatre patches centrés autour d'un point central.

Ce cas de figure permet uniquement de reconstruire des surfaces toriques : si l'on suppose que tous les patches sont placés dans cette configuration (comme sur la figure 2.7), nous avons un nombre f de faces, un nombre

1. Les indices sont compris entre 1 et n. Si un indice vaut $n + 1$, il sera considéré de manière cyclique comme valant 1

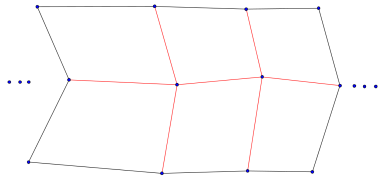


FIGURE 2.7 – Cas particulier de recollement C^1 de plusieurs patches où les conditions C^1 deux à deux entre les patches suffisent

a d'arêtes tel que $a = \frac{4f}{2} = 2f$, et une nombre s de sommets tel que $s = \frac{4f}{4} = f$. Ceci nous donne la caractéristique d'Euler et le genre de la surface :

$$\chi = s - a + f = f - 2f + f = 0 \tag{2.30}$$

$$g = 1 - \frac{\chi}{2} = 1 \tag{2.31}$$

On ne peut donc reconstruire de cette façon que des surfaces topologiquement équivalentes à un tore.

2.3.2.3 Recollement G^1 de plusieurs patches

Soient plusieurs patches $\mathbb{A}_i$ et leurs surfaces respectives $\mathtt{S}^{\mathbb{A}_i}$ ayant un sommet en commun représenté sur la figure 2.4. On étudie maintenant leur recollement G^1. Nous avons montré à la partie 2.3.1.3 qu'un recollement G^1 de deux patches pouvait s'exprimer sous la forme d'un programme linéaire, sous réserve que les coefficients α_i et β_i de chaque jonction soient fixés au préalable. Pour n patches, les contraintes linéaires G^1 de recollement (2.26) le long de chacune des arêtes de la figure 2.6 s'écrivent :

$$\begin{aligned}
&\beta_1 P + (1-\beta_1)P_{1,2} - \alpha_1 P_{n,1} - (1-\alpha_1)P_{2,3} = 0 \\
&\beta_2 P + (1-\beta_2)P_{2,3} - \alpha_2 P_{1,2} - (1-\alpha_2)P_{3,4} = 0 \\
&\dots \\
&\beta_n P + (1-\beta_n)P_{n,1} - \alpha_n P_{n-1,n} - (1-\alpha_1)P_{1,2} = 0
\end{aligned} \tag{2.32}$$

Ces contraintes peuvent être regroupées dans le système matriciel (2.33). Elles permettent de fixer les points P et $P_{i,i+1}$ entre eux. Attention, déterminer les coefficients α_i et β_i ne fixe pas la position des points. La position est déterminée dans le processus de reconstruction. Ces contraintes déterminent la configuration locale de ces points, à homothétie, rotation et translation près.

$$\begin{pmatrix} \beta_1 & (1-\beta_1) & -(1-\alpha_1) & & & & -\alpha_1 \\ \beta_2 & -\alpha_2 & (1-\beta_2) & -(1-\alpha_2) & & & \\ \vdots & & \ddots & \ddots & \ddots & & \\ \vdots & & & \ddots & \ddots & \ddots & \\ \beta_{n-1} & & & & -\alpha_{n-1} & (1-\beta_{n-1}) & -(1-\alpha_{n-1}) \\ \beta_n & -(1-\alpha_n) & & & & -\alpha_n & (1-\beta_n) \end{pmatrix} \times \begin{pmatrix} P \\ P_{1,2} \\ \vdots \\ P_{n,1} \end{pmatrix} = 0 \tag{2.33}$$

L'espace vectoriel des solutions contient la droite engendrée par la solution triviale $P = P_{1,2} = \cdots = P_{n,1} = 1$ c'est à dire le cas dégénéré où le point P est stationnaire. Le problème est donc de choisir des valeurs de α_i et β_i qui autorisent une autre solution (non triviale) à ce système de contraintes. Son existence est liée à l'existence d'une relation linéaire entre les lignes de la matrice de l'équation (2.33).

Nous avons développé et distingué deux possibilités : le recollement régulier, et le recollement choisi.

2.3.2.3.1 Recollement régulier Si l'on considère le cas équitable où tous les α_i sont égaux entre eux et tous les β_i aussi égaux, l'existence de cette solution contraint de choisir $\alpha_i = 1/2$ et $\beta_i = (1 - cos(\theta)) \quad \forall i \in [\![1..n]\!]$, avec $\theta = \dfrac{2\pi}{n}$. On a alors comme solution non triviale :

$$\begin{pmatrix} P \\ P_{1,2} \\ P_{2,3} \\ P_{3,4} \\ \vdots \\ P_{n,1} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ cos(\theta) \\ cos(2\theta) \\ \vdots \\ cos((n-1)\theta) \end{pmatrix}$$

Cette solution algébrique du système (2.33) fournit alors une configuration géométrique dans laquelle les points $P_{i,i+1}$ sont répartis en étoile (régulière) autour de P.

2.3.2.3.2 Recollement choisi Si, au contraire, on souhaite imposer une configuration géométrique spécifique comme sur la figure 2.8, nous devons déterminer les α_i et β_i à l'aide de données géométriques fournies préalablement.

En se donnant une carte de recollement comprenant la longueur des segments $[PP_{i,i+1}]$ et les angles θ_i, il est possible d'assigner des valeurs α_i et β_i de manière à exprimer les contraintes de recollement. On construit d'abord les points $I_{i,i+1}$, intersection du segment $[P_{i-1,i}P_{i+1,i+2}]$ et de la demi-droite $[PP_{i,i+1})$.

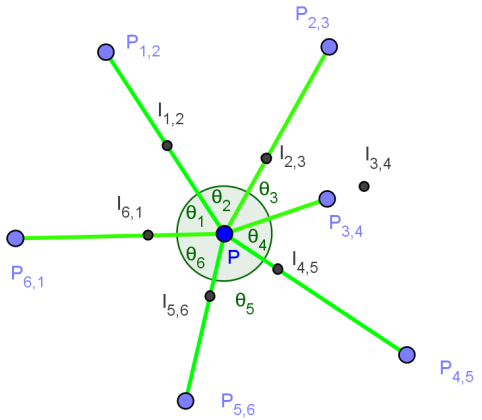


FIGURE 2.8 – Choix d'une configuration de recollement. Détermination des paramètres α_i et β_i

On obtient l'expression suivante pour α_i :

$$\alpha_i = \frac{I_{i,i+1}P_{i+1,i+2}}{P_{i-1,i}P_{i+1,i+2}}$$

En ce qui concerne β_i, il convient de distinguer deux cas : $I_{i,i+1} \in [PP_{i,i+1}]$ et $I_{i,i+1} \notin [PP_{i,i+1}]$

- Si $I_{i,i+1} \in [PP_{i,i+1}]$, alors $\beta_i = \dfrac{I_{i,i+1}P_{i,i+1}}{PP_{i,i+1}}$
- Si $I_{i,i+1} \notin [PP_{i,i+1}]$, comme pour le point $I_{3,4}$ de la figure 2.8 alors $\beta_i = \dfrac{-I_{i,i+1}P_{i,i+1}}{PP_{i,i+1}}$

Chapitre 3

Optimisation

L'optimisation est un domaine de recherche qui a été particulièrement étudié ces cinquante dernières années [HU02]. Un problème d'optimisation est souvent formulé de la manière suivante :

$$\begin{cases} \underset{x \in D}{Min} f(x) \\ \text{Sous contraintes } x \in C \end{cases} \tag{3.1}$$

La fonction f est une fonction à variable réelles, appelée communément fonction objectif ou bien fonction de coût, définie sur un domaine D. Le domaine C est un sous-domaine de D défini par un ensemble de contraintes imposées à x. Selon le problème initial, il peut être décrit sous plusieurs formes, et parfois peut être difficile à caractériser. Un exemple classique est la minimisation de la fonction (3.2) illustrée figure 3.1.

$$(x \sin x)^2 + 0,01|x|(cos(1000x) + 1) \tag{3.2}$$

Le minimum de cette fonction est 0, atteint lorsque $x = 0$. Cependant, elle présente tellement de variations aux alentours de 0, et tellement de minima locaux tous aussi proches de 0, que la recherche du minimum global peut s'avérer ardue. De plus, avec les incertitudes numériques qu'induisent les calculs répétitifs en flottants, le résultat sera très rapidement faussé.

A partir de la formulation (3.1), on doit se poser un certain nombre de questions :

- Existence d'une solution. C'est-à-dire est-ce que le domaine $C \subset D$ n'est pas réduit à l'ensemble vide ?
- Unicité de la solution. Dans le cas où $C \neq \emptyset$ et en nommant v le minimum de la fonction objectif f sur C, combien de $x_i \in C$ vérifient $f(x_i) = v$?
- Parmi les contraintes qui décrivent C, combien sont nécessaires à la description du problème ?
- Parmi les contraintes qui décrivent C, combien sont suffisantes ?

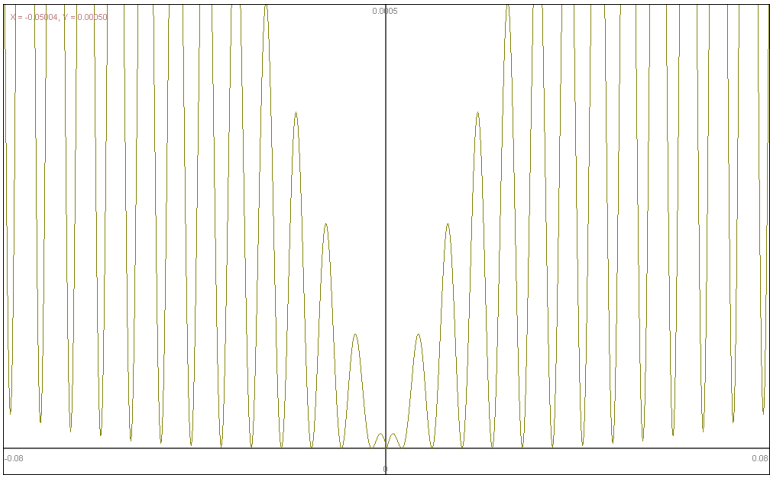


FIGURE 3.1 – Représentation graphique de la fonction (3.2)

– Algorithmes de calcul pour trouver x et trouver la valeur de la fonction objectif.
– Si le temps de calcul est prohibitif, certains algorithmes proposent d'arrêter le calcul au bout d'un temps prédéfini. Quelle est la fiabilité de la meilleure solution au moment de l'arrêt ? A-t-on un moyen de quantifier la qualité de la solution fournie (intervalle de sûreté, pourcentage d'erreur par rapport à la meilleure solution théorique, etc...) ?

On distingue deux types d'optimisation, l'optimisation sans contraintes, c'est à dire si $C = D$, et l'optimisation sous contraintes, $C \subsetneq D$.

3.1 Optimisation sans contraintes

L'optimisation sans contraintes est l'objet de plusieurs familles d'algorithmes. On trouve principalement les algorithmes de descentes et les algorithmes de région de confiance. La plupart de ces algorithmes nécessitent une certaine régularité de la fonction f ainsi que des propriétés sur C, ou bien une régularité sur g la fonction qui décrit le domaine C.

Parmi les algorithmes de descente, on trouve les méthodes du gradient, du gradient conjugué, l'algorithme de Newton, quasi-Newton, Newton modifié, etc... Le principe de ces algorithmes itératifs de minimisation consiste à établir une suite de valeurs x_i telle que x_{i+1} est dans une direction opposée à la valeur du gradient de la fonction en x. Ces algorithmes se différencient selon la manière de choisir cette direction opposée, et le choix du pas de convergence.

Il existe aussi les algorithmes de région de confiance, avec comme algorithme de référence l'algorithme de Levenberg-Marquardt [Lev44, Mar63], qui est de loin le plus utilisé.

3.2 Optimisation sous contraintes

Dans le cas d'optimisation sous contraintes, il faut distinguer deux cas :
- l'optimisation avec des contraintes égalité
- l'optimisation avec des contraintes inégalité

Dans les deux cas, on change le problème initial en un problème d'optimisation sans contraintes, que l'on peut résoudre (paragraphe précédent).

3.2.1 Contraintes égalité

Le problème est le suivant :

$$\begin{cases} \underset{x \in D}{Min} f(x) \\ g(x) = 0 \end{cases} \tag{3.3}$$

On va transformer le problème de manière à minimiser une autre fonction. En introduisant λ un vecteur contenant les multiplicateurs de Lagrange, le problème devient le suivant :

$$\begin{cases} \underset{x \in D}{Min} f(x) + \lambda g(x) \end{cases} \tag{3.4}$$

Le problème se résume ainsi à chercher les extréma de la fonction $H(x, \lambda) = f(x) + \lambda g(x)$ selon x. On en déduit alors un système d'équations équivalent au problème de minimisation de départ :

$$\nabla f(x) + \nabla g(x)^T \lambda = 0 \tag{3.5}$$

3.2.2 Contraintes inégalité

Le problème de minimisation avec contraintes inégalités est le suivant :

$$\begin{cases} \underset{x \in D}{Min} f(x) \\ g_i(x) \leq 0 \ \forall i \in [\![1..p]\!] \end{cases} \tag{3.6}$$

En procédant de même que précédemment, et en partant du principe que seules les contraintes « actives » jouent un rôle à l'optimum, on obtient un système d'équations dû à Kuhn-Tucker [KT50] :

$$\begin{cases} \nabla f(x) + \nabla g(x)^T \mu = 0 \\ \mu_i \geq 0 \text{ (grâce à } g_i \leq 0) \\ \mu_i g_i(x) = 0 \ (\mu_i = 0 \text{ si la contrainte n'est pas active}) \end{cases} \tag{3.7}$$

3.2.3 Méthodes de pénalité

On peut introduire des pénalités sur la fonction objectif. Ainsi, au lieu de minimiser $f(x)$ définie sous des contraintes définies par g, on va s'attarder à minimiser $f(x) + \sigma P(x, g)$ où P est la fonction de pénalité. Le terme σ représente l'importance de la pénalité vis-à-vis de la fonction objectif. Ce terme peut lui aussi faire partie de l'optimisation, par exemple dans un algorithme itératif, ou bien être fixé préalablement par une étude statistique, etc...

Dans le cas d'un problème avec des contraintes égalité (3.3), la pénalité peut être égale à $P(x, g) = ||g(x)||^2$. On dit alors qu'on applique une pénalité quadratique. On peut aussi choisir une norme différente, par exemple la norme l_1 ou la norme l_∞.

Dans le cas d'un problème avec des contraintes inégalité (3.6), la pénalité peut être ajoutée à la fonction objectif de plusieurs manières. Il est parfois préférable de forcer beaucoup plus les contraintes violées que les autres. On parle alors de pénalités barrières. Par exemple $P(x, g) = log(-g_i(x))$

3.2.4 Programmation linéaire

Un programme linéaire [CV83] est un problème d'optimisation tel que f est une fonction linéaire, à minimiser sur un domaine défini par des contraintes linéaires. On peut donc définir c, A et B de telle sorte que le problème s'écrive ainsi :

$$\begin{cases} \underset{x\in\mathbb{R}^n}{Min}\ c^T x \\ Ax \leq b \\ x \geq 0 \end{cases} \tag{3.8}$$

On peut écrire ce même problème sous sa forme standard :

$$\begin{cases} \underset{x\in\mathbb{R}^n}{Min}\ f^T x \\ Dx = e \\ x \geq 0 \end{cases} \tag{3.9}$$

On peut aussi définir un problème équivalent, appelé problème dual (terme à opposer au problème primal) :

$$\begin{cases} \underset{x\in\mathbb{R}^p}{Max}\ e^T y \\ D^T y + s = f \\ s \geq 0 \end{cases} \tag{3.10}$$

D'un point de vue géométrique, le problème de programmation linéaire (3.8) est la recherche d'un élément x de $\mathbb{R}^n$ qui ait la plus grande valeur de la fonction objectif, et qui de plus soit dans le domaine défini par les

contraintes $Ax \leq b$. Ces contraintes forment un polyèdre (parfois vide) de $\mathbb{R}^n$. Ce polyèdre définit les valeurs acceptables du programme linéaire. La fonction objectif peut être perçue comme un vecteur définissant la direction dans laquelle se trouve la meilleure solution. Ainsi, toujours d'un point de vue géométrique, la résolution d'un programme linéaire consiste à trouver, dans le polyèdre des solutions acceptables, le point x qui sera « le plus loin » dans la direction d'optimalité.

Il existe aussi d'autres formes de programmes linéaires, un peu plus compliquées et ayant moins de propriétés. On citera par exemple les programmes linéaires à nombres entiers. Ce sont des programmes linéaires identiques aux précédents, sauf que tout ou partie des variables sont des variables de type entier. Parmi ce type de problème, on trouve aussi des programmes linéaires décisionnels, où les variables sont booléennes.

Les algorithmes de résolution des programmes linéaires sont nombreux, mais deux d'entre eux sortent du lot.

- L'algorithme du simplexe est le plus connu et certainement le plus étudié. Il se détache des autres algorithmes d'optimisation en adoptant un point de vue géométrique. Malgré sa complexité théorique exponentielle (atteinte lors de cas particulièrement pathologiques), il est relativement rapide. Il existe différentes versions de cet algorithme : simplexe [DOW55], simplexe tableau [BR73], simplexe dual [AMO93], etc... La recherche de l'optimum se décompose en deux étapes : recherche d'une solution réalisable, recherche de la solution optimale, chacune de ces deux étapes nécessitant le même algorithme du simplexe.
- L'algorithme des points intérieurs [Kar84] quant à lui est un algorithme d'optimisation tel que nous l'avons évoqué dans la partie précédente. C'est un algorithme de pénalité prenant en compte une pénalité barrière. Contrairement à l'algorithme du simplexe, qui donne à chaque étape une solution sur le bord du polyèdre, cet algorithme parcourt l'intérieur du polyèdre, d'où son nom.

Il existe des logiciels et bibliothèques de calcul permettant de résoudre un programme linéaire. Les plus reconnus sont Lpsolve [Ber08], Soplex [Wun97, Wun96], GLPK [Mak04] et CGAL [FGK+00, cga08]. Parmi les logiciels payants, on ne citera que CPLEX, outil de la société ILOG, qui domine largement le marché. Celui-ci permet non seulement de résoudre les programmes linéaires, mais aussi les programmes linéaires en nombres entiers, les programmes linéaires mixtes, ou encore permettant de prendre en compte les fonctions objectifs quadratiques. De plus, il fournit un algorithme de résolution permettant de passer du simplexe, à l'algorithme des points intérieurs, selon le type de problèmes, et selon les difficultés numériques rencontrées au cours de la résolution.

Chapitre 4

Interpolation et Fitting

4.1 Principe

Le fitting et l'interpolation sont deux techniques qui servent à décrire des données brutes sous une forme plus descriptive (par exemple une courbe, si l'on considère que les données se présentent sous la forme d'un ensemble de points).

- L'interpolation consiste à construire une courbe, passant exactement par les données.
- Le fitting, qui s'apparente à de l'approximation, consiste à construire une courbe d'un certain type (polynôme, gaussienne, courbe de Bézier, spline, etc...) approchant des données. Le but est d'ajuster les paramètres de la courbe de façon à obtenir la « meilleure » approximation possible selon un critère qui doit être précisé au préalable.

Dans l'idéal, l'interpolation est préférable car les données ne sont pas altérées, mais excepté dans un cas dégénéré, le nombre de descripteurs de la courbe est nécessairement du même ordre que la taille des données initiales.

De multiples raisons peuvent conduire à utiliser moins de descripteurs. L'interpolation devient alors le plus souvent impossible et on doit se contenter d'une approximation par une méthode de fitting.

La différence majeure entre ces deux techniques réside dans le nombre de descripteurs du résultat final. À moins d'avoir des données initiales corrélées, il faut au moins autant de descripteurs que de données pour l'interpolation, tandis que le fitting en nécessite moins.

Parmi les techniques les plus connues, nous connaissons la régression linéaire (fitting), l'interpolation polynomiale (interpolation), etc...

4.1.1 Centre, médiane et moyenne

Considérons qu'on veuille approcher une famille A de nombres de cardinal $n \in \mathbb{N}^*$ par un seul nombre m (non nécessairement issu de A). En général, on appelle le nombre m la moyenne de A et, contrairement aux idées reçues,

on peut lui donner différentes valeurs. Par exemple, considérons la famille $A = \{1, 2, 2, 3, 5, 7, 10\}$.

4.1.1.1 La moyenne arithmétique

C'est la moyenne que tout le monde connait.

$$m_{\text{arith}} = \frac{\sum_{a \in A} a}{n} \tag{4.1}$$

Exemple : $m_{\text{arith}} = \frac{30}{7} \approx 4,285714286$

Nous pouvons remarquer que ce nombre minimise l'écart quadratique de la moyenne aux nombres de A. C'est à dire que m_{arith} est tel que :

$$m_{\text{arith}} = \underset{m \in \mathbb{R}}{\operatorname{argmin}} \sum_{a \in A} (a - m)^2$$

4.1.1.2 La moyenne arithmétique pondérée

Cette moyenne suppose que les nombres de A soient munis de poids λ

$$m_{\text{arith pondérée}} = \frac{\sum_{a \in A} \lambda_a a}{\sum_{a \in A} \lambda_a} \tag{4.2}$$

C'est une moyenne bien connue des lycéens, puisque cette moyenne avec coefficients est celle qui permet de calculer la moyenne du baccalauréat.

Si tous les poids valent la même valeur non nulle, cette moyenne est égale à la moyenne arithmétique.

4.1.1.3 La moyenne géométrique

Cette moyenne est méconnue, mais se substitue à la moyenne arithmétique dans certains cas où cette dernière n'aurait aucun sens.

$$m_{\text{geom}} = \sqrt[n]{\prod_{a \in A} a} \tag{4.3}$$

Exemple : $m_{\text{geom}} = \sqrt[7]{4200} \approx 3,293116095$

4.1.1.4 La médiane

La médiane d'un groupe de nombre est un nombre tel qu'il y ait dans le groupe autant de valeurs supérieure ou égale à ce nombre, que de valeurs inférieures ou égales.

Pour déterminer la médiane, on trie le groupe de nombre. Dans le cas où n est impair, la valeur au rang $\frac{n+1}{2}$ est la médiane. Dans le cas où n est pair, n'importe quelle valeur entre celle de rang $\frac{n}{2}$ et $\frac{n}{2}+1$ fait l'affaire, mais en général, la moyenne arithmétique de ces deux valeurs est prise.

Exemple : $m_{\text{mediane}} = 3$

4.1.1.5 Le centre

Le centre est la demi somme des valeurs maximales et minimales des nombres de A.

$$m_{\text{centre}} = \frac{\max_{a \in A} a + \min_{a \in A} a}{2} \tag{4.4}$$

Exemple : $m_{\text{centre}} = \frac{11}{2}$

Nous pouvons remarquer que ce nombre minimise l'écart uniforme de la moyenne aux nombres de A. C'est à dire que m_{centre} est tel que $\max_{a \in A}(a - m_{\text{centre}})$ est minimal.

4.1.2 Interpolation polynomiale

On s'intéresse désormais à des données de dimension deux. Étant donné un ensemble de n points (x_i, y_i) $\forall i \in [\![1..n]\!]$, l'interpolation polynomiale consiste à calculer un polynôme P dans un espace de polynômes donné tel que $\forall i$ $P(x_i) = y_i$. La dimension de l'espace de polynômes doit être supérieure ou égale au nombre de points si on veut garantir l'existence d'un résultat. Si la dimension de l'espace de polynômes excède n ; on voit apparaître des oscillations parasites. Certains coefficients sont indéterminés comme si l'on veut par exemple interpoler (-1,1) et (1,1) par un polynôme de degré 2, la solution est de la forme $(1 - \lambda) + 0x + \lambda x^2$.

4.1.3 Régression linéaire

Soit (x_i, y_i) $i \in [\![1..n]\!]$ un ensemble de n points de $\mathbb{R}^2$. On suppose que ces points ont été mesurés lors d'une expérience, et que l'on veuille représenter cet ensemble de points par une droite affine d'équation $y = ax + b$.

La régression linéaire consiste à trouver des paramètres a^* et b^* tels que $y = a^*x + b^*$ soit le plus proche possible des données mesurées : la droite de régression de y en x minimise la somme des carrés des écarts verticaux à la droite.

$$(a^*, b^*) = \underset{a,b \in \mathbb{R}}{argmin} \sum_i (y_i - (ax_i + b_i))^2 \tag{4.5}$$

On peut noter que cette droite passe par le point moyen $(\overline{x}, \overline{y})$, où $\overline{x}$ est la moyenne arithmétique des x_i. La covariance σ_{xy} est définie par la formule suivante :

$$\sigma_{xy} = \frac{\sum_{i=1}^{n} x_i y_i}{n} - \overline{xy} \tag{4.6}$$

On vérifie a posteriori la cohérence linéaire des données en calculant le coefficient de corrélation linéaire r, qui permet de comparer les deux droites de régression $y = ax + b$ et $x = a'y + b'$.

$$r = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{4.7}$$

Le coefficient de relation linéaire mesure ainsi la dispersion du nuage de point approximé, et permet de déterminer si la régression est pertinente :

- si $|r| = 1$, les deux droites sont confondues, et l'ajustement est exact.
- si $|r| \geq \frac{\sqrt{3}}{2}$, l'ajustement est considéré comme valide.
- sinon, la regression linéaire n'est pas pertinente.

4.2 Fitting : position du problème

Le fitting consiste à considérer l'équation matricielle générale $A \times X = B$, A et B étant donnés. A est une matrice de dimensions $n \times m$ et B est un vecteur de dimension n. L'inconnue est X, un vecteur de dimension m.

4.2.1 Terme d'erreur

Si B n'est pas dans l'image de A (figure 4.1), il n'existe pas de solution exacte à cette équation (il n'existe pas de solution X telle que $A \times X = B$), mais rien n'empêche de chercher une solution approchée, de préférence la meilleure possible.

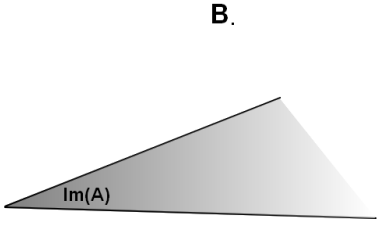


FIGURE 4.1 – Il n'existe pas de solution X telle que $A \times X = B$

Nous introduisons le terme d'erreur de reconstruction lié à l'approximation. Étant donné un X possible, l'erreur d'approximation commise est

supportée par le vecteur $E = A \times X - B$. Ce terme E est un vecteur dont chacune des composantes est l'erreur commise lors de l'approximation de B par $A \times X$ 4.2.

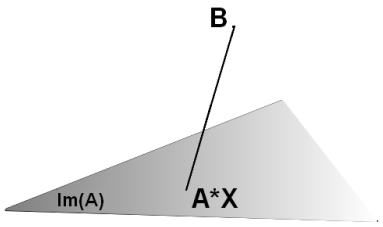


FIGURE 4.2 – Erreur de reconstruction introduite par un vecteur X

4.2.2 Reconstruction par moindres carrés (Least Square Fitting)

La méthode la plus couramment utilisée (Least Square Fitting [AW04, WR67]) consiste à rechercher une solution approchée X dont la norme euclidienne $||E||_2$ du vecteur d'erreur 4.2.1 est minimale. Ce problème de minimisation se ramène à la résolution d'un système linéaire.

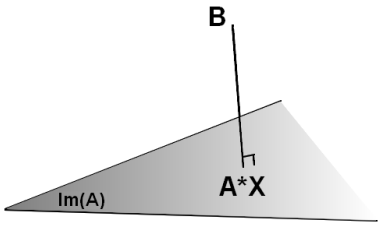


FIGURE 4.3 – Least square fitting

Géométriquement, la solution aux moindre carrés est le vecteur X tel que $A \times X$ est le projeté orthogonal de B sur $Im(A)$. X est obtenu par le calcul suivant (4.8) :

$$X = (A^T \times A)^{-1} \times A^T \times B \tag{4.8}$$

4.2.3 Reconstruction par programmation linéaire (LP Fitting)

Le principe de la méthode LP Fitting [MGM06], plus originale que le Least Square Fitting 4.2.2, est de chercher une solution approchée X dont la norme uniforme $||E||_\infty$ du vecteur d'erreur est minimale.

Au lieu de minimiser la norme euclidienne 4.2.2, on minimise donc la norme infinie du vecteur d'erreur E. Cela revient à minimiser chacune des

coordonnées de ce vecteur et, en les encadrant par $-h$ et h, de former un programme linéaire. Ce programme linéaire peut être résolu grâce à un solveur de programmes linéaires.

$$
\begin{aligned}
& \left\{ \min_X \left(||E||_\infty \right) \right. \\
\iff & \left\{ \min_X \left(\max_i |E_i| \right) \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,h}(h) \\ -h \leq E_i \leq +h \ \ \forall i \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,h}(h) \\ -h * \mathbb{1} \leq A * X - B \leq h * \mathbb{1} \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,h}(h) \\ -h * \mathbb{1} \leq A * X - B \\ A * X - B \leq h * \mathbb{1} \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,h}(h) \\ A * X + h * \mathbb{1} \geq B \\ A * X - h * \mathbb{1} \leq B \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,h}(h) \\ -A * X - h * \mathbb{1} \leq -B \\ A * X - h * \mathbb{1} \leq B \end{array} \right.
\end{aligned}
$$

En utilisant un principe similaire, la programmation linéaire permet ainsi de minimiser la norme $||E||_1$ du vecteur d'erreur :

$$
\begin{aligned}
& \left\{ \min_{X} \left(||E||_1 \right) \right. \\
\iff & \left\{ \min_{X} \left(\sum_i |E_i| \right) \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,h_i} (\sum_i h_i) \\ -h_i \leq E_i \leq +h_i \ \ \forall i \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,H} (\sum_i h_i) \\ -H \leq A * X - B \leq H \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,H} (\sum_i h_i) \\ -H \leq A * X - B \\ A * X - B \leq H \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,H} (\sum_i h_i) \\ A * X + H \geq B \\ A * X - H \leq B \end{array} \right. \\
\iff & \left\{ \begin{array}{l} \min_{X,H} (\sum_i h_i) \\ -A * X - H \leq -B \\ A * X - H \leq B \end{array} \right.
\end{aligned}
$$

Deuxième partie

Reconstruction de surfaces

Chapitre 5

Principe de la reconstruction de surfaces

5.1 Introduction

Soit $(M_i)_{1 \leq i \leq p}$ un ensemble de points 3D. La reconstruction de surfaces est la recherche d'une surface qui approche au mieux ce nuage de points. Il existe de nombreux types de surfaces, surfaces triangulaires, surfaces de subdivision, surfaces implicites et surface paramétrées pour ne citer que les plus utilisées. Il est possible de regrouper ces différentes notions dans le diagramme figure 5.1.

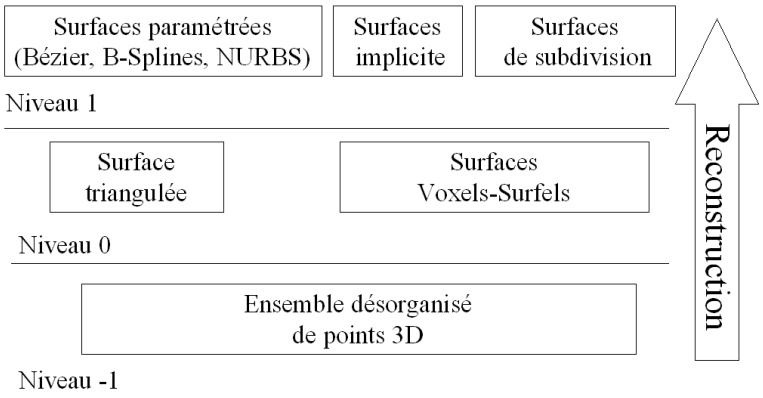


FIGURE 5.1 – Les différents niveaux de reconstruction

Au niveau -1 se trouvent les surfaces représentées par un ensemble de points désorganisés, ou un ensemble de points munis de normales. Ce sont des modèles primitifs non continus.

Le niveau 0 est celui des premiers modèles continus :

– Un maillage comprenant l'ensemble des points de départ, et constitué de polygones. On trouve principalement les maillages triangulaires, ou

quadrangulaires (plus favorables à l'application de schémas de subdivision).

– Un ensemble contigu de voxels, ou un ensemble de surfels [**?**].

Le principal défaut des surfaces de niveau 0 est leur manque de régularité. On introduit au niveau supérieur les surfaces lisses. Le niveau 1 rassemble ainsi les surfaces C^1 et G^1. On y trouve entre autres les surfaces paramétrées (surfaces de Bézier, B-Splines, NURBS, etc...), les surfaces implicites et les surfaces de subdivision.

On peut étendre cette classification aux surfaces de plus grande régularité (niveau 2 pour les surfaces C^2 et G^2, etc...).

La reconstruction de surfaces se résume le plus souvent à passer d'un niveau à un niveau supérieur. Il existe de nombreuses méthodes de reconstruction de surfaces, et les recherches sont nombreuses sur le sujet [ABK98, BBX95, BC00, BDH96, CRE01, Guo97, HDD+92, HS89, Jüt97, LS86, OF03, RT95, VMA86].

5.2 Reconstruction de surfaces paramétrées

Parmi toutes les variantes possibles de reconstruction de surfaces, nous nous intéressons au problème de reconstruction qui consiste à prendre en entrée une surface modélisée par un ensemble désorganisé de points (niveau -1) ou un maillage (niveau 0), et à fournir une ou des surfaces paramétrées (niveau supérieur ou égal à 1).

Rappelons la forme des surfaces paramétrées considérées :

$$\begin{array}{rccc} \mathtt{S}: & X \subset \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & \sum_{i=1}^{N} P_i * f_i(s,t) \end{array}$$

Le problème de reconstruction en une seule surface paramétrée est le suivant :

Données Soit $(M_i)_{1\leq i\leq p}$ un ensemble de points 3D (et éventuellement $\mathcal{M}$ le maillage associé). Soit $(f_i)_{1\leq i\leq N}$ une base de fonctions donnée.

Résultat Le résultat de l'algorithme sont les points de contrôle de la surface paramétrée $\mathtt{S}(s,t) = \sum_{i=1}^{N} P_i * f_i(s,t)$ passant au plus près du nuage de points $(M_i)_{1\leq i\leq p}$.

L'idéal serait de calculer une surface qui passe exactement par l'ensemble de points initial. Pour les raisons évoquées partie 4.1.2, l'interpolation nécessite autant de points de contrôle que de points initiaux, ce qui n'est pas toujours satisfaisant.

Plutôt que de chercher une surface d'interpolation, on est conduit à se contenter le plus souvent d'une approximation (fitting). Cependant, cet objectif reste délicat : dans le cas d'une surface non homéomorphe à un disque, ou d'une surface complexe, il sera souvent impossible d'approcher correctement l'ensemble initial (avec un nombre de points de contrôle limité à priori). Dans cette situation courante, une seule surface n'est pas suffisante pour obtenir une approximation satisfaisante. Il faut en utiliser plusieurs et c'est la réunion de ces surfaces appelées patches qui forme la surface finale. La régularité de cette surface finale dépend naturellement de la façon dont sont recollés les patches les uns avec les autres (C^0, C^1, G^1, etc...). La résolution du problème d'approximation doit donc tenir compte des contraintes liées à l'approximation ainsi que des contraintes de recollement décrites en partie 2.3.2.

Ce problème général de reconstruction a déjà été largement étudié [EH96, WARV02, SM91, BBCS99] depuis 25 ans dans le cadre de problématiques de reverse engineering. La diversité des méthode proposées est telle qu'un inventaire exhaustif serait fastidieux. Nous suivrons la méthodologie introduite par Eck & Hoppe dans [EH96]. Cette méthode comporte trois ou quatre étapes selon le modèle initial :

0. *Triangulation*

   Si la surface initiale est donnée par un ensemble de points (M_i), on calcule un maillage $\mathcal{M}$ dont les points M_i sont les sommets (voir section 1.3).

   Si la donnée est directement le maillage $\mathcal{M}$, cette étape est inutile.

1. *Quadrangulation*

   L'approximation des données initiales par une seule surface n'étant pas toujours satisfaisante, il convient de partitionner le maillage $\mathcal{M}$. Ce partitionnement est appelé segmentation. Cependant, afin de pouvoir recoller chaque partie d'une manière convenable, la segmentation doit vérifier un certain nombre de contraintes topologiques qui en font une quadrangulation [BMRJ04, DBG$^+$06, TACSD06]. Un quadrangle est un sous ensemble connexe du maillage $\mathcal{M}$ homéomorphe à un disque, et tel que la frontière est formée de quatre points particuliers (appelés coins) et de quatre bords reliant ces coins. Cette quadrangulation produit des quadrangles $\mathcal{Q}_k$ auxquels on interdit de se joindre en T : un coin ne doit pas être sur un bord d'un autre quadrangle (voir figure 6.15 section 6.4.1). Cette partie sera détaillé en section 6.4.

2. *Paramétrisation*

   La paramétrisation [FH05, HLS07, SPR06] s'applique à chaque quadrangle $\mathcal{Q}_k$. On associe à chaque point M_i^k d'un quadrangle $\mathcal{Q}_k$, une image dans $[0,1] \times [0,1]$. Autrement dit, le principe est de calculer un plongement linéaire par morceau du quadrangle $\mathcal{Q}_k$ dans $[0,1]^2$. De

nombreuses propriétés, telles que la préservation des angles, des longueurs, des aires peuvent être considérées dans ce cadre, car on cherche à avoir aussi peu de déformations que possible (voir section 7).

3. *Fitting*

   Enfin, le fitting [MGM06, Far02] est l'étape de transformation du nuage de points paramétrisé, en un ensemble de surfaces paramétrées. Cette étape va déterminer les points de contrôle des surfaces $\mathtt{S}_k$ approximant les nuages de points , en prenant en compte les conditions souhaitées de recollement et de régularité de la surface. Cette étape revient à minimiser globalement ou indépendamment sur chaque quadrangle l'écart entre les données M_i^k et le point sur la surface reconstruite $\mathtt{S}_k(s_i, t_i)$.

Chapitre 6

Segmentation et quadrangulations

La segmentation est un terme issu de l'analyse d'images. Le principe de la segmentation est de séparer et regrouper un ensemble de données en plusieurs groupes distincts. La segmentation d'images a fait l'objet de beaucoup d'études, en se basant sur plusieurs techniques, entre autres les techniques basées régions ou basées frontières [FMR+02]. On peut citer la croissance de régions, les algorithmes à base de lignes de partage des eaux ou les contours actifs.

Si le terme de segmentation d'images est bien connu des personnes travaillant en analyse d'images, il existe cependant d'autres types de segmentations. La segmentation de maillages en fait partie.

6.1 Segmentation de maillages

La segmentation de maillages a pour but de partitionner le maillage en morceaux de surfaces. Les principales approches de segmentations sont le découpage en carreaux surfaciques et le découpage en partie significatives [DBC07].

La segmentation en carreaux surfaciques rassemble les zones du maillage ayant les mêmes caractéristiques géométriques. Pour cela, on utilise les données issues de la surface : la courbure, la convexité de certaines zones du maillage ou bien l'approximation de certaines zones par des primitives telles que des plans, des sphères ou des cylindres.

La segmentation en parties significatives se rapporte plus à ce que l'œil et le cerveau humain peuvent faire naturellement : découper les zones. Par exemple, sur un modèle d'humanoïde, le cerveau a tendance à séparer les membres et la tête du reste du corps. Pour cela, non seulement les caractéristiques surfaciques sont analysées, mais aussi les caractéristiques volumiques.

Ainsi, en plus des caractéristiques de courbure et de convexité, il faut analyser le squelette et les extrémités (protrusions).

6.2 Découpage d'un maillage en une surface homéomorphe à un disque

Ce travail est un travail commun au LAIC et au LIMOS, avec la collaboration de Jean-Marie Favreau, Vincent Barra et Yan Gerard.

Considérons le problème de recouvrement d'un maillage par une texture [Cay03, SH02], ou le problème de paramétrisation (voir section 7). La seule donnée du maillage n'est pas suffisante pour satisfaire ces applications. En effet, pour pouvoir plaquer une texture sur un objet quelconque représenté par un maillage, il faut que l'on puisse identifier son maillage à une surface homéomorphe à un disque et faire une correspondance entre les coordonnées de texture et le point correspondant sur l'objet. Cela nécessite en général de découper avant de le plonger dans le plan. Cette étape bien connue des infographistes, est désignée par le terme UV-mapping, où u et v sont les coordonnées dans le plan.

Le problème de découpage est difficile, et s'il fait appel au savoir faire des infographistes, c'est parce qu'il n'a pas encore été résolu automatiquement de façon satisfaisante. En effet, il n'y a pas de manière optimale de découper, chaque algorithme donnant des solutions souvent équivalentes, mais très différentes. Plus précisément, si les découpages se font en fonction de la longueur des arêtes du maillage, Erickson et Har-Peled [EHP04] ont montré que le calcul de la coupe minimale d'un maillage était un problème NP-difficile. Ils se placent dans cette optique pour parler de découpage de surface en une surface homéomorphe à un disque. Leur algorithme heuristique, de complexité polynomiale, consiste à réduire le genre de la surface à zéro et ensuite à relier tous les bords de la surface obtenue entre eux afin d'obtenir une surface homéomorphe à un disque. Le vocabulaire, les définitions et les algorithmes seront détaillés dans la section 6.3.2.2.

Données : une triangulation $\mathcal{M}$ non homéomorphe à une sphère
tant que *le genre de M n'est pas nul* **faire**
  Trouver c le plus court lacet valide non séparant ;
  Couper $\mathcal{M}$ selon c ;
tant que *le nombre de bords de* $\mathcal{M}$ *n'est pas 1* **faire**
  Trouver c le plus court chemin de $\mathcal{M}$ entre deux bords;
  Couper $\mathcal{M}$ selon c ;

Algorithme 1 : Algorithme global de découpage

On notera qu'en pratique, le découpage se résume à dédoubler les arêtes et les sommets le long de la découpe, pour que localement, le voisinage de

chaque arête soit ainsi homéomorphe à un demi plan.

On nomme les arêtes de la triangulation $\mathcal{M}$ de manière unique selon un alphabet, et on donne une orientation à l'arête. Comme le décrit Hétroy [Het03], le découpage donne l'information de recollement, y compris le sens de recollement (on définit la lettre a si le sens de parcours est celui de la triangulation, sinon, on définit a^{-1}, si l'arête est parcourue dans le sens opposé). Ainsi, sur la triangulation découpée, homéomorphe à un disque, on relève le mot w représentant le bord. On peut toujours modifier le mot obtenu en regroupant les arêtes, afin de les identifier comme des chemins simples. À partir de là, comme décrit dans [FK97], on obtient le théorème suivant :

Théorème 3 *Toute surface compacte connexe peut être représentée sous la forme d'un polygone plan W, dont les arêtes sont orientées et marquées par des lettres (on marque a^{-1} l'arête a parcourue dans le sens contraire). La suite des lettres représentant les arêtes successives du bord de W forme un mot w qui a l'une des trois formes suivantes :*

- $w = aa^{-1}$
- $w = a_1 b_1 a_1^{-1} b_1^{-1} a_2 b_2 a_2^{-1} b_2^{-1} ... a_g b_g a_g^{-1} b_g^{-1}$
- $w = c_1 c_1 c_2 c_2 ... c_g c_g$

Définition Le polygone W obtenu est appelé schéma polygonal canonique de la surface.

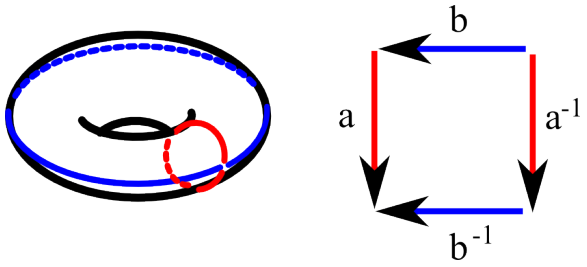


FIGURE 6.1 – Schéma polygonal canonique : A gauche, le tore découpé, à droite, le polygone plan le représentant.

6.3 Théorie pour le pavage de surface

Le découpage et le pavage de surface sont deux théories reliées aux segmentations de maillages. Dans cette section, nous allons les mettre en relation. Afin d'expliquer plus en détail le pavage de surfaces, nous devons introduire un cadre un peu plus général que celui abordé jusque là. En effet, si le pavage de surface parait simple à définir, le procédé utilisé nécessite un cadre théorique adapté.

6.3.1 Cellules et pavages

6.3.1.1 Maillage initial

Définition Un 2-*mesh* $\mathcal{M}$ est un complexe simplicial ayant la propriété d'être une 2-*variété connexe compacte (avec ou sans bord).* $\mathcal{M}$ est l'union de 0-simplexes appelés sommets, de 1-simplexes appelés arêtes et de 2-simplexes appelés triangles, et est localement homéomorphe à un disque ou un demi disque.

6.3.1.2 Cadre topologique : notion de 2-m-cellule

En informatique graphique, il est courant de considérer des cellules comme des simplexes, ou des polygones convexes car ils ont une structure bien connue avec des faces, des arêtes, des sommets. Dans ce contexte, un complexe cellulaire est un ensemble de cellules (des polyèdres) qui satisfont les deux conditions vues à la section 1.3.1 :

1. Chaque face d'une cellule est incluse dans le complexe cellulaire.
2. L'intersection de deux cellules est une cellule.

Cependant, ces complexes cellulaires, ainsi que les CW-complexes [Whi49] sont insuffisants pour ce cadre. Les cellules que nous sommes susceptibles de manipuler peuvent ne pas être homéomorphes à un disque, et peuvent être par exemple des cylindres. Nous allons donc nous baser sur les complexes cellulaires abstraits [Kov89].

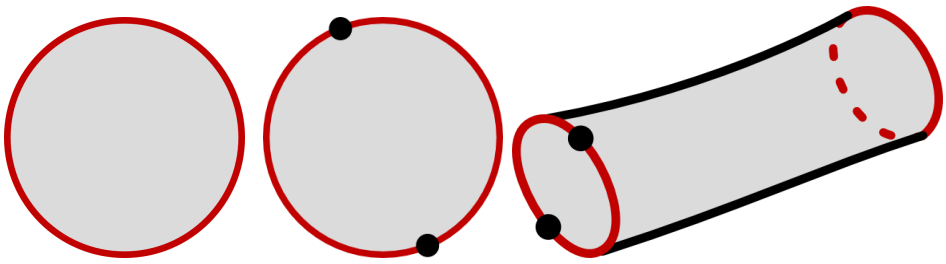
Un complexe cellulaire abstrait est un ensemble d'éléments abstraits appelés cellules, chaque cellule ayant une dimension et étant soumise à une relation d'ordre partiel permettant de dire si une cellule est sur le bord d'une autre. Dans le contexte de pavage, si nous considérons des variétés connexes compactes comme cellules, les notions de bord sont induites. Nous pouvons donc nous placer dans le contexte des variétés connexes compactes (avec ou sans bord), il nous reste à décrire correctement les deux conditions 1 et 2 que doivent remplir les m-cellules que nous définissons.

Définition Un complexe m-cellulaire $\mathcal{C}$ (de dimension inférieure ou égale à deux) est un ensemble de m-cellules avec les propriétés suivantes :

- (condition de dimension) Les 2-m-cellules sont des 2-variétés compactes, connexes à bord, les 1-m-cellules sont des courbes simples (ouvertes ou closes), et les 0-m-cellules sont des points.
- (condition de bord) Le bord d'une i-m-cellule est un $(i-1)$-sous-complexe, avec $i = \{1, 2\}$
- (condition de cohérence) L'intersection de deux m-cellules est une m-cellule.

Le deuxième point assure que les bords des 2-m-cellules soient bien définis par une structure (figure 6.2) composée d'une séquence de 1-m-cellules

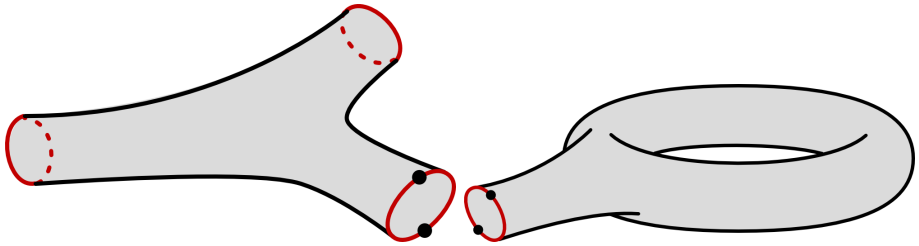
(arêtes) et de 0-m-cellules (sommets). On nommera de manière intuitive face d'une i-m-cellule les $i-1$-m-cellules incluses dans son bord. De même, le corps d'une i-m-cellule est celle-ci privée de son bord.



(a) Une 2-m-cellule avec un bord décrit par une 1-m-cellule fermée (il n'y a qu'une arête)

(b) Une 2-m-cellule avec un bord ayant deux 1-m-cellules et deux 0-m-cellules

(c) Une 2-m-cellule avec deux bords, décrits par trois 1-m-cellules et deux 0-m-cellules



(d) Une 2-m-cellule de genre 0, avec trois bords, quatre 1-m-cellules et deux 0-m-cellules. Deux 1-faces n'ont pas de 0-faces.

(e) Une 2-m-cellle de genre 1, avec un bord décrit par deux 1-m-cellules

FIGURE 6.2 – Quelques exemples de 2-m-cellules. Leurs bords sont décrits par une structure de 1-complexes.

Les polyèdres non dégénérés de dimension 2 sont des 2-m-cellules. Par la suite, un polygone avec n côtés sera appelé n-polygone.

Comme les 2-m-cellules sont des 2-variétés, elles ont leur propre topologie. L'intérieur d'une i-m-cellule est définie comme l'ensemble, privé des $(i-1)$-faces.

6.3.1.3 Tuiles et pavages

Une M-tuile est définie par un plongement d'une 2-m-cellule vers un 2-mesh.

Définition On considère les plongements $F : C \longrightarrow \mathcal{M}$ d'une 2-m-cellule C dans un 2-mesh $\mathcal{M}$ vérifiant les propriétés suivantes :

- F est continue.
- Soit v un sommet de C. Alors $F(v)$ est un 0-simplexe de $\mathcal{M}$.
- Soit e une arête de C. Alors $F(e)$ est un ensemble de 1-simplexes de $\mathcal{M}$.
- Soit b le corps de C. Alors $F(b)$ est un ensemble de 2-simplexes de $\mathcal{M}$.
- Un plongement est habituellement injectif, mais nous autorisons dans ce cadre que deux sommets ou deux arêtes puissent avoir les mêmes images : F ne devra être injective que sur l'intérieur de C. [1]

On appelle M-tuile la classe d'équivalence de tels plongements, où on définit que F et F' sont équivalents si et seulement si pour toute cellule X de C, $F(X) = F'(X)$.

Pour des raisons de lourdeur de notation, on définit une M-tuile par une de ces instances $(C, \mathcal{M}, F)$.

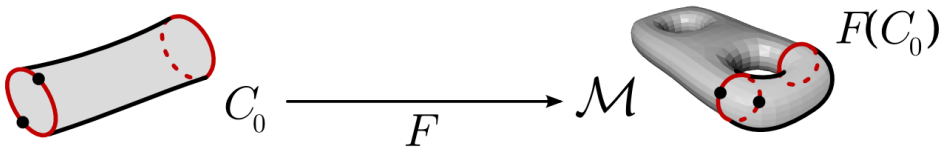


FIGURE 6.3 – Une M-tuile composée de la 2-m-cellule 6.2(c) et d'un plongement F sur un 2-mesh de genre 2.

Une M-tuile $(C, \mathcal{M}, F)$ est complètement décrite par la 2-m-cellule C et l'image par F de C, ses arêtes et ses sommets.

On pourrait se contenter d'associer uniquement les images sur $\mathcal{M}$ de chaque sommet, arête, face. Cependant, cette représentation n'est pas suffisante et ne permet pas de garantir la cohérence topologique (ce que permet l'existence de F satisfaisant les conditions de la définition 6.3.1.3).

Lorsque C est un n-polygone, la M-tuile est une tuile au sens classique du terme. La figure 6.3 montre un exemple de M-tuile.

Définition Un *M-pavage* d'un 2-mesh $\mathcal{M}$ par un complexe m-cellulaire $\mathcal{C}$ est la classe d'équivalence de plongements $F : \mathcal{C} \longrightarrow \mathcal{M}$ tels que :

- Pour tout 2-m-cellule C de $\mathcal{C}$, $F_{|C}$ est une M-tuile.
- La restriction de F à l'union des intérieurs des m-cellules qui ne sont pas des bords doit être injective. [2] De même, on suppose que si deux

1. Plus précisément, nous autorisons les sommets du bord de C à fusionner : ces deux sommets peuvent avoir la même image par F. On suppose aussi que si deux points x et y respectivement à l'intérieur des arêtes e et e' ont la même image, alors les arêtes entières e et e' ont aussi la même image $(F(e) = F(e'))$.
2. Il est ainsi impossible de recouvrir l'intérieur des m-cellules.

points x et y respectivement à l'intérieur des arêtes e et e' ont la même image, alors les arêtes entières e et e' fusionnent, avec $F(e) = F(e')$. [1]

– F est surjective.

Deux plongements F et F' sont équivalents si et seulement si pour toute i-m-cellule X du complexe m-cellulaire $\mathcal{C}$, $F(X) = F'(X)$.

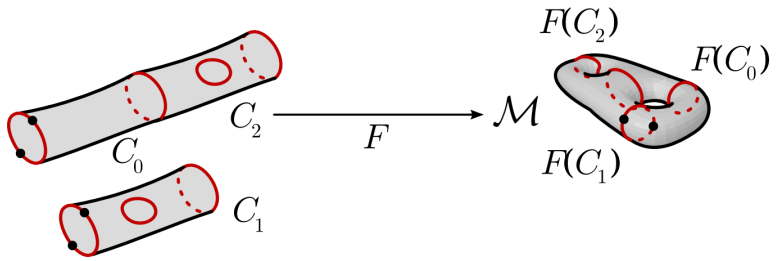


FIGURE 6.4 – Un M-pavage $(C, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$ (figure 6.3) par un complexe m-cellulaire ayant trois 2-m-cellules

Comme pour le cas des M-tuiles, un M-pavage est totalement décrit par le complexe m-cellulaire $\mathcal{C}$ et les images de chaque m-cellule de $\mathcal{C}$ dans $\mathcal{M}$.

Si les 2-m-cellules de $\mathcal{C}$ sont des n-polygones, le M-pavage est un pavage de $\mathcal{M}$ au sens classique du terme. La figure 6.4 montre un M-pavage.

6.3.1.4 Pourquoi un tel cadre ?

Le contexte de travail des M-tuiles et des M-pavages sera utilisé dans la section 6.3.2 pour produire une tuile unique sur un 2-mesh sans bords, et en section 6.4 pour produire des quadrangulations.

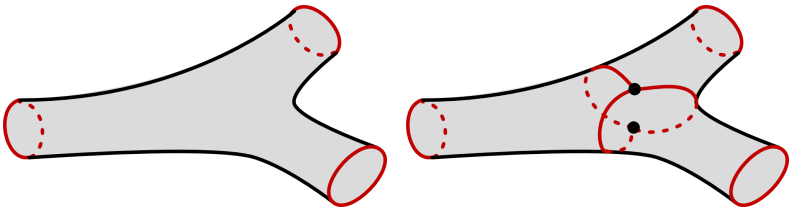
L'avantage de cette notation est de pouvoir décrire la génération de la M-tuile, à chaque étape de chaque algorithme, et d'inclure toutes les informations nécessaires à la génération de tuiles homéomorphes à un disque.

On peut aussi, à l'inverse, manipuler des M-tuiles non triviales, et utiliser les informations incluses dans le plongement pour retrouver les informations topologiques du maillage initial, ou encore retrouver les informations de recollement sur la tuile afin de reconstruire le maillage.

Colin de Verdière *et al.* [ECDVL07] ont proposé récemment un algorithme de décomposition d'une surface en pantalons. Un pantalon est une surface de genre nul avec trois bords distincts (comme illustré sur la figure 6.5(a)). Les M-tuiles et le M-pavage permettent aussi de décrire cette décomposition, en mettant en avant la représentation finale en pantalons, mais aussi le plongement entre les 2-m-cellules de genre zéro à trois bords, et le 2-mesh initial.

1. Cette condition permet d'éviter les « jonctions en T ».

Remarquons qu'une telle décomposition peut faire l'objet d'une décomposition en cylindres. En effet, on peut encore décomposer un pantalon en trois cylindres comme sur la figure 6.5(b). Chaque cylindre a un trois 1-faces et deux 0-faces, répartis sur deux frontières.



(a) Un pantalon, surface de genre 0 avec trois bords, chacun décrit par une unique 1-face.

(b) M-pavage d'un pantalon par trois cylindres. Chaque cylindre a deux frontières, la première ayant une unique 1-face, et la seconde avec deux 1-faces et deux 0-faces.

FIGURE 6.5 – Un pantalon et le M-pavage correspondant avec trois cylindres.

Cet exemple montre la large flexibilité de cette approche, permettant de décrire des segmentations géométriques et topologiques variées, en gardant une trace des états intermédiaires de l'algorithme qui les a générés.

6.3.2 Pavage avec une unique tuile

Soit P un n-polygone, et $\mathcal{M}$ un 2-mesh. Paver $\mathcal{M}$ avec une unique tuile $T = (P, \mathcal{M}, F)$ n'est pas trivial si $\mathcal{M}$ n'est pas homéomorphe à un disque.

6.3.2.1 Découpage et pavage

Le cœur de l'algorithme de pavage, est le calcul d'un découpage de la surface. En effet, le lemme suivant nous donne l'équivalence :

Lemme 4 *Un pavage $T = (P, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$ est équivalent à un ensemble Γ de 1-faces du maillage, avec la propriété que le complémentaire $\overline{\Gamma}$ de Γ dans $\mathcal{M}$ est homéomorphe à un disque. Γ est appelé un découpage de $\mathcal{M}$.*

Preuve Soit $T = (P, \mathcal{M}, F)$ un pavage. L'image des arêtes de P donne un découpage.

Considérons désormais un découpage Γ de $\mathcal{M}$ avec $\overline{\Gamma}$ homéomorphe à un disque. On peut construire un pavage de $\mathcal{M}$ par une unique tuile $T = (P, \mathcal{M}, F)$ de la manière suivante :

– L'image du corps des 2-m-cellules de P est définie comme $\overline{\Gamma \cup B}$, où B est le bord du maillage.

– L'image des sommets V de P sur $\mathcal{M}$ sont définis par l'ensemble des sommets de $\Gamma \cup B$. En utilisant un prolongement par continuité de l'homéomorphisme de $\overline{\Gamma \cup B}$ vers le disque, l'image des sommets de $\Gamma \cup B$ sont sur le bord du disque. Ils définissent les sommets du polygone.
– Chaque côté de P est aussi défini par cet homéomorphisme.

Remarque Le polygone que l'on construit de cette manière a au moins autant de sommets que le découpage. Cependant, il est possible de baisser ce nombre en supprimant les sommets du bord de P qui ne prennent part qu'à deux arêtes de $\Gamma \cup B$ (Figure 6.6). Comme nous le verrons en section 6.3.2.5 cette reduction peut devenir importante.

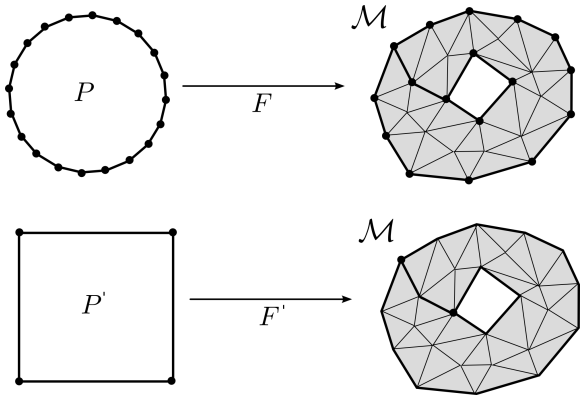


FIGURE 6.6 – Une M-tuile $(P, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$. P est le polygone construit lors de la preuve du lemme 4. La M-tuile $(P', \mathcal{M}, F')$ est obtenue en supprimant les sommets de P dont l'image prend part à exactement deux arêtes de $\Gamma \cup B$.

Par la suite, les 0-faces de $\mathcal{M}$ qui sont dans $F(V)$ seront appelées multipoints. Cette définition s'étend naturellement lorsque F est le plongement associé au M-pavage contenant plus d'une M-tuile , et avec des M-tuiles non polygonales.

6.3.2.2 Calculer un découpage avec une longueur minimale

Maintenant que nous pouvons obtenir une tuile à partir d'un découpage, nous allons décrire le processus topologique de découpage d'un maillage en un disque.

Ce type de découpage est un processus topologique dont le but est de modifier le genre et le nombre de bords de la surface jusqu'à l'obtention d'une surface homéomorphe à un disque. Nous avons déjà abordé l'algorithme section 6.2.

Définition

- Un 1-chemin sur un 2-mesh $\mathcal{M}$ est une liste de n arêtes orientées $s_0, ..., s_{n-1}$ telle que pour tout i entre 0 et $n-2$, le second sommet de l'arête s_i est le premier sommet de l'arête s_{i+1}.
- Un 1-cycle sur un 2-mesh $\mathcal{M}$ est un 1-chemin $s_0, ..., s_{n-1}$ tel que le premier sommet de s_0 est le second de s_{n-1}.
- Un 1-chemin de $\mathcal{M}$ est dit valide sur la M-tuile $(C, \mathcal{M}, F)$ si c'est l'image d'un chemin $p : [0, 1] \to C$ by F. [1]
- Un 1-chemin valide de $(C, \mathcal{M}, F)$ entre deux bords b_1 et b_2 est un 1-chemin de $\mathcal{M}$ image d'un chemin $p : [0, 1] \to C$ par F, avec $p(0) \in b_1$ et $p(1) \in b_2$.
- Un 1-lacet trivial de $(C, \mathcal{M}, F)$ est un 1-cycle de $\mathcal{M}$ image d'un cycle par F et homotope à une boucle réduite à un point.
- Un 1-lacet non-séparant de $(C, \mathcal{M}, F)$ est un 1-cycle sur $\mathcal{M}$ image d'une boucle par F qui ne coupe pas C en deux composantes connexes.

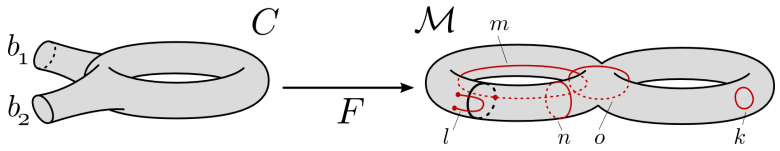


FIGURE 6.7 – Une M-tuile $(C, \mathcal{M}, F)$ d'un 2-mesh $\mathcal{M}$. C est une 2-m-cellule avec deux 1-m-cellules b_1 et b_2. k est un lacet trivial séparant, l est un 1-chemin non valide, m est un 1-chemin valide entre les bords b_1 etb_2, n est un 1-lacet valide non trivial séparant, et o un 1-lacet non séparant.

A l'aide de ces définitions représentées figure 6.7, la méthode de découpage global est décrite dans l'algorithme 2, et les différentes étapes décrites dans les sections suivantes.

Données : un 2-mesh $\mathcal{M}$ non homéomorphe à une sphère
$T = (C, \mathcal{M}, F)$ est une M-tuile initialisée par $\mathcal{M}$;
tant que *le genre de M n'est pas nul* **faire**
  Trouver c le plus court 1-lacet valide non séparant ;
  Couper T selon c ;
tant que *le nombre de bords de* $\mathcal{M}$ *est supérieur à 1* **faire**
  Trouver c le plus court chemin de T entre deux bords différents;
  Couper T selon c ;

Algorithme 2 : Algorithme global de découpage

La finalité de ce découpage est de produire une tuile unique $T = (P, \mathcal{M}, F)$.

1. Cette définition permet d'interdire au chemin de traverser les bords de C sur le 2-mesh

6.3.2.3 Trouver un 1-lacet non séparant

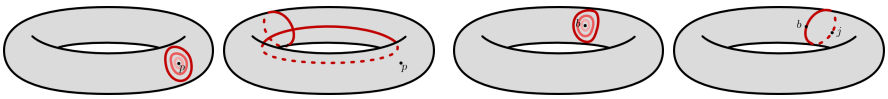
Soit $T = (C, \mathcal{M}, F)$ une M-tuile, et une distance d sur les arêtes de $\mathcal{M}$. Le 1-lacet non séparant le plus court se trouve à l'aide d'une liste de points base, points de départ potentiels au 1-lacet.

Cette liste est construite à l'aide d'une propagation sur la M-tuile. On choisit un point p à l'intérieur de C. Les triangles du 2-mesh sont ajoutés un par un lors de la propagation, en préservant la connexité, et en faisant en sorte que la surface recouverte soit homéomorphe à un disque [Het03]. À la fin de ce processus de propagation, il résulte des arêtes frontières sur le maillage. Certaines arêtes n'étant pas les frontières initiales de la M-tuile. On peut facilement calculer les points sur le maillage qui composent ce bord. Ce sont ces points qui forment les points base. En effet, tout 1-lacet non trivial passera par l'un de ces points. (Voir figure 6.8(a))

Pour chacun des points base b, on refait le même processus de propagation, mais en tenant compte de la 2-m-cellule que l'on a déjà construite. Si l'on a atteint le bord d'une 2-m-cellule, on ne continue pas la propagation. Si au sein d'une 2-m-cellule la propagation se rejoint en un point j, alors on stoppe le processus. Les deux chemins allant de b à j forment le 1-lacet non séparant. Ce processus est illustré figure 6.8(b).

Le 1-lacet non séparant non trivial le plus court est choisi en prenant le plus court parmi les 1-lacets calculés à partir des points base.

Si aucun lacet non séparant n'est trouvé à partir du premier point base, alors il n'y en a pas, la M-tuile est de genre 0.



(a) Propagation à partir du point p, calcul des points base.

(b) À partir d'un point base, on propage à nouveau et on stoppe dès que le front de propagation ne se rejoint pas en une frontière déjà connue.

FIGURE 6.8 – Les deux étapes de calcul d'un 1-lacet non trivial non séparant.

6.3.2.4 Découper une M-tuile

L'étape suivante dans l'algorithme 2 après la recherche d'un 1-lacet non trivial non séparant est le découpage de la M-tuile selon ce lacet. Selon la définition d'un 1-chemin valide c de $(C, \mathcal{M}, F)$, il existe un chemin $p : [0, 1] \to C$ tel que $F(p) = c$.

Découper une M-tuile selon ce chemin consiste à définir quelles seront les nouvelles faces de cette M-tuile. Le corps de la nouvelle 2-m-cellule est celui de l'originale privé de $p([0, 1])$. Les sommets de cette M-tuile sont ceux de la précédente plus ceux qui sont à la fois dans $p([0, 1])$ et dans le bord.

Les sommets sont aussi dupliqués s'ils font partie de $p([0,1])$. Naturellement, les bords de la 2-m-cellule sont définis par les bords précédents plus l'arête dupliquée qui aura pour image $p([0,1])$ dans $\mathcal{M}$. (Figure 6.9).

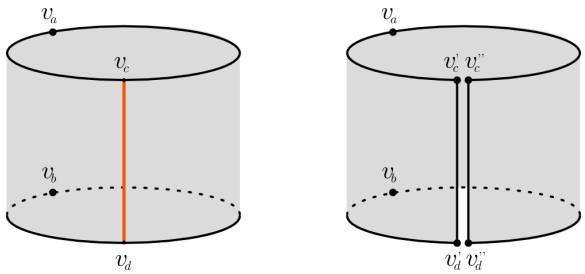


FIGURE 6.9 – Découpage de la M-tuile selon le chemin rouge. À gauche : la M-tuile (en gris) avec quatre arêtes, quatre sommets (v_a, v_b, v_c et v_d). À droite : le résultat du découpage de la M-tuile avec les sommets v_a, v_b, et les sommets dupliqués (v'_c, v''_c) et (v'_d, v''_d).

Ce découpage définit un mapping surjectif π de la nouvelle 2-m-cellule vers la 2-m-cellule initiale. De plus, on a $F \circ \pi = F_{new}$, avec F_{new} le plongement associé à la nouvelle M-tuile. La figure 6.10 montre le diagramme correspondant au déroulement de l'algorithme.

6.3.2.5 Contrôler le nombre d'arêtes de la M-tuile

Le nombre d'arêtes de la M-tuile est un moyen de caractériser la complexité du pavage. Cependant, l'algorithme 2 ne permet pas de contraindre le nombre d'arêtes de la M-tuile, il est uniquement basé sur la longueur des chemins, et nous devions développer un autre moyen de contrôler ce critère.

Le nombre d'arête du polygone final ne dépend pas de la topologie de la surface. La figure 6.11 montre la non unicité de ce nombre d'arêtes.

Si on minimise le nombres d'arêtes, on va pouvoir garantir que le pavage dépend uniquement de la topologie, la disposition de la surface n'ayant un impact que sur la localisation des plus courts lacets. De plus, le découpage en tuiles multiples, décrit en section 6.3.3, dépend de ce découpage, et le nombre de tuiles est directement relié au nombre d'arêtes de cette tuile unique. Enfin, si on ne prend pas en compte ce critère, on peut produire des arêtes de petite taille comme sur la figure 6.11 à droite, très contraignantes lors de la paramétrisation. Toutes ces raisons justifient de se pencher sur la question.

6.3.2.6 Nombre minimal d'arêtes

L'algorithme décrit dans le paragraphe 6.3.2 produit une tuile polygonale unique $T = (P, \mathcal{M}, F)$ à partir de n'importe quel 2-mesh (non homéomorphe

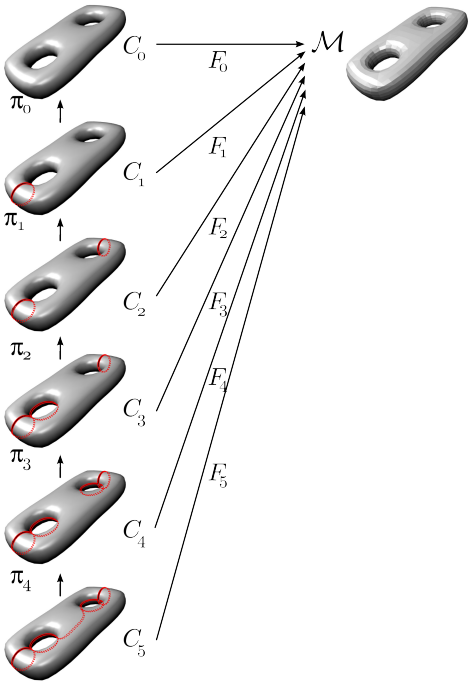


FIGURE 6.10 – Diagramme du procédé itératif. $(C_0, \mathcal{M}, F_0)$ est la M-tuile initiale. $(C_i, \mathcal{M}, F_i)$ est la M-tuile à la $i^{\text{ème}}$ itération de l'algorithme 2, et π_i le mapping surjectif du découpage de la M-tuile.

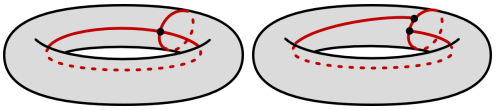
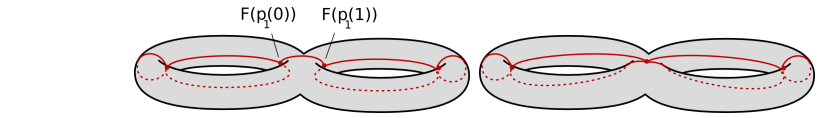


FIGURE 6.11 – Deux exemples de pavage du tore. Le polygone de la tuile de la figure de gauche a 4 côté, alors que celui de droite en a 6.

à une sphère), mais le nombre de 1-faces n'est pas contraint. En minimisant ce nombre, nous retrouvons le schéma polygonal canonique (voir section 6.2) de $\mathcal{M}$, et cela nous permet de caractériser la topologie de la surface [FK97].

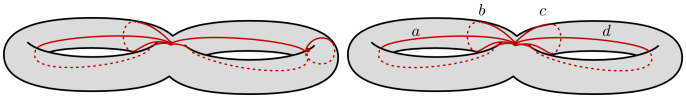
Formellement, le nombre minimal de 1-faces peut être calculé en réduisant itérativement le nombre de 1-faces. Une 1-face f_1, décrite par le chemin $p_1 : [0,1] \to P$, et où $F(p_1(0)) \neq F(p_1(1))$, est choisie. Selon le processus de découpage, il existe une unique 1-face f_2 avec un chemin $p_2 : [0,1] \to P$ ayant $F(p_1) = F(p_2)$. On peut supprimer f_1 et f_2 en regroupant $p_1(0)$ et $p_1(1)$, en supposant que les autres chemins de découpe qui ont $(F(p_1(0))$ ou $F(p_1(1))$ comme terminaison, sont déformés sur le 2-mesh afin de garder leur structure (Voir figure 6.12). Si la surface originale n'a pas de bord, on obtient ainsi une seule image sur le 2-mesh pour tous les 0-faces de la tuile.

On peut aussi construire un tel découpage, en changeant l'étape de sélec-



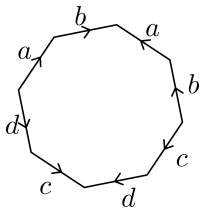
(a) Projection d'une tuile $(P, \mathcal{M}, F)$ sur $\mathcal{M}$.

(b) Projection après avoir fusionné $F(p_1(0))$ et $F(p_1(1))$.



(c) Projection après avoir fusionné deux autres sommets.

(d) Projection après avoir fusionné les derniers sommets. La tuile est désormais composée d'un sommet et quatre arêtes.



(e) Le schéma polygonal correspondant au maillage avec un nombre minimal d'arêtes.

FIGURE 6.12 – Étapes formelles de réduction du nombre d'arêtes d'une tuile.

.

tion du 1-lacet valide non séparant afin qu'elle n'utilise qu'un seul multipoint. L'algorithme 3 décrit ce procédé :

Données : Un 2-mesh $\mathcal{M}$, non homéomorphe à une sphère, sans bord.
$T = (P, \mathcal{M}, F)$ est une M-tuile initialisée à $\mathcal{M}$;
Trouver c le 1-lacet valide non séparant de T ;
Couper T selon c ;
Trouver c le 1-lacet valide non séparant de T entre les deux bords générés précédemment, avec $p(0) = p(1)$ (p est le chemin associé à c);
On pose $b = p(0)$;
Couper T selon c ;
tant que T *n'est pas homéomorphe à un disque* **faire**
    Trouver c le 1-lacet valide non séparant de T ayant $p(0) = p(1)$;
    Couper T selon c ;

Algorithme 3 : Découpage avec un seul multipoint.

Il y a un inconvénient majeur à cause de cet algorithme. Le nombre de 1-chemins partant de b dépasse le plus souvent l'arité du sommet (du point de vue du maillage). Il doit y avoir une modification locale du 2-mesh. Un procédé permettant d'éviter d'avoir des chemins tangents partant de b est présenté en section 6.4.

6.3.2.7 Méthode de découpage intermédiaire

Nous avons proposé un algorithme intermédiaire, (l'algorithme 4) afin de réduire mais pas minimiser le nombre de multipoints. C'est une version modifiée de l'algorithme original, qui minimise la création des multipoints lors du choix du 1-chemin entre deux bords.

Données : Un 2-mesh $\mathcal{M}$ non homéomorphe à une sphère.
$T = (P, \mathcal{M}, F)$ est une M-tuile initialisée à $\mathcal{M}$;
tant que *le genre de C n'est pas nul* **faire**
  Trouver c le plus court 1-lacet valide non séparant ;
  Couper T selon c ;
tant que *le nombre de bords de C est supérieur à 1* **faire**
  Trouver c le plus court chemin de T entre deux bords différents **en minimisant la création des multipoints**;
  Couper T selon c ;

Algorithme 4 : Méthode de découpage intermédiaire

La sélection du 1-chemin est définie par la procédure suivante. On crée un groupe composé de deux type d'éléments : un bord de C sans multipoints, ou bien un multipoint. On recherche le plus court chemin entre deux éléments de ce groupe :

- Si ce sont deux multipoints, la définition du 1-chemin est claire.
- Si l'un des éléments est un multipoint, il faut créer un multipoint sur le second.
- Si les deux éléments sont des bords, deux cas sont possibles :
  - Si les deux éléments ont la même image sur le maillage, alors on crée un seul multipoint.
  - Sinon, il faut créer un multipoint sur chacun d'eux.

6.3.3 Pavage avec des tuiles multiples

Paver avec une tuile unique permet d'atteindre beaucoup d'applications. Cependant, en considérant des applications comme la reconstruction de surfaces, ou la segmentation de maillage, une seule tuile ne suffit pas. La première application nécessite des formes de tuiles relativement proches d'un carré, la seconde nécessite par définition plusieurs régions. La forme et le nombre de côtés des tuiles sont des critères importants dans cette optique pour définir la qualité du pavage. Cette section décrit le pavage de surfaces basé sur la construction d'une unique tuile sur le maillage, vu dans la section précédente.

Soit un pavage $T = (C, \mathcal{M}, F)$, avec C un ensemble de polygones. Alors un pavage $T' = (C', \mathcal{M}, F')$ avec une seule tuile polygonale peut être facilement construit en fusionnant itérativement les tuiles de C, en faisant attention à préserver le genre à chaque étape. La construction inverse, c'est à dire paver avec des tuiles polygonales multiples à partir d'un tuile unique,

est l'objet de cette partie. C'est une approche naturelle et qui nous permet de réutiliser le procédé de découpage vu précédemment.

L'algorithme 5 propose ce pavage d'un 2-mesh $T = (P, \mathcal{M}, F)$, en utilisant aussi bien la tuile unique $T = (P, \mathcal{M}, F)$ obtenue par l'algorithme 1, 3 ou bien 4.

Définition Un 1-chemin découpant sur la tuile $T = (P, \mathcal{M}, F)$ de bord b est un 1-chemin sur $\mathcal{M}$ image d'un chemin $p : [0, 1] \rightarrow P$ par F, avec $p(0) \in b$, $p(1) \in b$ et tel qu'il existe un d dans $]0, 1[$ tel que $p(d) \notin b$.

Données : Un 2-mesh $\mathcal{M}$ sans bords, non homéomorphe à une sphère
$T = (\mathcal{C} = P, \mathcal{M}, F)$ est une tuile calculée avec la méthode de la section 6.3.2;
tant que *la M-tuile T doit être coupée* **faire**
  Sélectionner une 2-m-cellule C dans $\mathcal{C}$ qui doit être coupée ;
  Trouver un 1-chemin découpant c de C sur T ;
  Découper C selon c;
  Mettre à jour T;

Algorithme 5 : Méthode générique pour le pavage par découpe successive d'une tuile unique.

Le nombre de tuiles dépend uniquement du nombre de découpage effectués. De plus, le nombre de multipoints dépend à la fois du nombre de multipoints de la tuile originale, et du nombre de découpages. Chaque étape de l'algorithme 5 peut augmenter ou laisser tel quel le nombre de multipoints. Supposons que nous disposons d'un 1-chemin découpant. $p : [0, 1] \rightarrow P$ est le chemin associé sur le complexe cellulaire. La création d'un éventuel multipoint dépend de la localisation des points $p(0)$ et $p(1)$. Si $F(p(0))$ et $F(p(1))$ sont déjà des multipoints, il n'y a pas de création de multipoints. Sinon, il y a création de multipoints.

Les objectifs de découpe peuvent être différents selon le résultat souhaité. Si le nombre de multipoints doit rester constant, nous devons nous limiter à relier les multipoints sur le complexe m-cellulaire. C'est le cas de la partie 6.4.

6.4 Quadrangulation

6.4.1 Définition et algorithme

Définition Un quadrangle est un sous ensemble connexe du maillage $\mathcal{M}$ homéomorphe à un disque, et tel que la frontière est formée de quatre points particuliers (appelés coins) et de quatre bords reliant ces coins.

Il résulte de l'algorithme défini à la section 6.3.2 une tuile ayant un nombre pair de 1-faces. Si on déplie la surface obtenue, on obtient la fi-

gure 6.13. Une solution possible est de faire en sorte de découper entre deux multipoints (en violet sur la figure), en les reliant sur le complexe cellulaire associé.

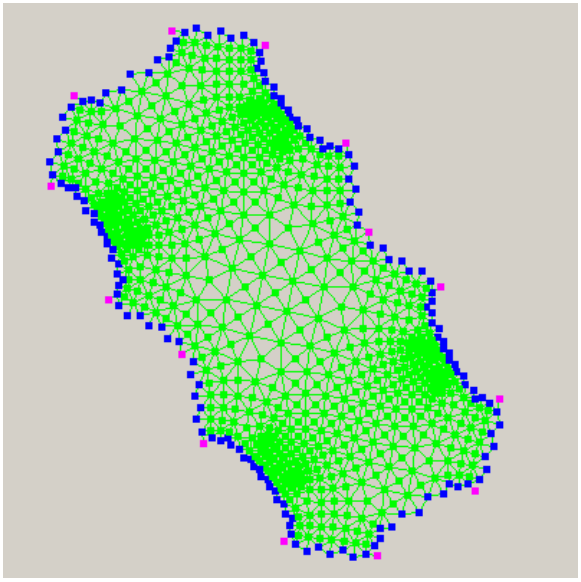


FIGURE 6.13 – Le double tore découpé, déplié avec la méthode 7.2.1. En bleu, les chemins de découpe. En violet, les multipoints.

L'algorithme 6 propose un découpage en quadrangles, sous la condition que chaque découpage respecte le critère $\mathcal{Q}$:

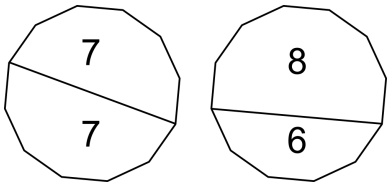
Définition Une M-tuile T vérifie le critère $\mathcal{Q}$ si la 2-m-cellule associée à T est un n-polygone au nombre de côtés pair, avec $n \geq 4$. Un pavage $\mathcal{T}$ satisfait le critère $\mathcal{Q}$ si et seulement si chacune de ses tuiles le satisfont.

Données : un 2-mesh $\mathcal{M}$ sans bords, non homéomorphe à une sphère
$T = (P, \mathcal{M}, F)$ est une tuile resultat de l'algorithme de la section 6.3.2;
tant que T *ne contient pas que des 4-polygones* **faire**
  Selectionner une 2-m-cellule C dans T qui ne soit pas un 4-polygone ;
  Chercher le 1-chemin découpant c le plus court sur T dans la tuile C, et qui préserve le critère $\mathcal{Q}$;
  Découper C selon c;

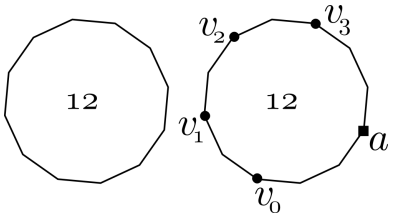
Algorithme 6 : Découpage en quadrangles par découpages successifs.

Le 1-chemin découpant le plus court est choisi parmi les 1-chemins construits à partir de toutes les paires de multipoints potentielles préservant le critère $\mathcal{Q}$

(comme illustré sur la figure 6.14(a)). Pour chaque 0-face a d'un n-polygone, on sélectionne des 0-faces valides v_i^a qui préservent le critère $\mathcal{Q}$. Ensuite, on calcule le 1-chemin le plus court entre a et chacun des v_i^a (Figure 6.14(b)).



(a) À gauche : un mauvais découpage qui ne préserve pas le critère $\mathcal{Q}$. À droite : un bon découpage.



(b) Exemple : un 12-polygon. a est la 0-face sélectionnée, et les v_i les extrémités potentielles d'un 1-chemin découpant preservant le critère $\mathcal{Q}$.

FIGURE 6.14 – Découpage d'un polygone pair en deux. Le nombre de côtés est indiqué dans le polygone.

Cet algorithme produit des tuiles qui ont la particularité de ne pas se « joindre en T », ce type de jonctions n'étant pas souhaitable lorsqu'on veut faire de la reconstruction de surfaces.

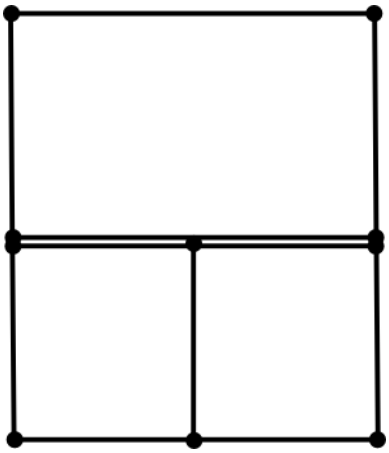
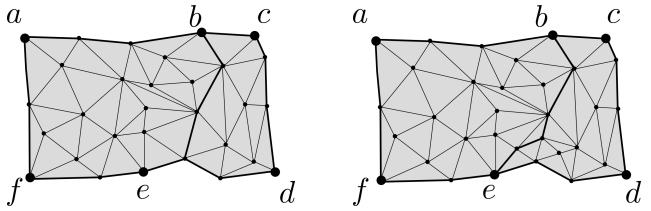


FIGURE 6.15 – Exemple de jonction en T.

6.4.2 Décollage de bords

Le découpage tel qu'il est construit présente quelques problèmes. Dans certains cas, les quadrangles obtenus ne sont pas satisfaisants, comme illustré sur la figure 6.16(a). Ces quadrangles peuvent présenter des pincements au niveau des sommets lors de la creation d'un chemin du multipoint e au

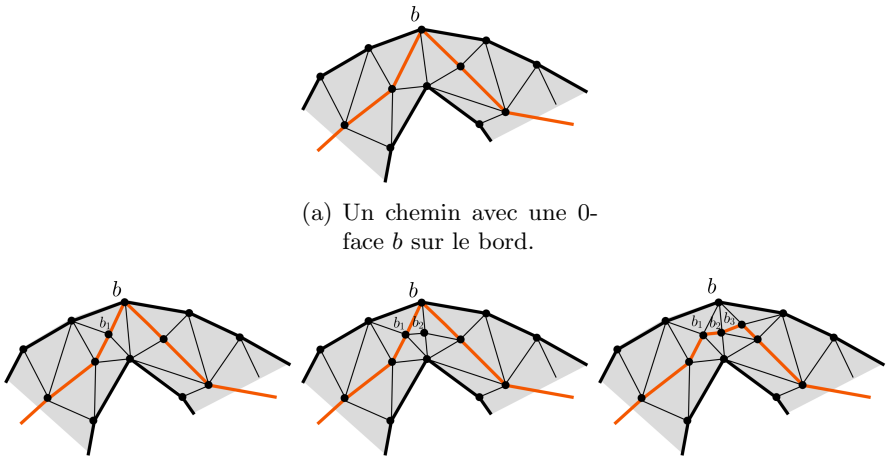
multipoint b. Un processus de décollage des bords doit alors être appliqué. Ce processus est décrit sur la figure 6.17(b), on raffine le maillage autour des bords non satisfaisants. Chaque 0-face qui n'est pas un multipoint voit les arêtes qui l'entourent sectionnée en deux. Le raffinement ajoute alors des sommets et des arêtes, mais sans changer la forme initiale.



(a) Une tuile hexagonale avec six sommets a, b, c, d, e, f et un chemin découpant entre b et e qui longe le bord .

(b) La même tuile après l'étape de décollage du bord entre b et e.

FIGURE 6.16 – Décollage de bords : le chemin b-e découpe la surface sans longer le bord.



(a) Un chemin avec une 0-face b sur le bord.

(b) Ètapes intermédiaires du décollage de bords autour de b. Toutes les arêtes contenant b, excepté les arêtes du bord, sont coupées en deux, et le maillage raffiné.

FIGURE 6.17 – Procédure de décollage du bord.

Erickson *et al.* [EHP04] ont montré que le découpage de l'algorithme 1 est une bonne approximation du découpage minimal. En utilisant la longueur minimale pour produire la quadrangulation à partir de la tuile unique, nous produisons une bonne approximation de la quadrangulation minimale.

Proposition 5 *Soit $\mathcal{M}$ un 2-mesh de genre g, sans bords, non homéomorphe à une sphère, et composé de n points. Une bonne approximation de la quadrangulation minimale s'obtient en un temps d'ordre $O(g^2 n \log n)$.*

Preuve Selon [EHP04], le découpage d'un maillage $\mathcal{M}$ de genre g avec n points se calcule en un temps de l'ordre de $O(g^2 n \log n)$. La tuile polygonale P obtenue a m 0-faces. La suite de l'algorithme de quadrangulation est composée du découpage du polygone prenant appui sur le maillage, et selon les plus courts chemins. L'algorithme de Dijkstra effectue cette opération en $O(n \log n)$, et ce pour chacune des 0-faces de P. Le découpage d'une tuile en deux tuiles plus petites se fait donc en $O(mn \log n)$.

Soit un polygone avec m sommets, m pair, nous allons prouver par récurrence que le nombre de découpage de la tuile en quadrangles est égal à $\frac{m}{2} - 2$, sans introduire d'autres multipoints. En effet, cette proposition est vraie pour $m = 4$. Supposons la propriété vraie jusqu'à $m - 2$. Pour $m > 4$, la construction, qui préserve le critère $\mathcal{Q}$, découpe le polygone en deux autres polygones plus petits de p et q sommets. D'après le critère $\mathcal{Q}$, p et q sont pairs aussi, supérieurs ou égaux à 4, et avec $p + q = m + 2$. La proposition implique que le nombre de découpages de ces deux tuiles est égal à $\frac{p}{2} - 2$ et $\frac{q}{2} - 2$. Le nombre de découpages est donc $\frac{p}{2} - 2 + \frac{q}{2} - 2 = \frac{m}{2} - 2$. La quadrangulation d'une tuile unique est donc de complexité $O(m^2 n \log n)$. De plus, nous avons remarqué que le nombre de sommets du polygone et le genre sont étroitement liés, la complexité globale de l'algorithme de quadrangulation est donc $O(g^2 n \log n)$.

6.4.3 Pavage avec un nombre minimal de quadrangles

L'algorithme qui produit un pavage en quadrangle $4g$, présenté dans [Het03], à partir d'un 2-mesh $\mathcal{M}$ de genre g . Cependant, il ne produit pas de pavage minimal au sens du nombre de quadrangles. Il est possible avec l'algorithme 3 de paver une surface avec un nombre minimal de quadrangles, et de relier le nombre de quadrangles au genre de la surface :

Proposition 6 *Soit $\mathcal{M}$ un 2-mesh compact, connexe, sans bords, de genre $g \geq 1$. $\mathcal{M}$ peut être pavé avec un nombre minimal de quadrangles égal à $2g - 1$.*

Preuve Soit un polygone à n sommets, avec n pair. Le nombre de découpages nécessaire pour produire une quadrangulation est $\frac{n}{2} - 2$, selon la proposition 5. Le nombre de quadrangles obtenus est donc $\frac{n}{2} - 1$. De plus, le nombre minimal de cotés du polygone est donné par le schéma polygonal. Il y a donc au minimum $4g$ sommets sur un 2-mesh de genre g [FK97]. Le nombre minimal de quadrangles est alors de $2g - 1$.

6.5 Autres techniques de quadrangulation

Il existe d'autres techniques de segmentation en quadrangle. Elles sont presque toutes issues de techniques de remaillage de triangulations en quadrangles.

Dans [All07, TACSD06, DBG+06], la quadrangulation est définie à l'aide des fonctions propre, issue de l'analyse spectrale de la surface. Soit une fonction f définie sur la surface. On définit le laplacien Δf de f par $\Delta f = -Lf$ où L est un opérateur linéaire :

$$L_{ij} = \begin{cases} \sum_k w_{ik} & \text{si } i = j \\ -w_{ij} & \text{si l'arête } (i,j) \text{ est dans le maillage} \\ 0 & \text{sinon} \end{cases} \tag{6.1}$$

Les poids w_{ij} peuvent être différents selon l'application initiale choisie. Dans le cas de graphes simples, il est usuel de prendre l'inverse de l'arité du sommet. Dans le cas des maillages 3D, il est plus intéressant de prendre les poids discrets harmoniques (équation (7.1) section 7.2.2). Les vecteurs propres de l'opérateur linéaire L forment ce qu'on appelle les fonctions propres du laplacien. Les extrema de ces fonctions propres nous aident à former le complexe de Morse-Smale [EHNP03], qui optimisé forme une quadrangulation sans jonction en T.

Dans [EH96], la méthode est différente, une famille de points est choisie sur le maillage afin d'en extraire le diagramme de voronoï associé (sur le maillage) et la triangulation de Delaunay. Le nombre de triangles construits est volontairement pair. Les triangles sont ainsi fusionnés deux à deux afin de produire la quadrangulation finale, à l'aide d'un algorithme d'optimisation sur le graphe obtenu.

Chapitre 7

Paramétrisation

La paramétrisation (également appelée mapping) a fait l'objet de beaucoup d'investigations, que ce soit depuis les premiers pas de la cartographie jusqu'aux domaines plus récents de l'application de texture et de reconstruction de surfaces.

Pour un résumé des techniques de paramétrisation, se référer à [SPR06, LM00, FH05].

7.1 Propriétés des techniques de paramétrisation

Généralement, la paramétrisation plonge un maillage surfacique défini dans $\mathbb{R}^3$, homéomorphe à un disque, dans un plan. On définit ainsi un plongement f, aussi appelé paramétrisation, qui transforme le maillage initial défini en 3D en un maillage final 2D. Il est souvent souhaitable que le maillage final respecte les mêmes caractéristiques que le maillage initial.

On peut considérer le processus de mapping comme une déformation : les triangles vont se tasser, s'allonger, se déformer, ou bien s'étirer, se rétrécir, etc... La ressemblance du maillage final au maillage initial permet de cataloguer les différentes classes de paramétrisation :

- Le mapping isométrique respecte les longueurs des arêtes.
- Le mapping conforme préserve les angles.
- Le mapping « equiareal » préserve les aires.
- Le mapping harmonique, est un mapping conforme vérifiant les propriétés de Cauchy-Riemann issues de la géométrie différentielle.

A noter que le mapping isométrique a la particularité d'être à la fois un mapping conforme et equiareal.

S'il est très difficile d'obtenir un mapping isométrique, c'est parfois parce que l'espace d'arrivée n'est pas toujours le même que l'espace de départ. En particulier, obtenir un mapping isométrique 2D d'une forme 3D complexe se révèle la plupart du temps impossible, car « l'aplatissement » déforme considérablement les arêtes, et les propriétés de longueur sont violées. De

plus, comme nous le verrons dans la partie suivante, la forme paramétrée qui doit être obtenue peut être contrainte sur les bords.

La plupart des algorithmes de paramétrisation sont construits de telle manière que la paramétrisation obtenue soit conforme (ou la plus conforme possible). C'est un compromis entre le mapping isométrique, qu'on pourrait qualifier de parfait, mais impossible à atteindre sauf cas dégénéré, et les applications comme le plaquage de textures qui sont guidées par la moindre déformation possible de la texture lorsqu'on la plaque sur le maillage.

7.2 Différentes catégories de paramétrisation

Le principe de calcul est de trouver le plongement d'un maillage sur un autre maillage planaire qui vérifie certaines des propriétés précédentes. On choisit alors la technique de paramétrisation qui convient le mieux à nos besoins.

Les techniques de paramétrisation se classent en deux catégories :

- La paramétrisation à bords libres
- La paramétrisation à bords fixes

7.2.1 Paramétrisation à bords libres

La paramétrisation à bords libres, que l'on appelle aussi dépliage de surfaces, existe dans de nombreuses formes. On pourra citer par exemple ABF [SdS01, SLMB05], le circle packing [RS87, CS03, Ste03] ou encore Least Squares Conformal Maps [LPRM02].

Ces trois méthodes produisent des paramétrisations conformes (quasi-conforme pour le circle packing). Elles sont particulièrement utilisées lors du dépliage de surfaces, ou encore de l'application de textures. Cependant, à propos de cette dernière, il est important de noter que si le maillage initial (homéomorphe à un disque) représente un objet, celui-ci présentera à coup sûr des jonctions de texture aisément distinguables. En effet, les bords libres du maillage final forment des frontières non régulières sur la texture, et laissent apparaitre sur l'objet, grâce à la texture, les lieux de découpe.

La figure 7.1 montre le double tore déplié avec le circle packing, déjà montré figure 6.13, ainsi qu'un plaquage de texture. Il est clair sur cette figure que le bord a, qui se recollera avec le bord a^{-1}, laissera apparaître des coutures sur l'objet.

De plus, de tels algorithmes doivent fournir un mapping bijectif, et aussi prendre en compte le non recouvrement. Il est possible qu'un dépliage ne produise pas un maillage au sens propre du terme, certaines parties pouvant se recouvrir. La condition de non recouvrement est une condition primordiale et fournit la grande difficulté de cette paramétrisation.

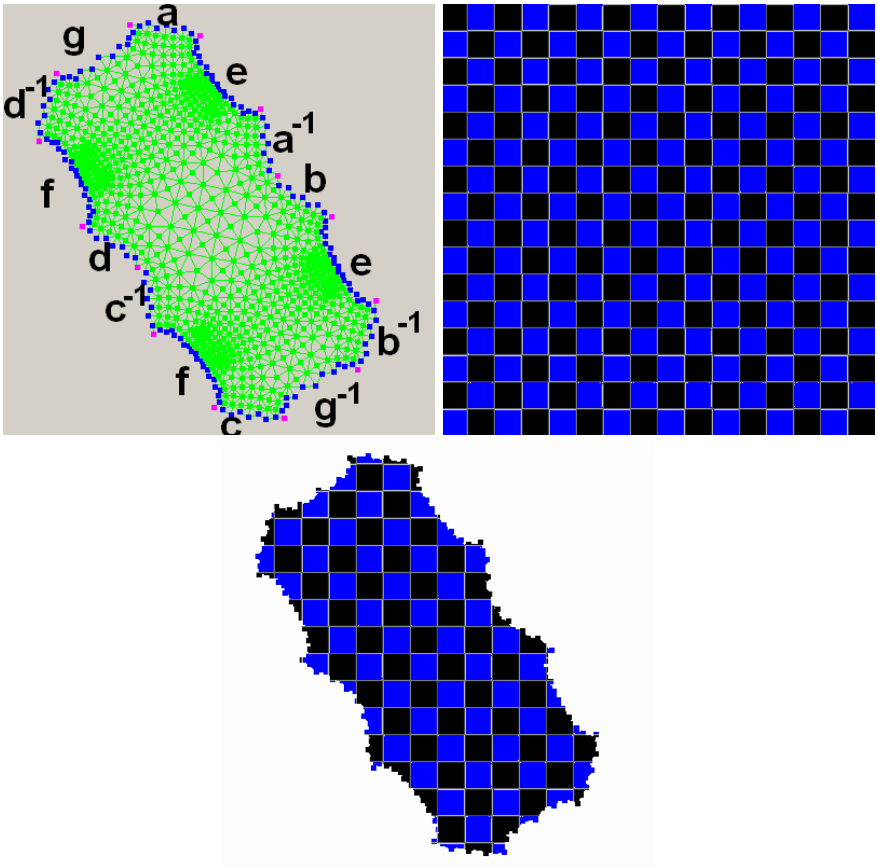


FIGURE 7.1 – Double tore déplié avec le circle packing. Schéma polygonal canonique associé. Coordonnées de texture.

7.2.2 Paramétrisation à bords fixes

Selon l'application choisie, il est parfois plus pertinent de fixer les bords du maillage final. Par exemple, dans le cas de la reconstruction de surfaces, un maillage homéomorphe à un disque peut être approché par une surface paramétrée. Cette surface paramétrée prend des valeurs de paramètres dans $[0,1]^2$. Fixer les bords du maillage final permet aussi un plus grand confort dans le raccordement de textures (pour faire disparaitre les coutures, et en imaginant que la texture est périodique), et d'éviter les problèmes de jonction soulevés à la section précédente.

Les premières paramétrisations à bords fixes, qui n'apparaissaient pas sous un tel nom, se trouvent dans les travaux de Tutte. En effet, on retrouve dans la théorie des graphes des problématiques similaires à la manipulation de maillages. Ce qu'on appelle aujourd'hui le Mapping barycentrique de Tutte [Tut60, Tut63] est devenu la base de toute paramétrisation à bords fixe. Conçu préalablement pour identifier une homotopie entre deux réseaux,

le mapping barycentrique a rapidement été adapté comme paramétrisation. Le principe est relativement simple. On identifie le maillage initial à un réseau de ressorts, on repère son bord. On fixe le bord du maillage final en donnant une image à chaque point du bord du maillage initial. Et on « lâche » le tout, les ressorts faisant le reste. L'état stable du système représente alors la solution.

C'est une vision très mécanique (et surtout empirique) du processus. Pour être un peu plus formel, Pinkall et Polthier [PP93] ont défini l'énergie que représente un tel système mécanique. La minimisation de cette énergie représente ce qu'on appelle une méthode barycentrique.

Si au départ Tutte avait choisi un poids uniforme sur chacune des arêtes de son graphe, les méthodes barycentriques actuelles de paramétrisation se distinguent en choisissant des poids différents, afin de respecter des propriétés de maillage différentes. Parmi les plus connues et les plus utilisées, on trouve :

- Discrete Authalic parameterization, la paramétrisation introduite par Desbrun [DMA02].
- Les cartes harmoniques discrètes (Discrete Harmonic Maps) introduites par [EDD+95], qui minimisent l'énergie introduite par Pinkall et Polthier [PP93].
- Les Mean value coordinates, introduites par Floater [Flo03].

La bibliothèque de calcul CGAL [cga08] permet de résoudre une bonne partie des méthodes précitées, et a été utilisée lors des résultats de la partie 9.

La figure 7.2 montre la configuration d'une arête à l'intérieur du maillage initial. Tous les poids seront affectés en fonction de la configuration locale sur le maillage initial. Seules les arêtes à l'intérieur du maillage ont besoin d'avoir des poids associés, les autres étant déjà fixées.

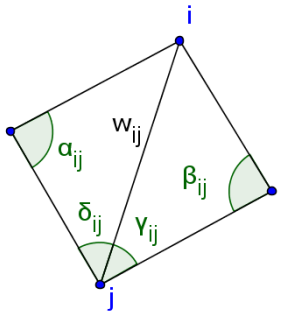


FIGURE 7.2 – Configuration d'une arête à l'intérieur du maillage initial

Les Discrete Harmonic Maps permettent de définir des poids au système mécanique qui minimisent l'énergie de Pinkall et Polthier. Ces poids prennent la valeur indiquée équation (7.1). On doit cependant ajouter que cette paramétrisation ne produit pas toujours un maillage. En effet, les poids

peuvent parfois être négatifs, ce qui peut donner lieu à des recouvrements, même si cela n'arrive que rarement.

$$w_{ij}^{DHM} = \cot \alpha_{ij} + \cot \beta_{ij} \tag{7.1}$$

Les poids associés à la méthode Mean Value Coordinates sont quant à eux indiqués dans équation (7.2). Contrairement à la paramétrisation précédente, Mean Value Coordinates assure une bijection entre le maillage initial et le maillage final à la seule condition que le bord soit convexe.

$$w_{ij}^{MVC} = \frac{\tan \delta_{ij}/2 + \tan \gamma_{ij}/2}{||i - j||_2} \tag{7.2}$$

7.3 Paramétrisation des quadrangles segmentés

Dans la section 6.4, nous avons vu comment produire une quadrangulation à l'aide du cadre de travail des M-pavages. La paramétrisation se fait par quadrangle et indépendamment des autres quadrangles. On extrait du M-pavage les informations d'un quadrangle :

- les points du maillage
- les facettes du maillage
- les quatre coins
- les points fictifs (les points qui ont été ajoutés par l'étape de décollage des bords 6.4.2)

Nous avons paramétré les quadrangles à l'aide de la librairie de calcul CGAL, en utilisant les Discrete Harmonic Maps 7.2.2. Cependant, il faut indiquer à CGAL comment on doit paramétrer le bord, puisque nous avons nos propres informations sur les coins. Nous verrons par la suite qu'il n'est pas nécessaire d'orienter les quadrangles, ceux-ci pouvant être libres. Par contre, l'étape de fixation des bords dans CGAL doit parcourir le bord afin de fixer leurs coordonnées dans le maillage final. Il faut pour cela choisir un des quatre coins pour qu'il soit le point $\{0, 0\}$. Les autres coins seront les points $\{1, 0\}$, $\{1, 1\}$ et $\{0, 1\}$. L'ordre est primordial, le sens de parcours l'est moins (puisqu'il fixe l'orientation du quadrangle). Les points du bord situés entre deux coins sont répartis sur l'arête du carré correspondante, selon les distances relatives (figure 7.3).

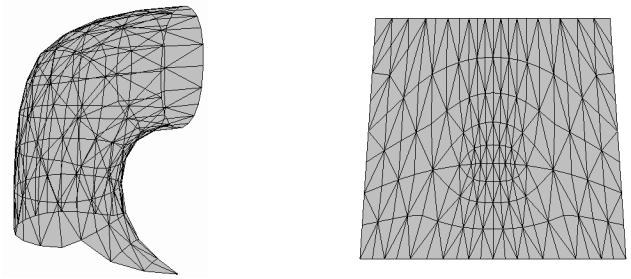


FIGURE 7.3 – Paramétrisation d'un tube

Chapitre 8

Fitting et Reconstruction

Nous avons décrit dans les sections précédentes les étapes nécessaires avant le fitting telles qu'elles sont décrites en section 5.2. La triangulation éventuelle, suivie du partitionnement en quadrangles (quadrangulation), et de la paramétrisation. À ce stade, nous disposons donc de multiples ensembles de points M_i^k paramétrés et des maillages en forme de quadrangles $\mathcal{Q}_k$ associés, avec des points fictifs (résultant de l'étape de décollage de bords 6.4.2).

Le but de cette section est de trouver un ensemble de surfaces paramétrées qui approxime au mieux le nuage de point initial, en tenant compte des contraintes de recollement que l'on pourrait choisir entre les quadrangles.

8.1 Reconstruction d'une surface paramétrée

Nous allons tout d'abord commencer par étudier la reconstruction d'un seul de ces nuages de points par une surface paramétrée.

8.1.1 Position du problème

Soit $(M_i)_{1\leq i\leq p}$, un ensemble de points paramétrés que l'on souhaite reconstruire. Rappelons la forme des surfaces paramétrées que nous prenons :

$$\begin{array}{rccc} \mathtt{S}: & X \subset \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & \displaystyle\sum_{i=1}^{m} P_i * f_i(s,t) \end{array}$$

Le problème de la reconstruction de surfaces paramétrées est donc le suivant :

Soit $(f_i)_{1\leq i\leq m}$ une base de fonctions donnée. Nous recherchons les points de contrôle de la surface paramétrée `S` passant au plus près du nuage de points $(M_i)_{1\leq i\leq p}$ paramétré par $(s_i,t_i)_{1\leq i\leq p}$. Dans le meilleur des cas, nous avons donc :

$$\mathtt{S}(s_i, t_i) = M_i \quad \forall i \in [\![1..p]\!] \tag{8.1}$$

Le problème de reconstruction peut être réduit en l'écrivant d'une autre manière :

$$\begin{aligned}
&\mathtt{S}(s_i, t_i) = M_i \quad \forall i \in [\![1..p]\!] \\
\Longleftrightarrow &\sum_{i=1}^{m} P_i * f_i(s_j, t_j) = M_j \quad \forall j \in [\![1..p]\!] \\
\Longleftrightarrow &\begin{cases}
\sum_{i=1}^{m} P_i^x f_i(s_j, t_j) = x_j \quad \forall j \in [\![1..p]\!] \\
\sum_{i=1}^{m} P_i^y f_i(s_j, t_j) = y_j \quad \forall j \in [\![1..p]\!] \\
\sum_{i=1}^{m} P_i^z f_i(s_j, t_j) = z_j \quad \forall j \in [\![1..p]\!]
\end{cases} \\
\Longleftrightarrow &\begin{cases}
A \times P^x = M^x \\
A \times P^y = M^y \\
A \times P^z = M^z
\end{cases}
\end{aligned}$$

En séparant coordonnées par coordonnées, il y a donc trois systèmes à résoudre. Chaque système détermine les coordonnées en x, y et z des points de contrôle P_i. Il existe plusieurs façons d'effectuer la reconstruction, selon la méthode de résolution du système (LP-Fitting ou Least Square Fitting). Comme les trois systèmes sont identiques, nous prendrons par la suite le premier système.

Avec cette méthode, on peut donc reconstruire des surfaces de Bézier (2.5), des surfaces B-Splines (2.8), ou d'autres surfaces paramétrées en définissant une base de fonctions $(f_i)_{1 \leq i \leq m}$.

Il faut bien garder à l'esprit que l'on résout trois systèmes, chacun représentant une coordonnée de l'espace. Le résultat final de reconstruction donne les points de contrôle de la surface cherchée. Il s'agit maintenant de quantifier et interpréter les résultats que donnent les différentes reconstructions possibles.

8.1.2 Conclusions

Avant d'aborder les aspects expérimentaux, nous pouvons d'ores et déjà donner quelques conclusions. L'intérêt théorique du LPF réside dans l'approximation au plus près des données, tandis que le LSQF fait une approximation en moyenne. Cela signifie que le comportement de la seconde reconstruction peut mettre volontairement à l'écart un point, et ce pour pouvoir être plus près des autres données. Au contraire, le LPF considère que toutes

les données ont la même importance, même si l'une d'entre elles semble s'éloigner du lot. C'est à la fois la force et la faiblesse de cette technique, permettant de faire apparaître une excroissance dans une reconstruction d'os là où le LSQF la réduirait, mais en incorporant aussi dans la reconstruction finale les bruits dûs à une éventuelle erreur de mesure.

C'est pourquoi, lorsque les données sont issues de mesures, il peut être déconseillé de prendre le LPF, tandis que si les données sont exactes, il est préférable de choisir cette méthode. En règle générale, et ce malgré les bons résultats en données bruitées (voir section 9.2.3), le LPF sera utilisé avec des données non bruitées, par exemple en CAO lorsque les concepteurs cherchent à configurer une surface par là où elle doit passer.

8.2 Reconstruction globale

Nous disposons de multiples ensembles de points M_i^k paramétrés, avec $1 \leq k \leq N$, et des maillages en forme de quadrangles $\mathcal{Q}_k$ associés, avec des points fictifs (résultant de l'étape de décollage de bords 6.4.2).

On appellera par la suite patch la structure qui représentera le quadrangle. Chaque patch se voit attribuer une surface paramétrée qui le représentera. Pour chacun des patches, on lui assigne les points 3D initiaux, en prenant soin d'enlever les points fictifs introduits au cours du processus de segmentation. Tous les points ont leurs paramètres issus de l'étape de paramétrisation précédente. Vu depuis l'espace des paramètres, le patch a l'aspect de la figure 8.1, les quatre coins sont nommés 0, 1, 2 et 3. Le maillage ne nous est plus utile à partir de cet instant.

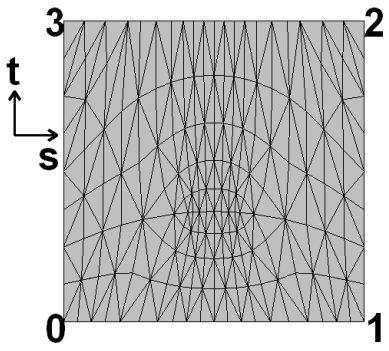


FIGURE 8.1 – Les quatres coins du patch. Les coordonnées paramétriques des points à reconstruire.

Nous devons aussi extraire les données de recollement issues de la M-tuile.

8.2.1 Déduction des contraintes de recollement

8.2.1.1 Graphe de recollement

Un graphe de recollement est un graphe permettant de donner les informations nécessaires au recollement. La figure 8.2 montre un graphe de recollement possible.

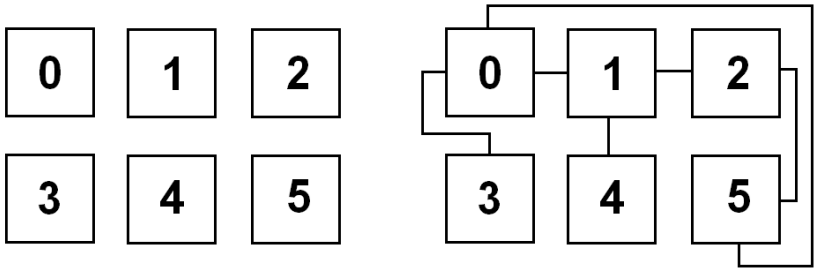


FIGURE 8.2 – Six patches et un graphe de recollement correspondant.

Ce type de graphe ne suffit pas. Nous devons prendre en compte des surfaces de topologie différente, pour cela, il faut distinguer les deux exemples suivants : le cylindre et le ruban de Möbius. L'orientation des arêtes qui doivent être recollées est très importante. Ainsi, la figure 8.3 montre une telle distinction.

FIGURE 8.3 – Graphe de recollement d'un cylindre (à gauche) et d'un ruban de Möbius (à droite).

En terme de codage, ces recollements peuvent s'écrire ainsi :

0 01 1 03
0 32 1 12 Pour le cylindre
0 01 1 03
0 32 1 21 Pour le ruban de Möbius

Pour une ligne comme « 0 01 1 03 », il faut comprendre : l'arête 01 du patch 0 se recolle avec l'arête 03 du patch 1, avec le point 0 du patch 0 qui coïncide avec le point 0 du patch 1, ainsi que le point 1 du patch 0 qui coïncide avec le point 3 du patch 1.

8.2.1.2 Choix de la qualité des surfaces paramétrées

Chaque patch sera représenté par une surface paramétrée. Pour les surfaces de Bézier, il y a deux paramètres à choisir : n et m. Le nombre de points de contrôle sera donc mn. Nous avons évoqué à la section 4 la différence entre approximation et interpolation, en précisant que le nombre de données expliquant les données initiales, doivent être en nombre inférieur. En d'autres termes, il doit y avoir moins de points de contrôle pour définir la surface qu'il n'y a de points 3D dans le quadrangle.

Par défaut, nous avons choisi de fixer le nombre de points de contrôle de chaque surface à des valeurs carrées connues comme les surfaces bicubiques (c'est à dire $4 \times 4 = 16$), les surfaces 5×5, etc... Il est aussi possible de choisir une valeur proche du nombre de points initiaux. Si P est le nombre de points, alors on peut choisir n et m comme la partie entière de la racine carrée de P.

$$n = m = \left\lfloor \sqrt{P} \right\rfloor \tag{8.2}$$

Si la surface se révèle de dimension trop faible, c'est tout le processus de quadrangulation qu'il faut revoir, puisque nous n'avons aucun moyen de régler ce paramètre.

8.2.1.3 Type de recollement

En ce qui concerne les différents types de recollements, il est possible d'appliquer une régularité uniforme sur l'intégralité de l'objet, par exemple recollement C^1 sur toutes les arêtes. C'est ce qui est fait actuellement et présenté en section 9.3. Cependant, rien n'interdit d'ajouter des contraintes différentes selon l'endroit de la jonction. On imagine très bien ajouter au codage de contrainte une information supplémentaire portant sur le type de recollement. Par exemple, sur le ruban de Möbius, on pourrait avoir une continuité C^0 sur l'arête 0 01 1 03, et une continuité G^1 sur l'arête 0 32 1 21.

8.2.1.4 Extraction des contraintes de recollement d'un M-pavage

Les informations de recollement entre les M-tuiles sont portées par le plongement et par le complexe cellulaire associé. Il est toujours possible de récupérer les informations du graphe de recollement. On obtient ainsi la

carte globale de recollement, il ne reste plus qu'à déterminer les points de contrôle des surfaces paramétrées.

8.2.2 Reconstruction de plusieurs surfaces paramétrées par LP-Fitting

Comme indiqué à la section 9.2.4, nous ne nous intéresserons qu'à la reconstruction coordonnées par coordonnées. Aussi, il y a quand même plusieurs stratégies de reconstruction multiple par LP-Fitting. En effet, lorsqu'on qu'on reconstruit un unique patch par LP-Fitting, on assigne une distance maximale h. Ici, nous avons N patchs avec N distances maximales aux données. Nous devons déterminer une stratégie de résolution.

8.2.2.1 Reconstruction globale sans recollement

Lorsqu'aucune contrainte de recollement n'est prise en compte, la résolution se ramène à N systèmes indépendants qu'on peut résoudre en un seul programme linéaire. La fonction objectif sera alors la minimisation de la somme des h_i.

8.2.2.2 Contraintes de recollement

Les contraintes de recollement sont traitées une par une. Cependant, comme évoqué à la section 2.3, l'utilisation de contraintes G^1 nous oblige à fournir des coefficients α et β. Une étude du graphe de recollement est alors nécessaire.

Sur les N patches, il y a $4N$ sommets potentiels. Or, dès qu'une contrainte de recollement existe, elle est au moins C^0, et donc implique l'égalité des sommets deux à deux. A partir de ce constat et du graphe de recollement, on détermine tous les sommets, en gardant en mémoire quelles sont les arêtes qui contribuent au sommet. Nous avons donc toutes les informations nécessaires, sauf la carte de recollement.

Deux choix s'offrent à nous. La carte de recollement peut être déterminée par l'utilisateur, ou bien automatiquement. Là aussi, il est possible de choisir le type de carte : régulière ou choisie. La carte régulière, semble un bon compromis entre l'automatisme et l'intervention de l'utilisateur. Mais il est possible d'introduire une carte de recollement choisie de manière automatique, en se basant sur les travaux de Floater [Flo97] en utilisant la méthode Shape-preserving localement autour du sommet sur le maillage pour déterminer la carte.

8.2.2.3 Quelle fonction objectif pour le recollement ?

À partir du programme linéaire sans recollement de la section 8.2.2.1, nous devons ajouter des contraintes linéaires d'égalité entre les points de

contrôle. Dans la partie 2.3, nous avons déterminé les contraintes entre les points de contrôle pour effectuer les recollements. Comme les contraintes sont des contraintes 3D, que les coordonnées sont traitées séparément, on en déduit des égalités à ajouter dans le programme linéaire.

Les systèmes de reconstruction de patches ne sont plus indépendants puisque des contraintes linéaires lient les points de contrôle des différents patches. La résolutions des N programmes linéaires ne peut donc se faire indépendamment. Comme précédemment, deux fonctions objectifs sortent du lot la maximum des h_i ou la somme des h_i. Nous avons testé expérimentalement ces deux fonctions objectif (voir section 9.3).

Chapitre 9

Résultats expérimentaux

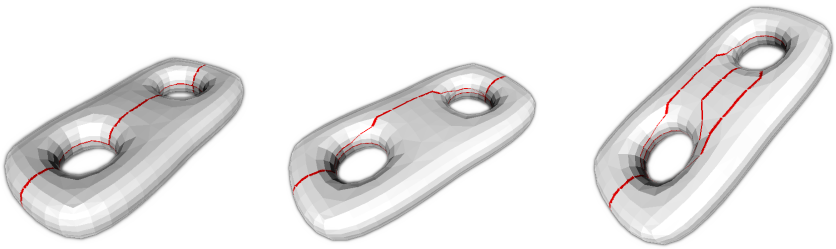
Ce chapitre regroupe les expérimentation que nous avons effectuées. Nous avons travaillé sur la plupart des échelons de la chaîne de reconstruction, depuis la segmentation jusqu'à la reconstruction.

9.1 Quadrangulation

Les résultats de quadrangulation qui sont présentés ici sont dûs à Jean-Marie Favreau, dans le cadre de notre collaboration sur le sujet des M-pavages et de la quadrangulation.

De nombreux maillages ont servi de base à notre expérimentation. Ceux-ci étaient de genre 1 à 4, et avaient entre 800 points à 50000 points.

Les figures 9.1 et 9.2 montrent les résultats obtenus.

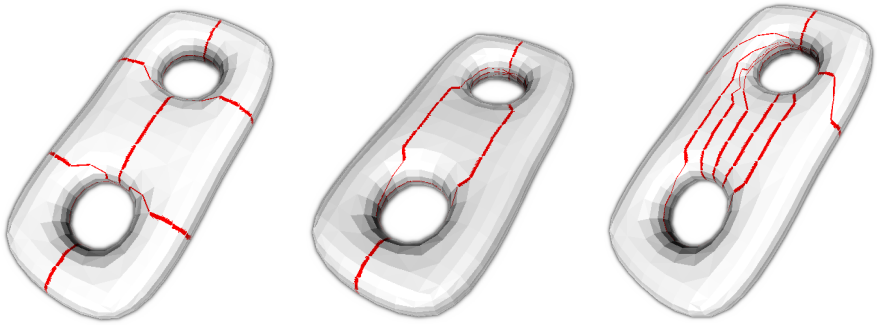


(a) Pavage en une unique tuile avec l'algorithme 1. Le résultat est un 14-polygone avec 4 multipoints.

(b) Pavage intermédiaire en une seule tuile, avec l'algorithme 4. Le résultat est un 10-polygone avec 2 multipoints.

(c) Pavage minimal en une unique tuile, avec l'algorithme 3. Le résultat est un 8-polygone avec 1 multipoint.

FIGURE 9.1 – Découpages à une seule tuile d'un 2-mesh de genre 2 sans bords. (830 points, 1664 triangles).



(a) Quadrangulation résultant de l'algorithme 6 avec comme base le résultat de l'algorithme 1 (Figure 9.1(a)). On obtient 6 quadrangles.

(b) Quadrangulation intermédiaire avec l'algorithme 6 avec comme base le résultat de l'algorithme 4 (Figure 9.1(b)). On obtient 4 quadrangles.

(c) Quadrangulation minimale de l'algorithme 6 avec comme base le résultat de l'algorithme 3 (Figure 9.1(c)). On obtient 3 quadrangles.

FIGURE 9.2 – Différentes quadrangulations d'un 2-mesh de genre 2 sans bords (830 points, 1664 triangles).

Selon l'algorithme initial de découpage en une tuile unique, le résultat est très différent.

La figure 9.2(a) montre un très bel exemple de quadrangulation facile à reconnaitre à l'œil. On distingue aisément les différentes composantes qui font le double tore.

On remarque aussi figure 9.2(c) que cette quadrangulation ne conviendra pas à la reconstruction de surfaces. En effet, celle-ci présente des grandes distorsions qui ne sont pas souhaitables dans de telles applications. Cela reste donc un résultat théorique, mais peu applicable en pratique.

Dans la figure 9.2(b) l'avancée annoncée par l'algorithme 4 n'a pas l'air d'être effective. Nous verrons dans la section 9.3 que c'est quand même une bonne quadrangulation. En effet, ce sont les deux figures 9.2(a) et 9.2(b) qui seront reprises pour faire les recollements multiples, et achever la chaîne de reconstruction.

Une autre figure (la figure 9.3) montre le gros désavantage de l'algorithme 1. Sur cette figure, un objet en forme de bretzel, on aperçoit les quatre coins d'un quadrangle, dont deux d'entre eux qui sont très rapprochés. La fusion des multipoints fait en sorte, dans la mesure du possible d'éviter une telle configuration.

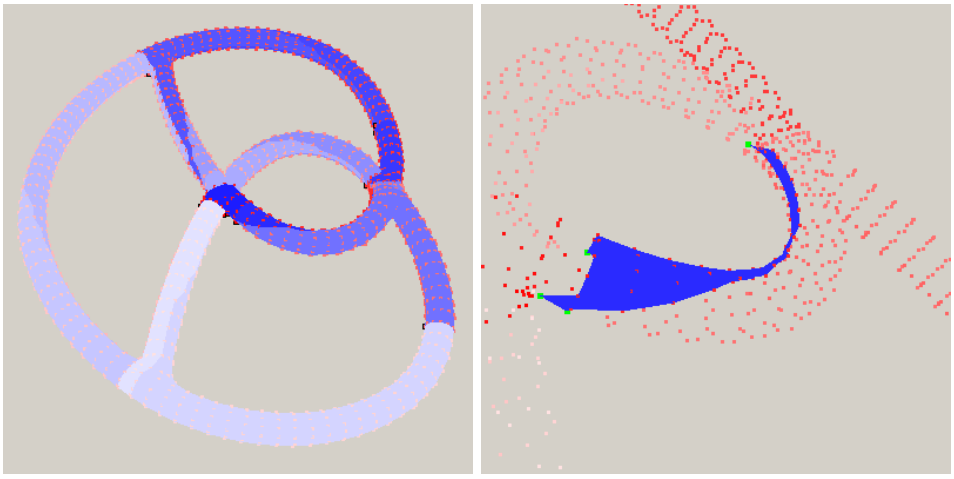


FIGURE 9.3 – Un quadrangle, issu d'un objet en forme de bretzel, résultant de l'algorithme 1. Ce quadrangle présente deux coins très proches. Cette configuration est non souhaitable lors d'une future paramétrisation ou reconstruction

9.2 Reconstruction d'une surface paramétrée

Nous avons évalué les différences entre le Least Square Fitting et le LP-Fitting, afin de confirmer les attentes théoriques que nous avons formulées en section 8.1.2.

Pour chercher la meilleure solution possible au problème de reconstruction (section 8.1.1), on peut alors choisir l'une ou l'autre des méthodes décrites à la partie 4.2.2 ou à la partie 4.2.3. Cela nous donne le Least Square Fitting (LSQF) pour la résolution avec les moindres carrés, ou le LP-Fitting [MGM06] (LPF) pour la résolution par programmation linéaire.

9.2.1 Protocole de test

Nous avons généré des points sur des surfaces paramétriques existantes : des surfaces de Bézier, des surfaces B-Splines, ou des sphères (paramétrées par leur angles sphériques). Pour plus de réalisme, nous avons perturbé aussi bien les coordonnées 3D des points, ainsi que leurs paramètres, selon un bruit gaussien, pour étudier le comportement de la reconstruction face au bruit.

Nous avons ensuite effectué la reconstruction selon les deux méthodes : Least Square Fitting, et LP-Fitting, et placé les résultats dans des histogrammes afin de comparer l'approche classique (LSQF) à notre nouvelle approche (LPF).

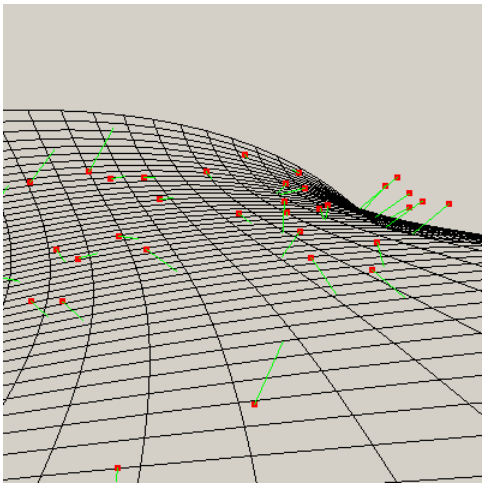


FIGURE 9.4 – Reconstruction du nuage de points rouge par une surface paramétrée. Vecteurs représentant l'erreur de reconstruction.

9.2.2 Interprétation des résultats

Pour chaque point M_i du nuage de point initial (en rouge sur la figure 9.4), il y a le point $\mathtt{S}(s_i, t_i)$ correspondant sur la surface paramétrée. On relève la distance euclidienne du point M_i au point $\mathtt{S}(s_i, t_i)$ (le vecteur $\overrightarrow{M_i\mathtt{S}(s_i, t_i)}$ est représenté en vert sur la figure 9.4), et on place le tout dans un histogramme.

Les figures 9.5 et 9.6 montrent la même sphère perturbée, reconstruite à l'aide d'une surface de Bézier et selon les deux méthodes. On remarque que la reconstruction par LSQF donne une boule presque parfaite, tandis que la reconstruction par LPF produit une sphère à l'aspect un peu bosselée. Cependant, c'est un effet attendu. En effet, nous avons introduit du bruit sur la sphère, et le bruit lors de l'utilisation de la norme $||.||_\infty$ a un impact plus prononcé. Cette différence entre les deux méthodes donne deux axes de reconstruction.

La méthode LSQF traitera beaucoup mieux les données bruitées avec le risque que certains détails importants, représentés par peu de points ou une densité de points plus faible, soient traités comme du bruit et disparaissent de la surface reconstruite. Il peut arriver que cette surface soit donc très loin d'un petit groupe de points, la fonction objectif quadratique préférant passer plus près d'un autre groupe de points plus dense.

La méthode LPF contient une contrainte uniforme. Celle-ci contraint la surface à être aussi près de tous les points, sans prendre en considération la caractéristique probabiliste de densité des points. La conséquence forte est qu'un jeu de données bruitées (ayant éventuellement des données fausses) donnera certainement un moins bon résultat que LSQF, alors que si les

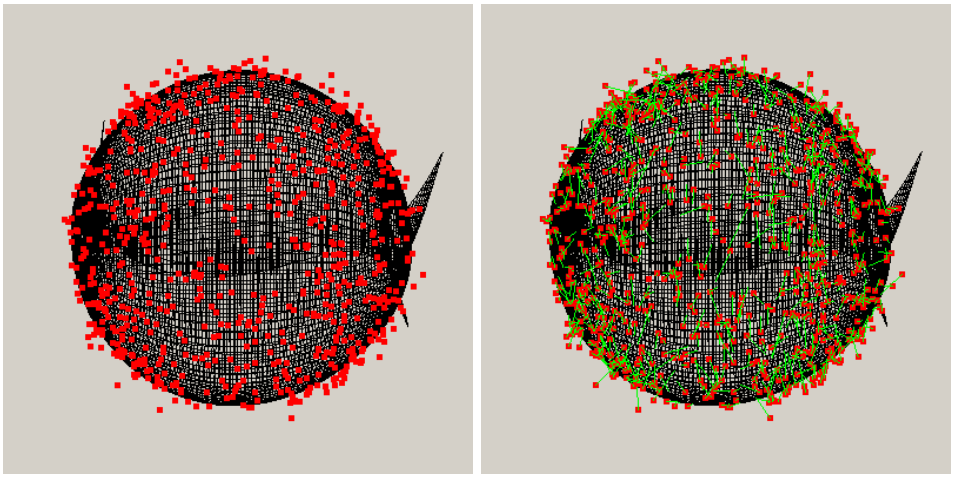


FIGURE 9.5 – Reconstruction d'une sphère avec Least Squares Fitting (LSQF)

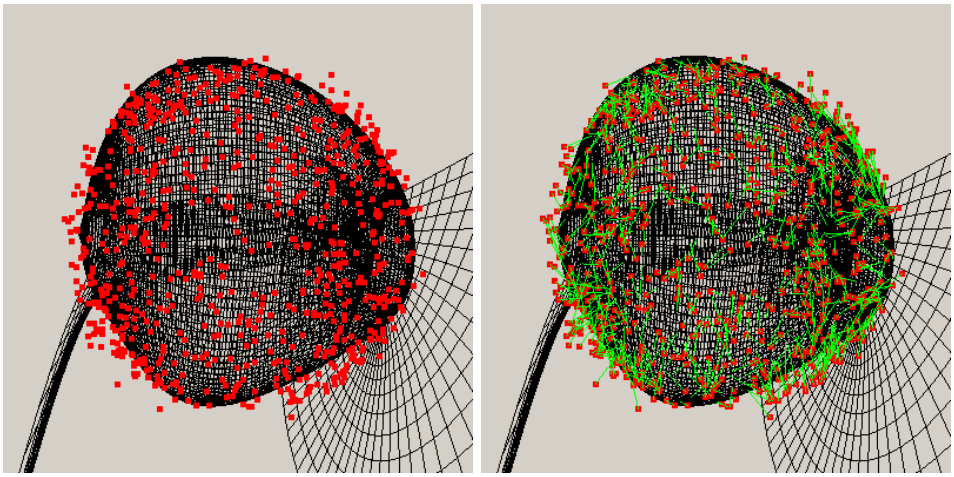


FIGURE 9.6 – Reconstruction d'une sphère avec LP Fitting (LPF)

données sont toutes aussi importantes les unes que les autres, cette méthode est certainement plus judicieuse.

9.2.3 Histogrammes

Pour chacune des surfaces reconstruites, nous avons testé les deux méthodes de reconstruction, puis relevé dans un histogramme, pour chacune la norme euclidienne de chaque vecteur d'erreur (en vert sur les figures).

Dans la plupart des cas, l'erreur maximale entre un point et son correspondant sur la surface est la plus basse avec le LP-Fitting (voir par exemple la figure 9.7). On remarque aussi sur cette figure qu'il y a une grand nombre

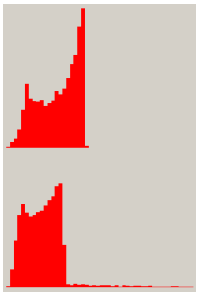


FIGURE 9.7 – Histogrames d'une sphère perturbée et reconstruite par une surface de Bézier 5*5 : LP-fitting (en haut) least squares fitting (en bas)

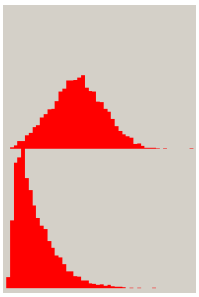


FIGURE 9.8 – Histogrames d'une autre sphère perturbée et reconstruite par une surface de Bézier 5*5 : LP-fitting (en haut) least squares fitting (en bas)

de points dont l'erreur est proche de l'erreur maximale.

Il y a cependant des cas où la tendance est inverse (comme sur la figure 9.8), c'est à dire la distance maximale est plus grande avec le LP-Fitting qu'avec le Least Square Fitting. Cela peut s'expliquer aisément : par construction de l'algorithme, l'erreur maximale coordonnées par coordonnées est toujours plus grande avec le LSQF qu'avec le LPF. Soient p_x, p_y et p_z les points où l'erreur par LQSF est maximale. Ces points peuvent être différents les uns des autres, et il peut arriver que l'erreur commise pour p_x en y et en z soit nulle. L'erreur maximale commise et relevée dans l'histogramme sera donc petite, malgré la grande erreur en x de p_x. Au contraire, les erreurs commises par LPF, même si elles sont plus petites, peuvent s'accumuler selon les trois coordonnées et former une erreur globale plus grande.

Comme les résolutions sont indépendantes, le LP-Fitting garantit une minimisation de l'erreur maximale coordonnées par coordonnées, mais ne garantit pas une minimisation de l'erreur euclidienne maximale. Ainsi, comme illustré figure 9.9, le LP-Fitting garantit uniquement que la surface passera dans une boite englobante autour de chacun des points. Malgré cela, dans la plupart des cas, l'erreur maximale du LP-Fitting reste aux alentours de 80% de l'erreur commise par Least Square Fitting, les figures 9.10 et 9.11

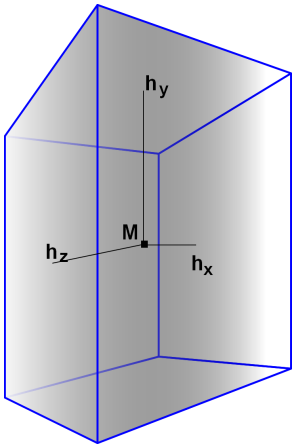


FIGURE 9.9 – Boite englobante autour du point M. La surface reconstruite par LP-Fitting passera au travers de la boite.

montrant des résultats plus standards.

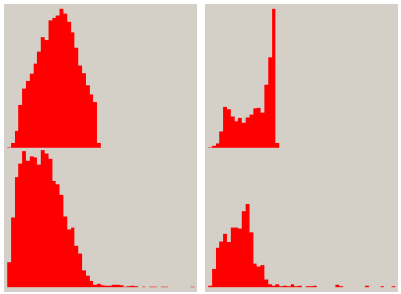


FIGURE 9.10 – Deux histogrammes de reconstruction : LP-fitting (en haut) least squares fitting (en bas)

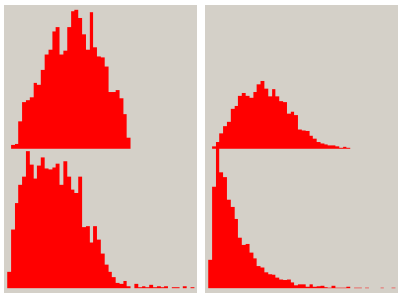


FIGURE 9.11 – Deux histogrammes de reconstruction de points très fortement perturbés : LP-fitting (en haut) least squares fitting (en bas)

Le résultat de cette expérimentation est que le LP-Fitting produit des

surfaces qui sont significativement plus proches du nuage de points initial en terme de distance euclidienne maximale. A noter également que la moyenne des erreurs maximales (qu'on devine sur les histogrammes) est certes plus grande avec LPF, mais pas excessive.

9.2.4 Discussion

Nous avons défini le fitting comme la minimisation indépendante des trois systèmes. Par LP-Fitting, il résulte trois programmes linéaires indépendants qui consistent à minimiser h_x, h_y et h_z. Il est possible de placer le tout dans un unique programme linéaire. Mais alors que choisir comme fonction objectif ? Le programme linéaire est le suivant :

$$\begin{cases} \underset{x \in \mathbb{R}^n}{Min}\ c^T x \\ Ax \leq b \end{cases} \tag{9.1}$$

On pourrait choisir de minimiser le maximum de h_x, h_y et h_z, afin de réduire la plus grande coordonnée de la boite englobante (figure 9.9). Pourtant, ce n'est pas une bonne idée. En effet, les applications comme la cartographie présentent des données 2.5D [WR89], et les coordonnées x et y sont souvent très facile à approximer. Si l'on ne contraint que le max de h_x, h_y et h_z, on ne contraint donc que h_z, laissant « libre » h_x et h_y entre 0 et h_z, alors qu'on pourrait les forcer à être proches de 0.

Comme les systèmes sont indépendants, la matrice A du programme linéaire est donc une matrice diagonale par bloc. Prendre comme fonction objectif la minimisation de la somme des $h_{\{x,y,z\}}$ est alors équivalent à la résolution de trois programmes linéaires.

Pour finir, un mot sur la difficulté de calcul. Mettre le tout dans un unique programme linéaire n'a pas que des avantages. En effet, cela influe beaucoup sur la taille mémoire lors de la résolution. La taille du programme linéaire peut elle aussi être très grande à l'origine si le nombre de points initiaux l'est. De même, en dimensions supérieures, c'est autant de difficulté supplémentaire qui s'ajoute. C'est pourquoi il est tout de même préférable de garder la résolution de trois programmes linéaires.

9.3 Reconstruction globale

La reconstruction de plusieurs surfaces paramétrées a été effectuée à l'aide des données précédemment étudiées.

9.3.1 Reconstruction d'une sphère

L'exemple qui illustre le mieux les recollements de patches sur une surface fermée est la sphère. La sphère a été partitionnée à l'aide de sa projection sur

le cube centré en son centre, et dont le graphe de recollement est représenté figure 9.12.

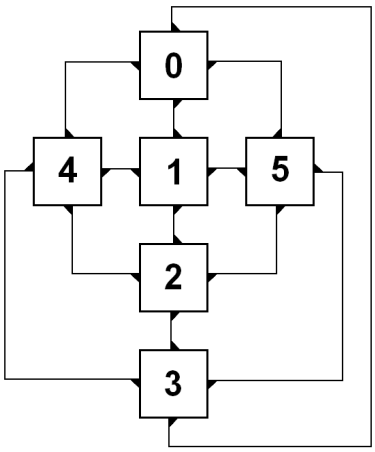


FIGURE 9.12 – Graphe de recollement de la sphère découpée en six morceaux

9.3.1.1 Surfaces bicubiques

Plusieurs reconstructions ont été effectuées, allant de la reconstruction sans recollement, avec recollement C^0, C^1 et G^1. De plus, les deux types de fonction objectif ont aussi été étudiées (voir figure 9.13). Les six surfaces de Bézier obtenues sont bicubiques (de degré 3×3 avec 16 points de contrôle).

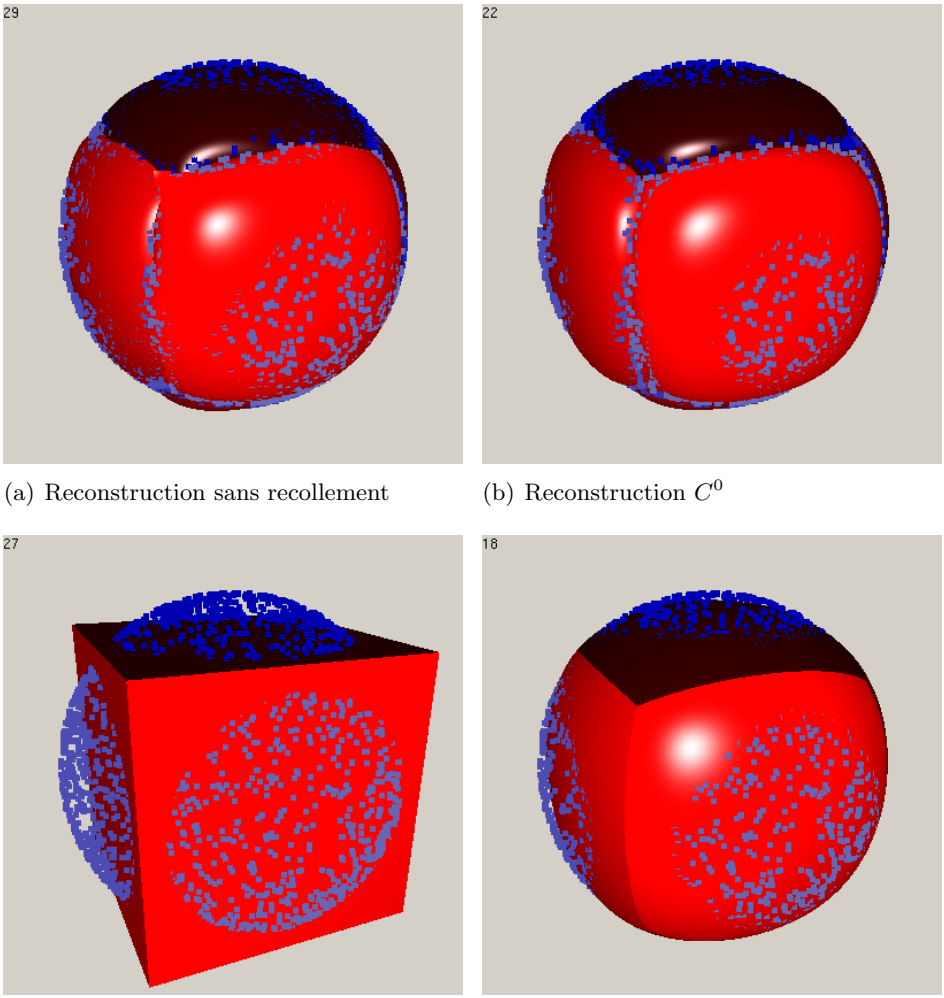
On remarque au premier coup d'œil que la reconstruction C^1 a échoué, et que la situation évoquée à la section 2.3.2.1 est embarrassante puisque la reconstruction d'une sphère avec la régularité C^1 donne... un cube. Certes le faible nombre de points de contrôle y est pour beaucoup, mais ce résultat est tout de même inacceptable.

Comme on pouvait s'y attendre, la reconstruction sans recollement (figure 9.15) laisse apparaître des trous et recouvrements entre chacune des surfaces.

De même, comme en témoigne la figure 9.16 la reconstruction G^1 apporte bien plus de régularité dans l'objet final que le simple recollement. On distingue nettement sur la figure 9.16(a) des repliements sur les arêtes de jonction, tandis que le même objet reconstruit avec un schéma de recollement G^1 (figure 9.16(b)) se comporte bien mieux.

D'un point de vue numérique, les différentes valeurs de la reconstruction de la sphère sans recollement sont reportées dans le tableau 9.1. Les différentes lignes représentent une caractéristique de la surface reconstruite face au nuage de points initial :

– norme inf en x : relevé de l'erreur maximale en x lors du processus de minimisation, c'est la valeur de la fonction objectif.



(a) Reconstruction sans recollement

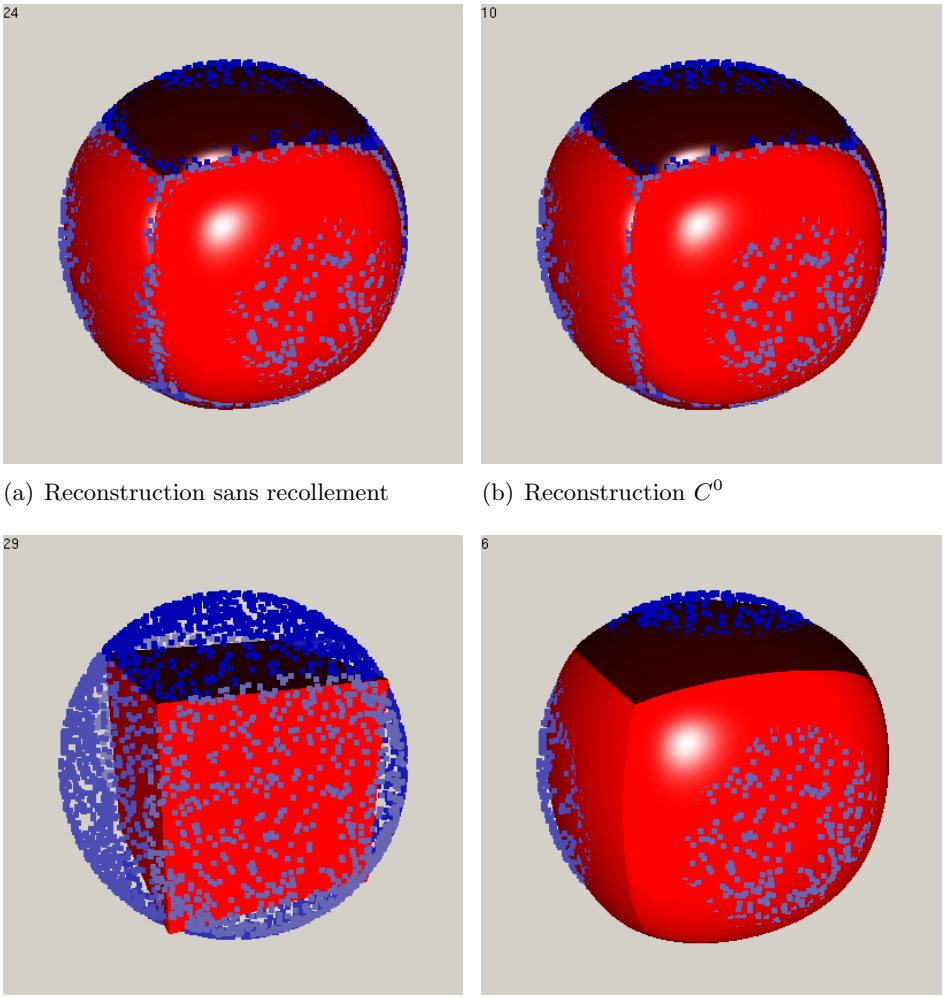
(b) Reconstruction C^0

(c) Reconstruction C^1

(d) Reconstruction G^1

FIGURE 9.13 – Reconstruction de la sphère avec comme fonction objectif le maximum des h_i

- norme 2 en x : moyenne quadratique des écarts en x entre les points de la surface reconstruite et les points initiaux.
- norme 1 en x : moyenne arithmétique des écarts en x entre les points de la surface reconstruite et les points initiaux.
- norme inf en y : relevé de l'erreur maximale en y lors du processus de minimisation, c'est la valeur de la fonction objectif.
- norme 2 en y : moyenne quadratique des écarts en y entre les points de la surface reconstruite et les points initiaux.
- norme 1 en y : moyenne arithmétique des écarts en y entre les points de la surface reconstruite et les points initiaux.



(a) Reconstruction sans recollement

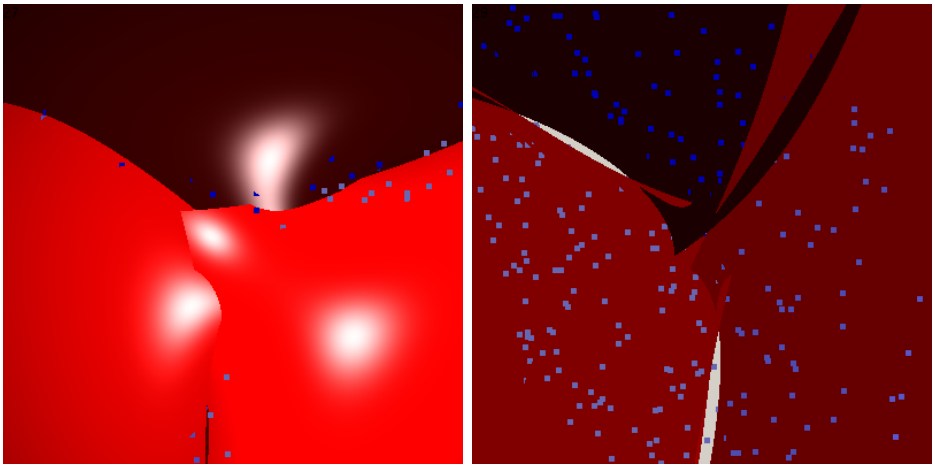
(b) Reconstruction C^0

(c) Reconstruction C^1

(d) Reconstruction G^1

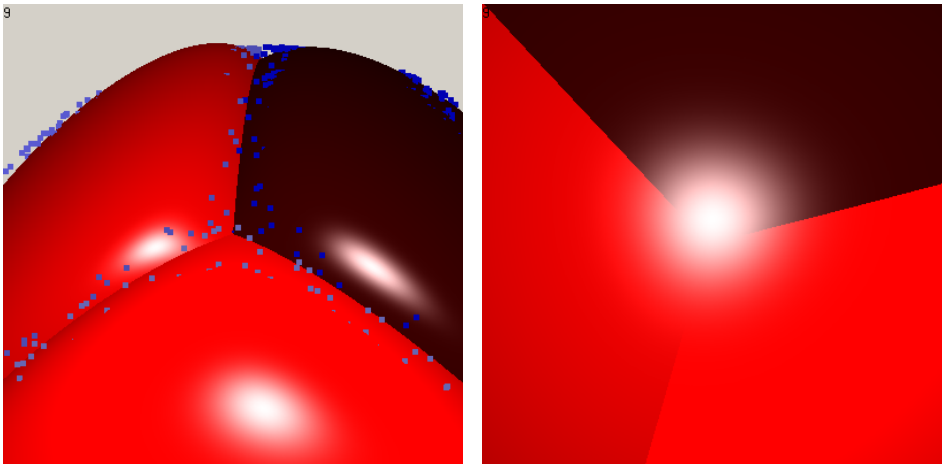
FIGURE 9.14 – Reconstruction de la sphère avec comme fonction objectif la somme des h_i

- norme inf en z : relevé de l'erreur maximale en z lors du processus de minimisation, c'est la valeur de la fonction objectif.
- norme 2 en z : moyenne quadratique des écarts en z entre les points de la surface reconstruite et les points initiaux.
- norme 1 en z : moyenne arithmétique des écarts en z entre les points de la surface reconstruite et les points initiaux.
- max en norme 2 : maximum de l'erreur quadratique entre les points 3D initiaux et le point de la surface reconstruite correspondant.
- moyenne en norme 2 : moyenne arithmétique de l'erreur quadratique entre les points 3D initiaux et le point de la surface reconstruite cor-



(a) Vu de l'extérieur (b) Vu de l'intérieur

FIGURE 9.15 – Trous laissé apparents par la reconstruction sans recollement.



(a) Recollement C^0 vu de près (b) Recollement G^1 vu de près

FIGURE 9.16 – Différences de régularité entre la reconstruction C^0 et la reconstruction G^1.

respondant.

Ce sont ces deux derniers critères qui nous permettent de comparer deux méthodes. En effet, même si nous avons utilisé la norme infinie pour nos calculs, le relevé des vecteurs verts de la section 9.2.2 détermine notre approche.

On remarque sur les tableaux 9.1 que le choix de la fonction objectif est très important, puisque nous l'avons évoqué en section 8.2.2.1, la fonction objectif doit être la somme des h_i. En effet, puisqu'il n'y a pas de recollement, les reconstructions des patches sont indépendantes. Ainsi minimiser

la somme des h_i revient à minimiser indépendamment chacun des h_i.

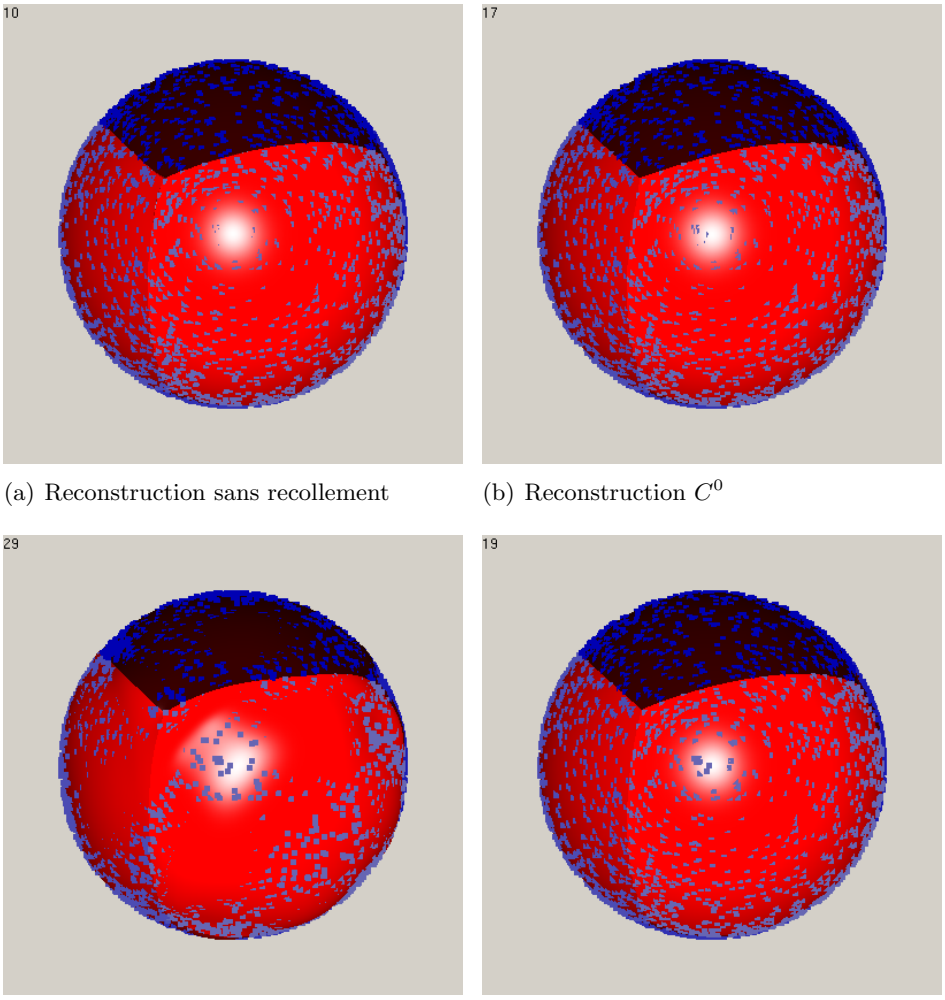
	patch 0	patch 1	patch 2	patch 3	patch 4	patch 5
norme ∞ en x	0.0233528	0.0233528	0.0233529	0.0233528	0.0233528	0.023353
norme 2 en x	0.360769	0.333157	0.336837	0.343509	0.391135	0.385316
norme 1 en x	8.82521	7.87473	7.95042	8.19578	10.0773	9.50521
norme ∞ en y	0.0232085	0.0232081	0.0232092	0.0232084	0.0232086	0.0232087
norme 2 en y	0.401886	0.414731	0.375516	0.346604	0.367189	0.33393
norme 1 en y	10.3747	10.5785	9.44913	8.17858	9.15855	7.71986
norme ∞ en z	0.0228206	0.0228209	0.0228209	0.0228214	0.0228205	0.0228205
norme 2 en z	0.382154	0.37729	0.337751	0.381566	0.341028	0.338624
norme 1 en z	9.83533	9.61719	8.01181	9.52584	8.35999	8.05759
max en norme 2	0.0400591	0.0400589	0.0400591	0.040059	0.0400588	0.0400587
moyenne en norme 2	0.0216133	0.0214012	0.0195119	0.0200739	0.0201808	0.0205908

	patch 0	patch 1	patch 2	patch 3	patch 4	patch 5
norme ∞ en x	0.00985548	0.0102255	0.0102518	0.0101033	0.0233528	0.0224026
norme 2 en x	0.146862	0.150241	0.14871	0.145477	0.391135	0.370489
norme 1 en x	3.55186	3.56792	3.53904	3.39953	10.0773	9.12219
norme ∞ en y	0.0225077	0.0102202	0.0232092	0.0102842	0.0104709	0.0103608
norme 2 en y	0.390841	0.147569	0.375516	0.148275	0.158675	0.149989
norme 1 en y	10.0738	3.42976	9.44913	3.42871	3.8898	3.4872
norme ∞ en z	0.0101705	0.0226085	0.0101717	0.0228214	0.010319	0.00996465
norme 2 en z	0.147632	0.376032	0.14827	0.381566	0.161085	0.144745
norme 1 en z	3.50748	9.58198	3.47285	9.52584	3.92582	3.36666
max en norme 2	0.026592	0.0268225	0.0273344	0.0269924	0.0275942	0.0260535
moyenne en norme 2	0.0140718	0.013943	0.0140007	0.0142571	0.0142738	0.0141081

TABLE 9.1 – Résultats de la reconstruction sans recollement d'une sphère avec des surfaces bicubiques. À gauche, fonction objectif : maximum des h_i. À droite, fonction objectif : somme des h_i

9.3.1.2 Surfaces de degrés supérieurs

Nous avons tenté de reconstruire la sphère avec des surfaces de degré supérieur, présentés sur les figures 9.17 et 9.18. Dans ces figures, les surfaces sont de degrés 6×6 et présentent donc 49 points de contrôle.



(a) Reconstruction sans recollement

(b) Reconstruction C^0

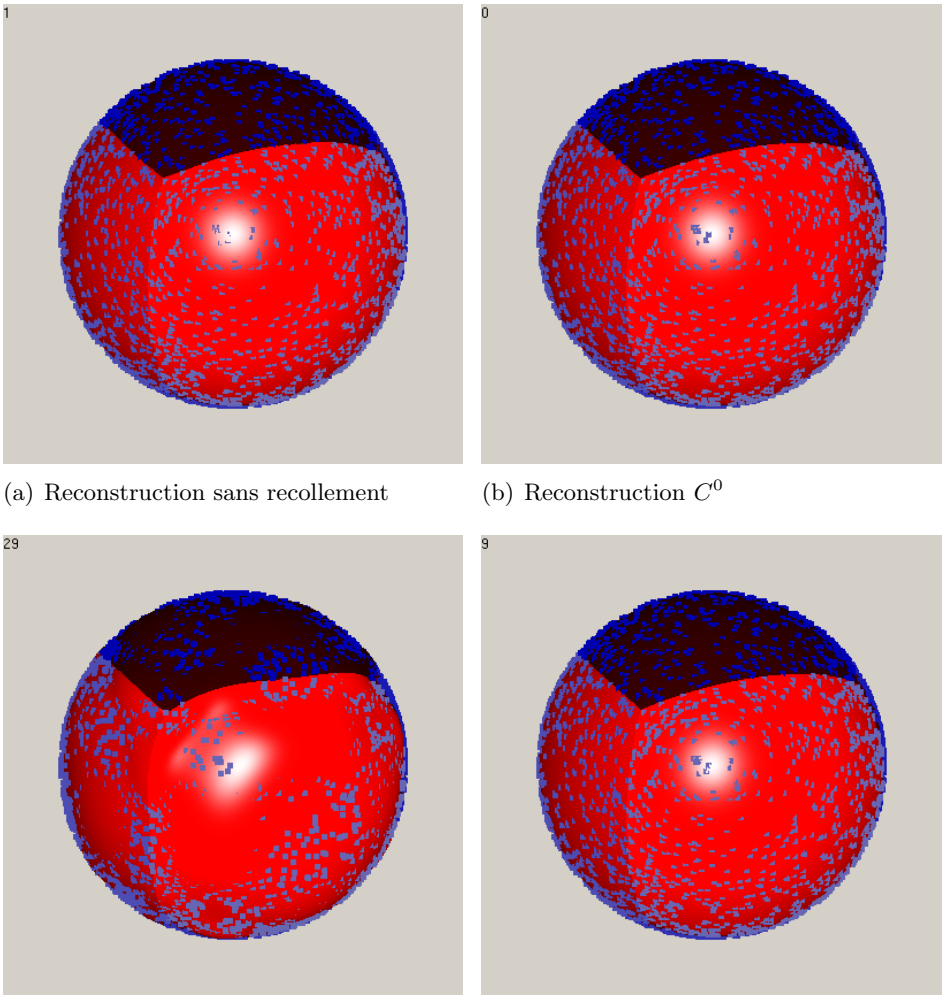
(c) Reconstruction C^1

(d) Reconstruction G^1

FIGURE 9.17 – Reconstruction de la sphère avec comme fonction objectif le maximum des h_i

On remarque que de loin, toutes les reconstructions se valent sauf la reconstruction C^1 sur laquelle la sphère a un aspect bosselé. Vu de plus près, et ce malgré un degré de surface supérieur, les problèmes de reconstruction C^1 sont toujours présents et sont représentés figure 9.19.

Si on étudie les données numériques qui ressortent du tableau 9.2, on remarque que les deux fonctions objectifs se valent, avec une légère avance pour



(a) Reconstruction sans recollement

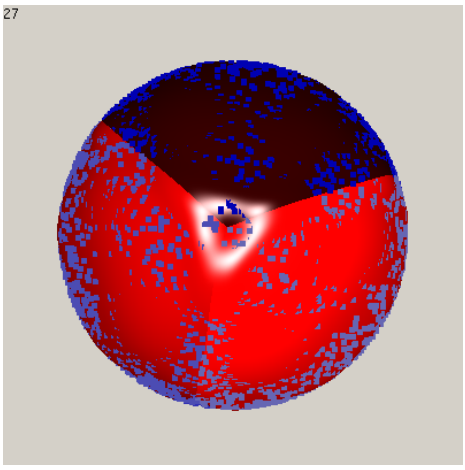
(b) Reconstruction C^0

(c) Reconstruction C^1

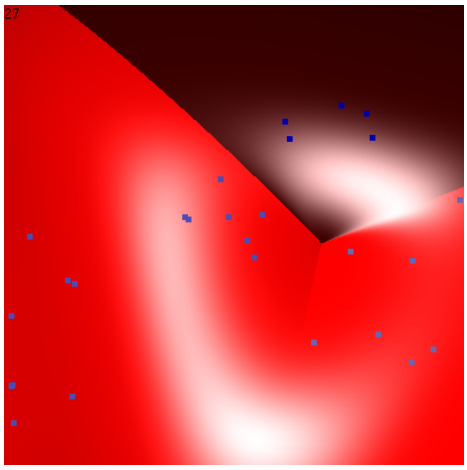
(d) Reconstruction G^1

FIGURE 9.18 – Reconstruction de la sphère avec comme fonction objectif la somme des h_i

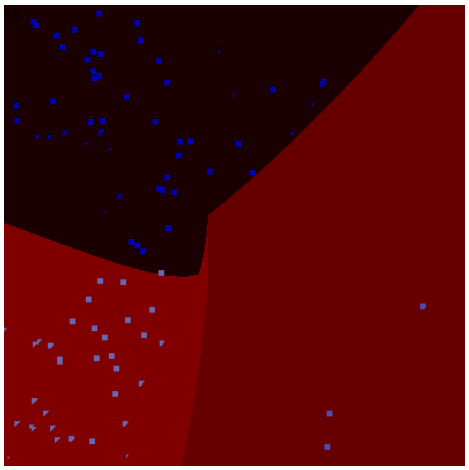
la somme des h_i. Ceci s'explique par l'homogénéïté des morceaux de surfaces à reconstruire, aucun n'ayant plus d'erreur de données que les autres. Cela nous donne donc une erreur plus faible en norme ∞ en x pour les patches 4 et 5, une erreur plus faible en norme ∞ en y pour les patches 0 et 2 et une erreur plus faible en norme ∞ en z pour les patches 1 et 3. Au final, les écarts euclidiens entre la surface et les points reconstruits sont plus faible, aussi bien l'écart maximum que sa moyenne arithmétique.



(a) Vue de l'extérieur



(b) Vue proche de l'extérieur



(c) Vue de l'intérieur. On remarque que le point de jonction des trois patches forme une pointe qui se dirige vers l'intérieur de la sphère

FIGURE 9.19 – Irrégularités dûes à la reconstruction C^1.

	patch 0	patch 1	patch 2	patch 3	patch 4	patch 5
norme ∞ en x	0.00127954	0.00127912	0.00127929	0.00127903	0.00127923	0.00127908
norme 2 en x	0.0227858	0.0233777	0.0225421	0.0222712	0.0242906	0.0224717
norme 1 en x	0.56359	0.615374	0.562592	0.562871	0.63374	0.550428
norme ∞ en y	0.00127701	0.0012768	0.00127713	0.00127707	0.00127663	0.00127668
norme 2 en y	0.0231068	0.0232055	0.0219506	0.0239753	0.0221199	0.0221404
norme 1 en y	0.578253	0.591127	0.552313	0.616788	0.557378	0.532687
norme ∞ en z	0.00128532	0.00128558	0.00128518	0.00128554	0.00128579	0.00128507
norme 2 en z	0.0233219	0.0229483	0.0224156	0.0231579	0.0238003	0.0226129
norme 1 en z	0.582407	0.596794	0.554623	0.589505	0.612818	0.552955
max en norme 2	0.00219672	0.00219195	0.00220469	0.00221292	0.00212178	0.00214913
moyenne en norme 2	0.00133138	0.00129229	0.0012857	0.00131214	0.00131221	0.00132905

	patch 0	patch 1	patch 2	patch 3	patch 4	patch 5
norme ∞ en x	0.00141881	0.00140744	0.00139049	0.0013988	0.000451644	0.000450375
norme 2 en x	0.024424	0.0245148	0.0257923	0.025105	0.00825281	0.00781262
norme 1 en x	0.604469	0.629689	0.659966	0.639292	0.214441	0.192583
norme ∞ en y	0.000474707	0.00139323	0.000489491	0.0013906	0.0014071	0.00140464
norme 2 en y	0.00822203	0.0260257	0.00847973	0.025188	0.0248638	0.0231786
norme 1 en y	0.203977	0.683115	0.213903	0.644997	0.629281	0.555328
norme ∞ en z	0.00141669	0.000447522	0.00143102	0.000451081	0.00143035	0.00140289
norme 2 en z	0.0241126	0.00855645	0.0242477	0.00844874	0.0253573	0.0253812
norme 1 en z	0.598455	0.224128	0.60127	0.219726	0.642986	0.62886
max en norme 2	0.00206026	0.00200886	0.00205403	0.00197324	0.0020061	0.00200306
moyenne en norme 2	0.00117959	0.00116621	0.00121903	0.00118095	0.00115584	0.00120064

TABLE 9.2 – Résultats de la reconstruction G^1 d'une sphère avec des surfaces de degré 6×6. À gauche, fonction objectif : maximum des h_i. À droite, fonction objectif : somme des h_i

9.3.2 Reconstruction de modèles complexes

Nous avons tenté de reconstruire le double tore présenté à la section 9.1. Ce modèle a été quadrangulé selon les trois méthodes présentés à la figure 9.2. Ces méthodes proposent des quadrangulations diverses :

1. Algorithme avec un nombre minimal de quadrangles (voir figure 9.2(c)).
2. Algorithme simple de quadrangulation (voir figure 9.2(a)).
3. Algorithme intermédiaire (voir figure 9.2(b)).

Nous allons détailler les résultats de chacune de ces méthodes sur le maillage du double tore.

9.3.2.1 Reconstruction avec un nombre minimal de quadrangles

Nous pouvons d'ores et déjà affirmer que dans un processus de reconstruction, la méthode 1 ne donne pas de résultats probants. En effet, la forme étroite des quadrangles ne favorise pas la reconstruction. Sur le résultat présenté à la figure 9.20, les surfaces paramétrées se ne comportent pas bien du tout. Ce comportement était prévisible et nous avions déjà abordé le sujet en section 9.1.

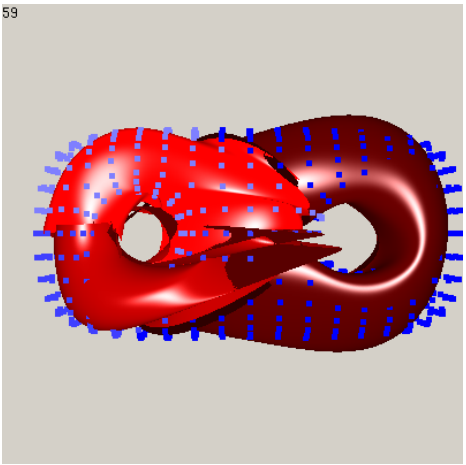
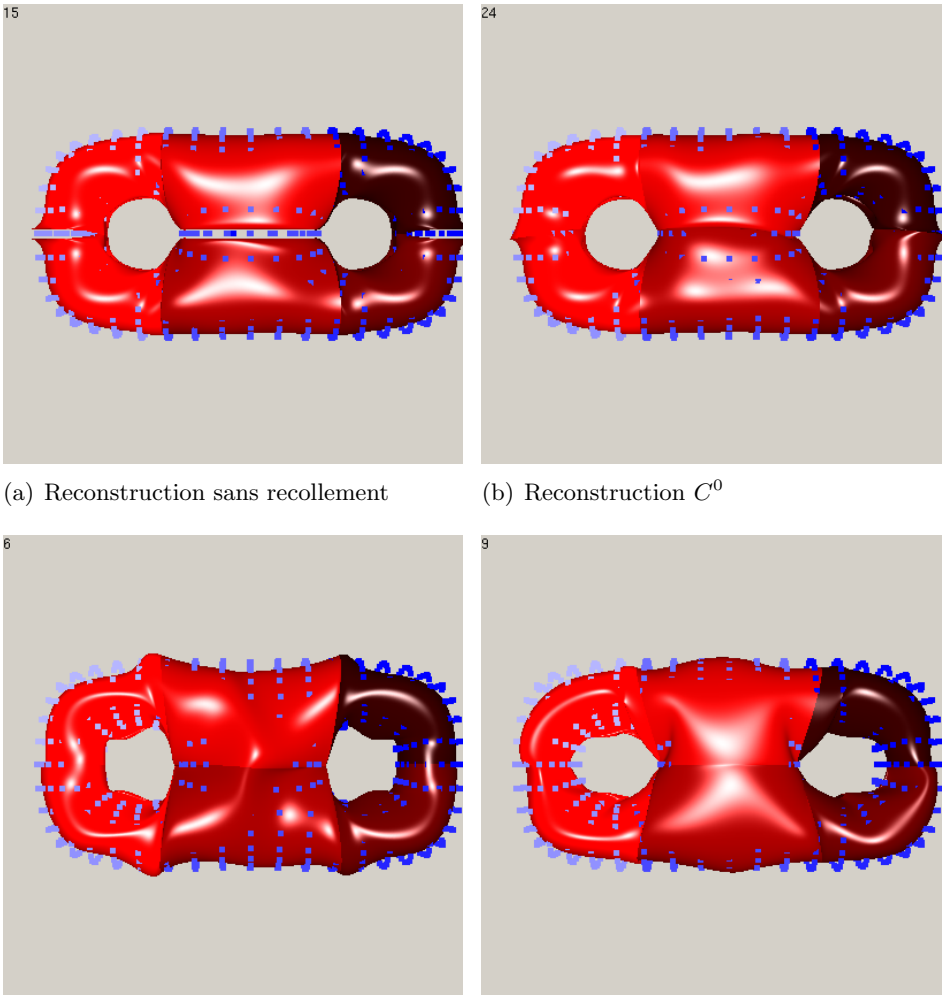


FIGURE 9.20 – Reconstruction sans recollement du double tore quadrangulé à l'aide de la méthode 1 avec des surfaces de Bézier de degrés 5×5 . La fonction objectif est la somme des h_i

9.3.2.2 Reconstruction avec l'algorithme simple de quadrangulation

Cette quadrangulation appliquée au double tore nous donne 6 surfaces de Bézier à reconstruire. Nous présentons des résultats de reconstruction



(a) Reconstruction sans recollement

(b) Reconstruction C^0

(c) Reconstruction C^1

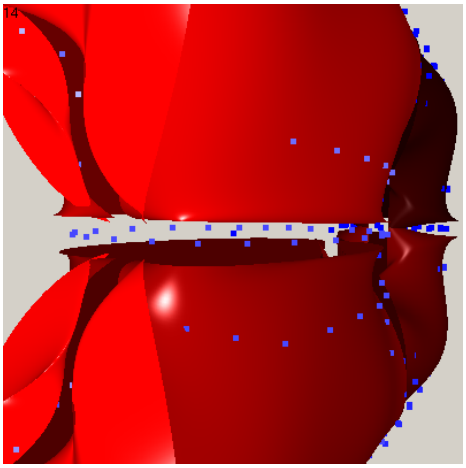
(d) Reconstruction G^1

FIGURE 9.21 – Reconstruction du double tore (6 surfaces de Bézier 5×5) avec comme fonction objectif la somme des h_i

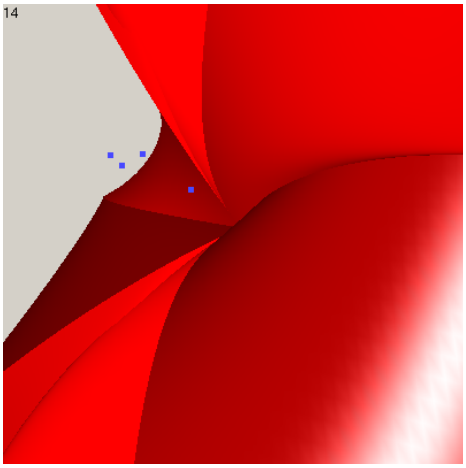
à la figure 9.21. Sur cette figure, contrairement à la partie précédente, ce qui saute aux yeux est l'aspect cabossé de presque toutes les surfaces. C'est normal, chaque surface ayant entre 150 et 180 points initiaux et seulement 36 points de contrôle.

La reconstruction sans recollement laisse clairement apparaître les lieux de découpe (détails figure 9.22(a)). Le défaut de la reconstruction C^0 et de la reconstruction C^1 se situe au niveau des jonctions de surfaces, des repliements parasites que l'on aperçoit figures 9.22(b) et 9.22(c).

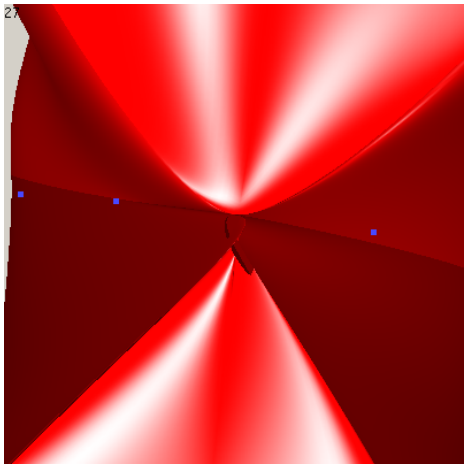
Le cas de la reconstruction G^1 est plus intéressant. On y voit une reconstruction parfaite au niveau des jonctions, comme nous l'avons spécifié. Par



(a) Vue approchée de la reconstruction sans recollement



(b) Vue approchée de la reconstruction C^0

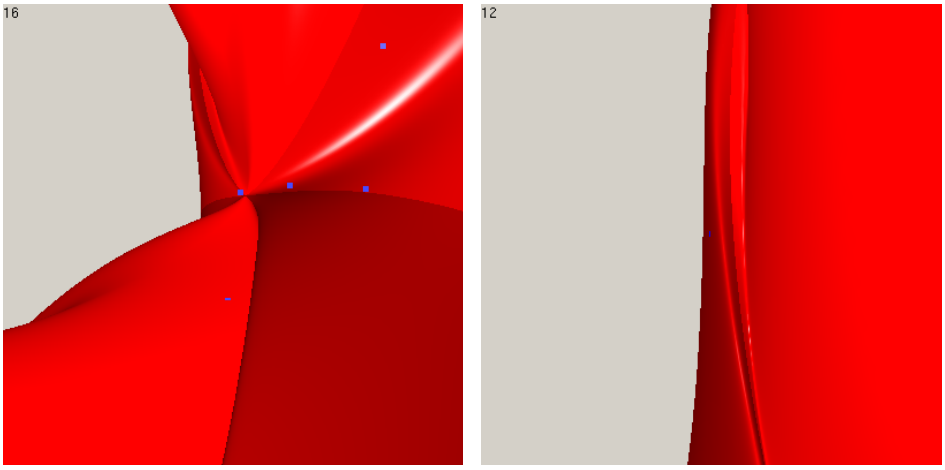


(c) Vue approchée de la reconstruction C^1

FIGURE 9.22 – Reconstruction du double tore (6 surfaces de Bézier 5×5) avec comme fonction objectif la somme des h_i (vue approchée)

contre, il y a un effet inattendu de repliement de surface (hors des jonctions). Ce repliement, présenté à la figure 9.23(b) est gênant. Cet effet pourrait être supprimé si l'on introduit lors de la reconstruction des contraintes de courbure.

D'un point de vue numérique, nous avons comparé les deux types de fonction objectif dans le tableau 9.3. Aucune des deux fonctions objectif ne tire son épingle du jeu.



(a) Vue approchée de la reconstruction G^1 (b) Vue approchée de la reconstruction G^1, des plis apparaissent sur la surface

FIGURE 9.23 – Reconstruction G^1 du double tore (6 surfaces de Bézier 5×5) avec comme fonction objectif la somme des h_i (vue approchée)

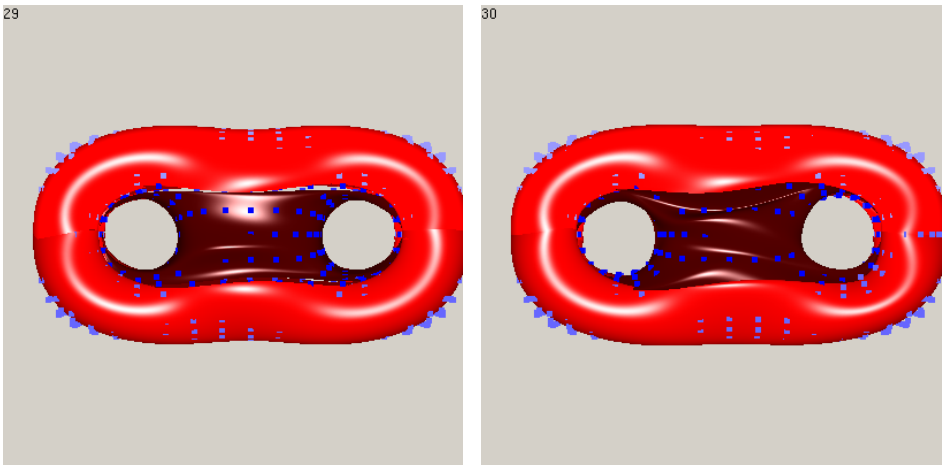
	patch 0	patch 1	patch 2	patch 3	patch 4	patch 5
norme ∞ en x	0.0737624	0.0737628	0.0635132	0.0635132	0.0738331	0.0738327
norme 2 en x	0.587705	0.587477	0.471031	0.470741	0.608627	0.607391
norme 1 en x	6.46137	6.46553	4.55153	4.54488	6.67934	6.65671
norme ∞ en y	0.0542403	0.054241	0.0427417	0.0427417	0.0542408	0.0542413
norme 2 en y	0.405408	0.392113	0.344225	0.344231	0.405364	0.3921
norme 1 en y	4.17524	4.03768	3.65833	3.6584	4.17454	4.03736
norme ∞ en z	0.086395	0.086395	0.0863964	0.086395	0.086397	0.0863918
norme 2 en z	0.79209	0.731102	0.754994	0.717941	0.745663	0.780275
norme 1 en z	9.39192	8.5079	8.10591	7.67277	8.52061	9.07082
max en norme 2	0.125304	0.12466	0.113474	0.110499	0.125923	0.120832
moyenne en norme 2	0.0761246	0.0723658	0.0728992	0.0704787	0.0745693	0.0752697

	patch 0	patch 1	patch 2	patch 3	patch 4	patch 5
norme ∞ en x	0.0736737	0.0736736	0.073674	0.0736737	0.0736739	0.0736733
norme 2 en x	0.580874	0.571688	0.482367	0.533478	0.573327	0.579189
norme 1 en x	6.18777	6.0929	4.80116	5.41391	6.12715	6.18707
norme ∞ en y	0.0538325	0.053832	0.053832	0.0538325	0.0538316	0.0538318
norme 2 en y	0.396122	0.395519	0.419443	0.418146	0.396142	0.395846
norme 1 en y	4.27456	4.26191	4.37767	4.36425	4.27185	4.26557
norme ∞ en z	0.0863945	0.0863955	0.0863943	0.0863943	0.0863929	0.0863928
norme 2 en z	0.761441	0.72962	0.710536	0.693635	0.779801	0.793476
norme 1 en z	8.73499	8.27827	7.64672	7.41422	9.29155	9.39698
max en norme 2	0.119567	0.125654	0.119051	0.113542	0.125654	0.116142
moyenne en norme 2	0.0737778	0.0709824	0.0737189	0.0756789	0.0745417	0.0754214

TABLE 9.3 – Résultats de la reconstruction G^1 du double tore avec des surfaces de degré 5×5. À gauche, fonction objectif : somme des h_i. À droite, fonction objectif : maximum des h_i

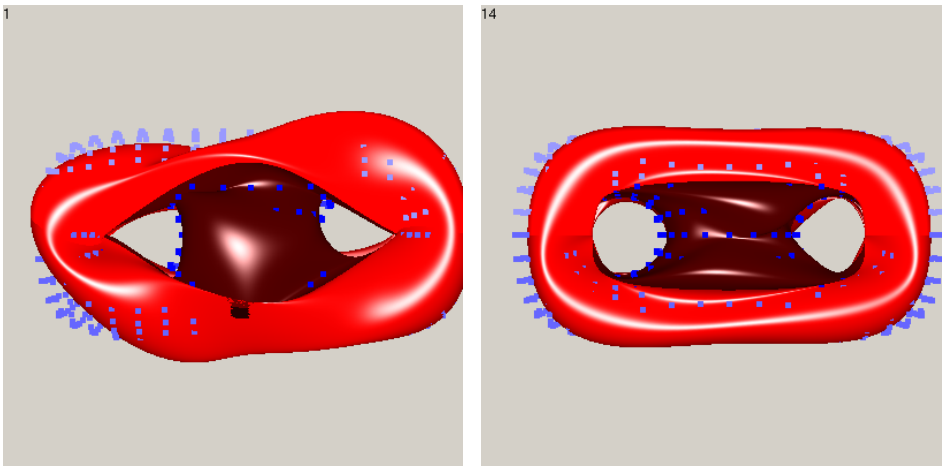
9.3.2.3 Reconstruction avec l'algorithme intermédiaire de quadrangulation

Cette section prend comme base de quadrangulation la méthode 3 de quadrangulation.



(a) Reconstruction sans recollement

(b) Reconstruction C^0

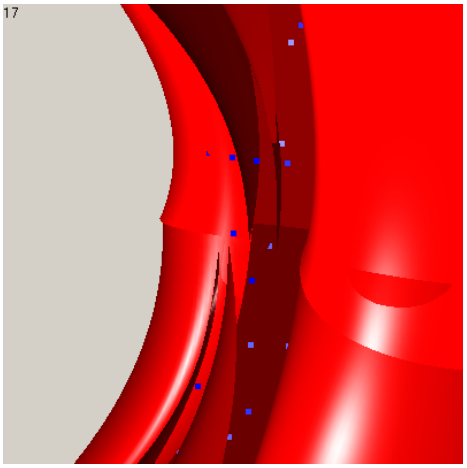


(c) Reconstruction C^1

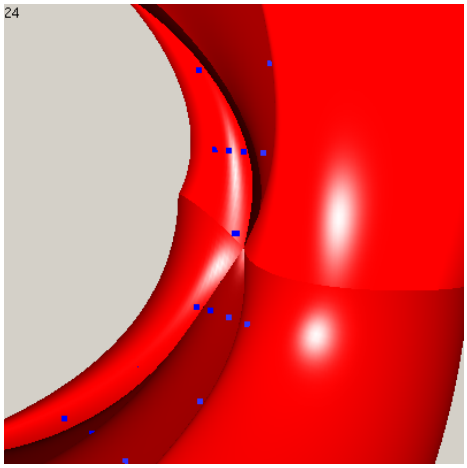
(d) Reconstruction G^1

FIGURE 9.24 – Reconstruction du double tore (4 surfaces de Bézier 5×5) avec comme fonction objectif le maximum des h_i

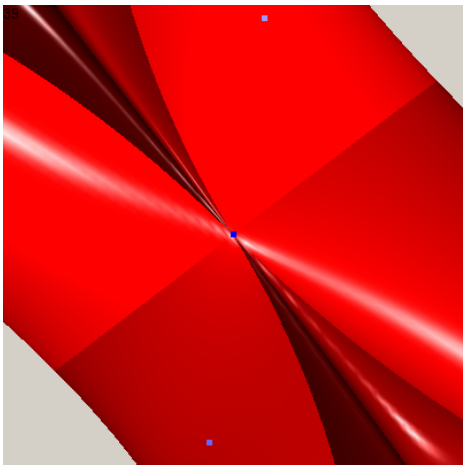
Mêmes remarques que précédemment, les reconstructions se comportent comme prévu, compte tenu du faible nombre (seulement 4) de surfaces, et de leur faible degré. On notera que pour les deux sommets, la jonction contient 8 patches. Nous sommes dans les conditions de recollement C^1. Nous voyons bien, et nous l'avions déjà évoqué à la section 2.3.2.1, que les points de contrôles sont tous confondu et forment un point stationnaire.



(a) Reconstruction sans recollement



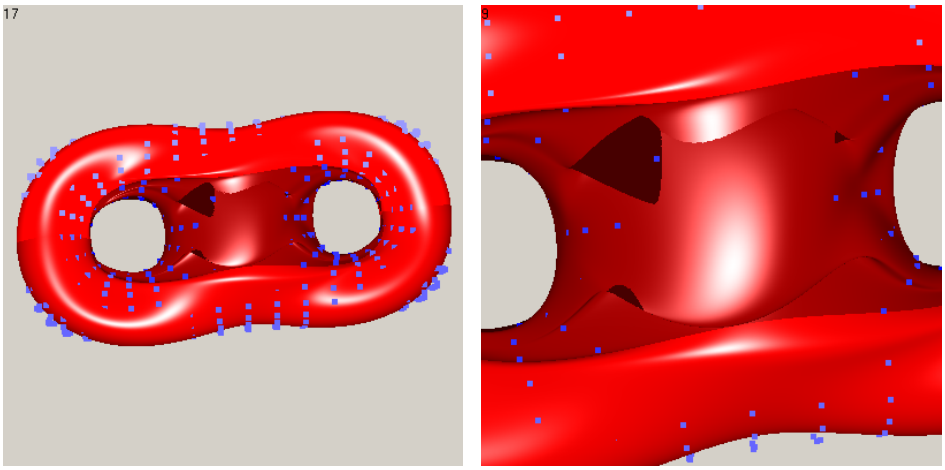
(b) Reconstruction C^0



(c) Reconstruction G^1

FIGURE 9.25 – Reconstruction du double tore (4 surfaces de Bézier 5×5) avec comme fonction objectif le maximum des h_i. Vue de près d'un sommet

On n'observe plus le phénomène de repliement constaté à la section précédente. Ceci est du à la forme complètement différente de la segmentation et qui change complètement la donne. Par contre, si l'on augmente le degré des surfaces de Bézier, on observe des phénomènes assez incongrus, comme illustrés figure 9.26. Les deux surfaces centrales sont tellement contraintes à cause de l'éloignement des coins, qu'elles laissent apparaître des recouvrements et des pincements.



(a) Reconstruction G^1 (b) Reconstruction G^1 vue de près

FIGURE 9.26 – Reconstruction du double tore (4 surfaces de Bézier 7×7) avec comme fonction objectif la somme des h_i. Le degré est trop fort, il y a des oscillations.

9.3.2.4 Discussion

On pourrait débattre de la meilleure solution de quadrangulation pour reconstruire une surface. Malheureusement, il n'y a pas de solution miracle, il n'y a que des solutions qui s'adaptent au contexte. En effet, quelle que soit la méthode de quadrangulation employée, quelle que soit la régularité de la reconstruction choisie, quelle que soit la dimension des surfaces à reconstruire, ou encore quelle que soit la fonction objectif choisie, il est primordial de garder à l'esprit que seuls les résultats visuels comptent.

Nous avons vu que la méthode 1 convenait pour le double tore. Mais nous avons vu qu'elle était insuffisante pour le bretzel de la figure 9.3. Il faut donc étudier au cas pas cas. De même, le nombre de points de contrôle des surfaces à reconstruire est à déterminer avec attention, et si la reconstruction donne des résultats numériques satisfaisants, il n'en est pas toujours de même pour l'aspect visuel de la surface.

9.3.3 Difficultés de résolution

La dimension du programme linéaire est importante. En effet, le programme linéaire contient deux contraintes d'approximation pour chacun des points initiaux M_i. Le nombre de variables est égal à la dimension de la base de fonction (25 pour une surface de Bézier de degrés 4×4) de chacun des patches. Même si les contraintes linéaires liant les points de contrôle font baisser la complexité du programme, cela reste un programme linéaire long à résoudre et nous sommes très loin d'une reconstruction temps réel.

De plus, il y a trois programmes linéaires à résoudre. Et, de même qu'à la section 9.2.4, il est préférable de ne pas regrouper ces trois programmes en un seul.

9.3.4 Améliorations possibles

Parmi les améliorations possibles, on peut inclure des contraintes de courbure à l'intérieur des patches si celles-ci sont linéaires. Cependant comme ces contraintes s'appliquent le plus souvent à chaque point, la taille du programme linéaire explosera.

La plupart des contraintes de courbures sont issues de l'énergie de déformation. Cependant, le problème de la minimisation de cette énergie est quadratique. De plus, il est nécessaire de déterminer le critère de régularisation associé [BBMN08]. En effet, le double problème de minimisation qui comprend l'attache aux données et la régularité de la surface, est un problème à deux fonctions objectifs contradictoires. Le premier fera en sorte que le courbe soit le plus près possible des données, avec éventuellement de fortes distorsions de la surface, et le second critère fera tendre la surface vers un plan, au détriment de l'approximation des données. Dans ce cas précis,

la détermination du critère est primordiale et peut, ou non, faire partie du processus d'optimisation.

D'autres améliorations sont encore possibles. La première est de permettre de choisir la type de reconstruction à appliquer à chacune des arêtes. Ceci nécessite une intervention de l'utilisateur, mais est primordiale si l'on souhaite le meilleur résultat possible. Nous n'avons pas encore expérimenté la seconde qui est le recollement choisi, qui peut s'avérer plus utile dans certains cas de jonction difficile.

Conclusion et Perspectives

Ce document présente un cheminement de reconstruction, depuis un nuage de points ou un maillage vers un ensemble de surfaces paramétrées. La reconstruction est un passage important entre une modélisation filaire d'un objet et une modélisation qui se prête beaucoup mieux au rendu, à la compression ou à l'usinage. Beaucoup d'algorithmes de reconstruction se basent sur une décomposition du problème en plusieurs étapes. Nous avons suivi le même principe en détaillant les étapes qui composent ce processus : un éventuel maillage, une segmentation du maillage, paramétrisation de chacun des morceaux de maillages et enfin reconstruction en tenant compte des contraintes éventuelles de recollement. Ce processus comporte de nombreuses techniques dont il faut maîtriser les tenants et aboutissants afin pour pouvoir évaluer le fonctionnement de notre méthode de reconstruction. Ces techniques sont en constante évolution encore aujourd'hui.

Le principal apport de cette thèse se situe dans l'utilisation de la programmation linéaire. Les techniques habituelles de reconstruction confortent la prédominance de la reconstruction aux moindres carrés. Même si nous n'évaluons pas le résultat final avec la norme uniforme, son utilisation dans les minimisations coordonnées par coordonnées n'est pas inintéressante en formant une sorte de « bounding box » autour du point à reconstruire. Comme les contraintes de recollement sont linéaires, la programmation linéaire permet simplement de les ajouter au programme linéaire. On intègre les contraintes de recollement que nous avons détaillées et le recollement est pris en compte. Une contribution non négligeable est la définition du recollement choisi. Nous avons défini une carte de recollement qui permet de positionner les points de contrôle autour d'un sommet lors d'une reconstruction G^1. Ce positionnement est fait grâce à l'attribution de valeurs aux coefficients α et β de chaque arête de recollement conformes à la carte de recollement donnée.

Le second apport de cette thèse se situe dans le cadre de pavages. Avec Jean-Marie Favreau, Vincent Barra et Yan Gérard, lorsque nous avons voulu mettre en commun nos travaux, ou plutôt bout à bout, nous avons défini un cadre pour le pavage de surfaces. Les M-pavages avec des M-tuiles que nous avons définis permettent de décrire de bout en bout les étapes successives nécessaires au découpage/pavage de maillages, et sont un moyen d'inclure toutes les informations.

Les perspectives de ce travail sont nombreuses. En premier lieu, et concernant le recollement, il y a trois points principaux à développer. Il faudra déterminer une carte de recollement à partir des données du maillage, implémenter ce recollement choisi correspondant, pour que nous puissions voir quelles sont les avancées qui sont apportées. Une étude plus approfondie

des recollements est à faire. Il serait notamment très intéressant de pouvoir choisir le type de recollement de chacune des arêtes, laissant un recollement C^0 là ou le G^1 est trop contraignant. Ce type de recollement peut devenir très intéressant si l'on souhaite reconstruire des objets où des arêtes sont apparentes. Enfin, pour conclure les perspectives concernant le recollement, nous devons réfléchir à l'inclusion de contraintes de courbure linéaires lors de la reconstruction.

Les autres perspectives concernent la segmentation de maillages. Beaucoup de possibilités sont disponibles, il y a tellement de choses à faire et à étudier, et en particulier pour la quadrangulation qui nous intéresse plus particulièrement. Afin de pouvoir effectuer une reconstruction C^1, il serait souhaitable d'avoir un graphe de recollement qui s'y prête. Cependant, garantir un tel graphe requiert un certain nombre d'informations topologiques au départ. On aimerait beaucoup contrôler le nombre de patches obtenus. Nous avons donné le nombre de quadrangles minimal d'un maillage de genre g, mais nous n'avons pas de moyen de déterminer une quadrangulation à N patches, où N serait donné au préalable. Pourtant, ce serait un moyen de contrôler le résultat final de reconstruction. De même, nous n'avons aucun moyen de contrôler le nombre de points à l'intérieur d'un patch. Cela pose des problèmes d'homogénéité de la surface produite. Toutes ces raisons poussent à continuer le travail dans la segmentation de maillages.

Enfin, pour les perspectives un peu plus générales de ce travail, il y a l'utilisation des surfaces de subdivision. Ces surfaces sont de plus en plus utilisées et tendent à devenir le standard de la modélisation géométrique, d'autant plus que le continuité fait partie intégrante de leur définition. Il serait donc intéressant de voir si une résolution par programme linéaire est adaptable et quels résultats elle offre.

Bibliographie

[ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, New York, NY, USA, 1998. ACM Press.

[AKTL06] Benjamin Albouy, Emilie Koenig, Sylvie Treuillet, and Yves Lucas. Accurate 3d structure measurements from two uncalibrated views. In *ACIVS*, pages 1111–1121, 2006.

[Ale02] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21 Part 2 :173–196, 2002.

[All07] P. Alliez. Quadrangle surface tiling through contouring. *LECTURE NOTES IN COMPUTER SCIENCE*, 4647 :29, 2007.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows : theory, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[Aur91] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3) :345–405, 1991.

[AW04] A. Atieg and G. A. Watson. Use of l_p norms in fitting curves and surfaces to data. In Jagoda Crawford and A. J. Roberts, editors, *Proc. of 11th Computational Techniques and Applications Conference CTAC-2003*, volume 45, pages C187–C200, April 2004.

[BBCS99] Fausto Bernardini, Chandrajit L. Bajaj, Jindong Chen, and Daniel Schikore. Automatic reconstruction of 3d CAD models from digital scans. *International Journal of Computational Geometry and Applications*, 9(4/5) :327–369, 1999.

[BBMN08] F. Brunet, A. Bartoli, R. Malgouyres, and N. Navab. L-tangent norm : A low computational cost criterion for choosing regularization weights and its use for range surface reconstruction. *Proceedings of 3D Data Processing, Visualization and Transmission*, 2008.

[BBX95] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics*, 29(Annual Conference Series) :109–118, 1995.

[BC00] Jean-Daniel Boissonnat and Frederic Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Symposium on Computational Geometry*, 2000.

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4) :469–483, 1996.

[Ber08] M. Berkelaar. Lp solve : Linear programming code, Dernière visite le 18/07/2008. http ://www.cs.sunysb.edu/algorith/implement/lpsolve/implement.shtml.

[BMRJ04] Ioana Boier-Martin, Holly Rushmeier, and Jingyi Jin. Parameterization of triangle meshes over quadrilateral domains. In *SGP '04 : Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 193–203, New York, NY, USA, 2004. ACM.

[BR73] I. Barrodale and FDK Roberts. An improved algorithm for discrete l 1 linear approximation. *SIAM J. Numer. Anal*, 10(5) :839–848, 1973.

[Cay03] François Cayre. *Contributions au tatouage des maillages surfaciques 3D*. PhD thesis, ENST/TSI et UCL/TELE, 2003.

[cga08] CGAL, Computational Geometry Algorithms Library, Dernière visite le 16/07/2008. http ://www.cgal.org.

[CMS98] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1) :37–54, 1998.

[CRE01] Elaine Cohen, Richard F. Riesenfield, and Gershon Elber. *Geometric Modeling with Splines : An Introduction*. A K Peters, 2001.

[CS03] C.R. Collins and K. Stephenson. A circle packing algorithm. *Computational Geometry : Theory and Applications*, 25(3) :233–256, 2003.

[Cur97] Brian Lee Curless. *New Methods for Surface Reconstruction from Range Images*. PhD thesis, STANFORD UNIVERSITY, 1997.

[CV83] Chvatal and Vasek. *Linear Programming*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York, 1983.

[DBC07] Sébastien Delest, Romuald Boné, and Hubert Cardot. État de l'art de la segmentation de maillage 3d par patchs surfaciques. In *GTMG : Groupe de Travail en Modélisation Géométrique*, Valenciennes 2007.

[DBG+06] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3) :1057–1066, 2006.

[Del34] B. Delaunay. Sur la sphere vide. a la memoire de georges voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7 :793–800, 1934.

[Del94] Hervé Delingette. *Modélisation, déformation et reconnaissance d'objets tridimensionnels à l'aide de maillages simplexes*. PhD thesis, École Centrale de Paris, 1994.

[Die78] J.A. Dieudonné. *Abrégé d'histoire des mathématiques : 1700-1900*. Hermann, Paris, 1978.

[DMA02] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3) :209–218, 2002.

[DOW55] G.B. Dantzig, A. Orden, and P. Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2) :183–195, 1955.

[Du88] W.H. Du. Etude sur la representation de surfaces complexes : application a la reconstruction de surfaces echantillonnees. *These, Department Image, TELECOM, Paris*, 1988.

[ECDVL07] Éric Colin De Verdière and Francis Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *J. ACM*, 54(4) :18, 2007.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer Graphics*, 29(Annual Conference Series) :173–182, 1995.

[EH96] Matthias Eck and Hugues Hoppe. Automatic reconstruction of B-Spline surfaces of arbitrary topological type. In *ACM SIGGRAPH 96*, pages 325–334, 1996.

[EHNP03] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-smale complexes for piecewise linear 3-manifolds. *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 361–370, 2003.

[EHP04] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *Discrete and Computational Geometry*, 31(1) :37–59, 2004.

[EM94] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1) :43–72, 1994.

[Far02] Gerald Farin. *Curves and surfaces for CAGD : a practical guide*. Morgan Kaufmann Publishers Inc., 2002.

[FGK+00] A. Fabri, G.J. Giezeman, L. Kettner, S. Schirra, and S. Schoenherr. On the design of cgal a computational geometry algorithms library. *Software Practice and Experience*, 30(11) :1167–1202, 2000.

[FH05] Michael S. Floater and Kai Hormann. Surface parameterization : a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[FK97] A.T. Fomenko and T.L. Kunii. *Topological modeling for visualization*. Springer-Verlag, 1997.

[Flo97] M.S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3) :231–250, 1997.

[Flo03] M.S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1) :19–27, 2003.

[FMR+02] Jordi Freixenet, Xavier Muñoz, D. Raba, Joan Martí, and Xavier Cufí. Yet another survey on image segmentation : Region and boundary information integration. In *ECCV '02 : Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 408–422, London, UK, 2002. Springer-Verlag.

[GD06] E. Grinspun and M. Desbrun. Discrete differential geometry : an applied introduction. *International Conference on Computer Graphics and Interactive Techniques*, 2006.

[Guo97] Baining Guo. Surface reconstruction : from points to splines. *Computer-Aided Design*, 29(4) :269–277, 1997.

[Hah06] S. Hahmann. *Mémoire d'Habilitation*. PhD thesis, INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, 2006.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2) :71–78, 1992.

[HDD+93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26, 1993.

[Het03] F. Hetroy. *Méthodes de partitionnement de surfaces*. PhD thesis, Institut National Polytechnique de Grenoble, 2003.

[HI97] A. Hilton and J. Illingworth. Marching triangles : Delaunay implicit surface triangulation. *University of Surrey*, 1997.

[HLS07] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization : Theory and practice. In *SIGGRAPH 2007 Course Notes*, volume 2, pages 1–122, San Diego, CA, August 2007. ACM Press.

[HS89] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406) :502–516, 1989.

[HU02] Jean-Baptiste Hiriart-Urruty. L'optimisation : deux ou trois choses que je sais d'elle. Technical report, Centre national d'études spatiales, Toulouse, FRANCE, 2002.

[Jüt97] Bert Jüttler. Surface fitting using convex tensor-product splines. *J. Comput. Appl. Math.*, 84(1) :23–44, 1997.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4) :373–395, 1984.

[Kov89] Vladimir A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 46(2) :141–161, May 1989.

[KT50] HW Kuhn and AW Tucker. Nonlinear programming. *Proc. Berkeley Sympos. math. Statist. Probability, California July 31-August 12,*, page 31, 1950.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes : a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4) :163–169, july 1987.

[Lev44] K. Levenberg. A method for the solution of certain problems in least squares. In *The Quarterly of Applied Mathematics 2*, volume 2, 1944.

[LM00] B. Levy and J.L. Mallet. Paramétrisation des surfaces triangulées. *International Journal of CADCAM and Computer Graphics*, 15(1), 2000.

[LPRM02] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3) :362–371, 2002.

[LS80] DT Lee and BJ Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9(3) :219–242, 1980.

[LS86] Peter Lancaster and Kestutis Salkauskas. *Curve and Surface Fitting : An Introduction.* Academic Press, London, England, 1986.

[Mak04] A. Makhorin. Glpk (gnu linear programming kit). *Free Software Foundation, January*, 2004.

[Mar63] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11 :431–441, 1963.

[MBVW95] MJ Milroy, C. Bradley, GW Vickers, and DJ Weir. G1 continuity of b-spline surface patches in reverse engineering. *Computer-Aided Design*, 27(6) :471–478, 1995.

[MGM06] Thibault Marzais, Yan Gérard, and Rémy Malgouyres. Lp fitting approach for reconstructing parametric surfaces from points clouds. In *GRAPP*, pages 325–330, 2006.

[Mil64] J. Milnor. On the Betti numbers of real varieties. *Proc. Amer. Math. Soc*, 15(2) :275–280, 1964.

[NH91] Gregory M. Nielson and Bernd Hamann. The asymptotic decider : resolving the ambiguity in marching cubes. In *VIS '91 : Proceedings of the 2nd conference on Visualization '91*, pages 83–91, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[OF03] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[Pet94] J. Peters. Constructing c 1 surfaces of arbitrary topology using biquadratic and bicubic splines. *Designing Fair Curves and Surfaces*, pages 277–293, 1994.

[Pet95] J. Peters. Biquartic c1-surface splines over irregular meshes. *Computer-Aided Design*, 27(12) :895–903, 1995.

[Pet02] J. Peters. Geometric continuity. *Handbook of Computer Aided Geometric Design*, pages 193–229, 2002.

[PP93] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1) :15–36, 1993.

[RS87] B. Rodin and D. Sullivan. The convergence of circle packings to the riemann mapping. *J. Differential Geom*, 26(2) :349–360, 1987.

[RT95] JM Reddy and GM Turkiyyah. Computation of 3d skeletons using a generalized delaunay triangulation technique. *Computer-Aided Design*, 27(9) :677–694, 1995.

[SdS01] A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers*, 17 :326–337, 2001.

[Ser53] J.P. Serre. Groupes d'homotopie et classes de groupes abeliens. *Ann. of Math*, 58(2) :258–294, 1953.

[SH02] Alla Sheffer and John C. Hart. Seamster : inconspicuous low-distortion texture seam layout. In *VIS '02 : Proceedings of the conference on Visualization '02*, pages 291–298, Washington, DC, USA, 2002. IEEE Computer Society.

[SLMB05] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov. Abf++ : fast and robust angle based flattening. *ACM Transactions on Graphics (TOG)*, 24(2) :311–330, 2005.

[SM91] Biplab Sarkar and Chia-Hsiang Menq. Smooth-surface approximation and reverse engineering. *Computer-Aided Design*, 23(9) :623–628, 1991.

[SPR06] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2) :105–171, 2006.

[sta08] Stanford repository, Dernière visite le 04/03/2008. http ://graphics.stanford.edu/data/3Dscanrep/.

[Ste03] K. Stephenson. Circle packing : a mathematical tale. *Notices Amer. Math. Soc*, 50(11) :1376–1388, 2003.

[TACSD06] Y. Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. Designing quadrangulations with discrete harmonic forms. In *Symposium on Geometry Processing*, pages 201–210, 2006.

[TG98] Costa Touma and Craig Gotsman. Triangle mesh compression. In *Graphics Interface, Vancouver*, 1998.

[TL94] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH*, pages 311–318, 1994.

[Tut60] WT Tutte. Convex representations of graphs. *Proceedings of the London Mathematical Society*, 3(1) :304, 1960.

[Tut63] W.T. Tutte. How to draw a graph. *Proc. London Math. Soc*, 13(3) :743–768, 1963.

[VMA86] B C Vemuri, A Mitiche, and J K Aggarwal. Curvature-based representation of objects from range data. *Image Vision Comput.*, 4(2) :107–114, 1986.

[WARV02] V. Weiss, L. Andor, G. Renner, and T. Várady. Advanced surface fitting techniques. *Comput. Aided Geom. Des.*, 19(1) :19–42, 2002.

[Whi49] JHC Whitehead. Combinatorial homotopy. *Bull. Amer. Math. Soc*, 55(3), 1949.

[WR67] E. T. Whittaker and G. Robinson. *The Method of Least Squares. Ch. 9 in The Calculus of Observations : A Treatise on Numerical Mathematics.* Dover Publications, 4th edition, 1967.

[WR89] RJ Watt and BJ Rogers. Human Vision and Cognitive Science. *Research directions in cognitive science : European perspectives*, 1989.

[Wun96] R. Wunderling. Paralleler und objektorientierter simplex-algorithmus. *ZIB, Berlin, Germany, Tech. Rep. TR*, pages 96–09, 1996.

[Wun97] R. Wunderling. Soplex : The sequential object-oriented simplex class library, 1997.

Résumé

La modélisation est un domaine qui est devenu primordial dans l'informatique graphique de nos jours. On peut modéliser des objets par des nuages de points, des maillages ou des surfaces de type surfaces paramétrées. Nous présentons la chaine complète de reconstruction allant du maillage à un ensemble de surfaces paramétrées. Ce processus fait intervenir plusieurs techniques comme la segmentation de maillages, la paramétrisation et le fitting. Nous présentons une technique de quadrangulation, mais avec un cadre de travail adapté. Les M-tuiles et les M-pavages permettent d'englober toutes les informations nécessaires lors de la segmentation, lorsqu'on applique des techniques de pavage/découpage sur le maillage. L'algorithme de fitting présenté est nouveau, prenant en compte une norme uniforme en lieu et place de la norme euclidienne dominante, permettant ainsi l'utilisation d'un programme linéaire pour le résoudre. Une étude au niveau des critères de recollement de surfaces est faite, incluant les différents cas possibles de régularité choisie.

Mots-clefs : Modélisation, Reconstruction de surfaces, Programmation linéaire, Quadrangulation, M-pavage M-tuiles, Recollement de surfaces, Fitting.

Abstract

Computer graphics is a well studied domain nowadays and digital geometry processing is one of its sub domain we are talking about. We can modeling objects by cloud of points, meshes, ou any particular surfaces such as parametric surfaces. Our work is to show the entire process of reconstruction from the mesh to a set of parametric surfaces. This process involves many techniques such as mesh segmentation, parameterization, or fitting. We show a quad-meshing technique, based on a new well defined framework. M-tiles and M-tiling can produce any information needed by the segmentation, when we apply any cutting/ tiling techniques on the mesh. The fitting algorithm is new, using uniform norm instead of the euclidian norm, that let us using a linear program given to a solver. Stitching equations between all parametric surfaces are included in the program, and studied with all kind of regularizing parameters.

Keywords : Geometry processing, Surface reconstruction, Linear programming, Quadrangulation, M-tiling M-tiles, Surfaces stitching, Fitting.