HAL open science

Géométrie Numérique

Bruno Lévy

To cite this version:

Bruno Lévy. Géométrie Numérique. Modélisation et simulation. Institut National Polytechnique de Lorraine - INPL, 2008. tel-00337998

HAL Id: tel-00337998

https://theses.hal.science/tel-00337998v1

Submitted on 10 Nov 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Géométrie Numérique

Mémoire d'Habilitation à Diriger des Recherches

Bruno Lévy

Décembre 2007

Remerciements

Je dédie ce mémoire à ma femme Nathalie, et à mes deux enfants Alicia et Nathan, pour leur soutien, leur patience et leur tolérance.

Je remercie Jean-Laurent Mallet, qui a été mon directeur de thèse, et qui m'a initié à ces techniques passionnantes qui combinent méthodes numériques et maillages. Je remercie également Jean-Claude Paul, qui m'a accueilli dans son projet Inria ISA, où j'ai été inité aux méthodes de type éléments finis. J'aimerais également remercier André Journel, Gene Golub et Khalid Aziz de l'université de Stanford. J'ai eu la chance de pouvoir visiter leurs équipes pendant mon post-doc et d'y apprendre des notions de calcul numérique qui m'ont été très utiles tout au long de ce travail. Je remercie Karl Tombre pour son soutien et pour avoir accepté de me parrainer pour l'habilitation à diriger des recherches.

Je remercie les membres du jury, Leif Kobbelt, Claude Puech et Greg Turk, qui ont accepté d'assister à ma soutenance, malgrès des emplois du temps de plus en plus chargés.

Après avoir remercié mes professeurs et responsables, je tiens à remercier chaleureusement mes étudiants: Nicolas Ray, Wan-Chiu Li, Gregory Lecot, Laurent Castanié, Bruno Vallet, Luc Buatois, Rodrigo Toledo qui m'ont fait confiance au tout début du projet Inria ALICE, et ont directement contribué à sa création. Ils ont également contribué en grande partie au travail présenté dans ce mémoire.

Je remercie également Alla Sheffer (university of British Columbia), Pierre Alliez (Inria Sophia), David Cohen-Steiner (Inria Sophia), Jérôme Maillot (Alias-Wavefront), Sylvain Petitjean (Inria Nancy), Jean-Michel Dischler (LSIIT Strasbourg) et Khaleb Zayer (MPII Saarebruck) avec qui nous avons pu réaliser de très beaux projets en coopération, dont le contenu scientifique est également présenté dans ce mémoire.

Finalement, j'aimerais conclure ces remerciements par un petit clin d'oeil à Gilles Kahn, qui nous a quitté beaucoup trop tôt et que j'aurais aimé mieux connaitre. La dernière phrase que je me souviens l'avoir entendu prononcer lors de l'une de nos conversations téléphoniques m'a permis de trouver un sens nouveau à ce travail: *le continu est une bonne approximation du discret*.

Table des matières

1 Introduction **5**
1.1 Problématique . . . 5
1.1.1 Contexte général . . . 5
1.1.2 Géométrie numérique . . . 6

2 Fondements Scientifiques **9**
2.1 Introduction . . . 9
2.2 Méthodes numériques . . . 9
2.2.1 Structures de données pour les matrices creuses . . . 9
2.2.2 Résolution de systèmes linéaires . . . 15
2.2.3 Fonctions de plusieurs variables . . . 18
2.2.4 Optimisation quadratique . . . 21
2.2.5 Optimisation non-linéaire . . . 22
2.2.6 Optimisation contrainte - méthode de Lagrange . . . 24
2.3 Analyse fonctionelle . . . 24
2.3.1 Espaces de Hilbert . . . 25
2.3.2 Opérateurs . . . 25
2.3.3 Modélisation par élements Finis . . . 26
2.4 Traitement numérique de la géométrie . . . 26

3 Paramétrisation **29**
3.1 Notion de paramétrisation . . . 29
3.1.1 La carte du monde et les coordonnées sphériques . . . 29
3.1.2 Analyse des déformations et anisotropie . . . 32
3.2 Surfaces triangulées . . . 36
3.2.1 Définition . . . 36
3.2.2 Paramétrisation d'une surface triangulée . . . 36
3.2.3 Gradient dans un triangle . . . 39
3.2.4 Cartes barycentriques . . . 40
3.2.5 Laplacien discret . . . 41
3.3 Libérer le bord . . . 45
3.3.1 Méthodes fondées sur l'analyse des déformations . . . 47
3.3.2 Plaquage de textures sous contraintes . . . 50
3.3.3 Cartes conformes au sens des moindres carrés (LSCM) . . . 54
3.3.4 Cartes conformes et fonctions harmoniques . . . 57
3.3.5 Cartes conformes analytiques et géométriques . . . 59
3.3.6 Méthodes géométriques (ABF et ABF++) . . . 61
3.4 Re-maillage . . . 68
3.5 Extrapolation de surfaces . . . 69
3.6 Simulation numérique de la lumière . . . 70

4 Paramétrisation Globale **71**
4.1 Variétés Différentielle . . . 73
4.1.1 Notion de Variété Différentielle . . . 73
4.1.2 Calcul extérieur . . . 74
4.1.3 Homologie et co-homologie . . . 75
4.2 Paramétrisation Globale Périodique . . . 81
4.2.1 Variétés différentielles triangulées . . . 81
4.2.2 Alignement avec les vecteurs de contrôle . . . 83
4.2.3 Invariance en translation . . . 83
4.2.4 Invariance en rotation . . . 85
4.2.5 Résolution numérique . . . 86
4.2.6 Reconstruction de la paramétrisation θ, ϕ . . . 87
4.2.7 Structure globale des cartes . . . 89
4.2.8 Re-paramétrisation des cartes comportant des singularités . . . 90
4.3 Re-maillage implicite . . . 91
4.4 Conversion de maillages en surfaces paramétriques . . . 94
4.4.1 Les surfaces T-Splines . . . 94
4.4.2 Sommets extraordinaires et T-NURCC . . . 96
4.4.3 Ajustement aux données . . . 96
4.4.4 Résultats . . . 102
4.5 Méthodes spectrales . . . 104
4.5.1 Le mode discret: Laplaciens de graphes . . . 106
4.5.2 Le mode continu . . . 111
4.5.3 Harmoniques Variétés . . . 115
4.5.4 Transformée en Harmoniques Variétés . . . 117
4.5.5 Calcul de la THV . . . 117

5 Perspectives **123**
5.1 Importance de l'abstraction . . . 124
5.2 Bases de fonctions mobiles . . . 125
5.3 Bilan et perspectives . . . 128
5.3.1 Bilan . . . 128
5.3.2 Perspectives . . . 129

Chapitre 1

Introduction

1.1 Problématique

Ce mémoire traite de plusieurs classes de méthodes de modélisation 3D à l'aide de maillages. Avant de décrire le type d'approches que j'ai mise en place, je vais tout d'abord tenter de motiver ces sujets de recherches, en en présentant les enjeux dans un contexte général.

1.1.1 Contexte général

Les progrès réalisés récemment dans la technologie des capteurs, la puissance croissante des ordinateurs, ainsi que l'émergence de nouveaux secteurs d'applications conduisent à la définition de nouveaux besoins: il devient ainsi nécessaire de parvenir à modéliser, identifier, contrôler et optimiser des systèmes complexes pour lesquels les techniques de calcul scientifique existantes ne sont plus suffisantes. Ces systèmes exhibent une complexité à plusieurs niveaux: ils peuvent mettre en jeux des *masses de données* conséquences, et ce à des *échelles mutiples*, couplant différents phénomènes physiques. D'autre part, ces systèmes vont mettre en jeu un grand nombre de composants hétérogènes, interdépendants, et dont le fonctionnement revêt à la fois des aspects continus et discrets. Ceci pose de difficiles problèmes, en termes de complexité algorithmique, et de difficultés numériques, concernant notamment des aspects non-linéaires.

Comprendre, modéliser et simuler de tels systèmes complexes joue un rôle central dans plusieurs secteurs de l'industrie, tels que la CAO/CFAO, l'énergie et la santé (c.f. Figure 1.1):

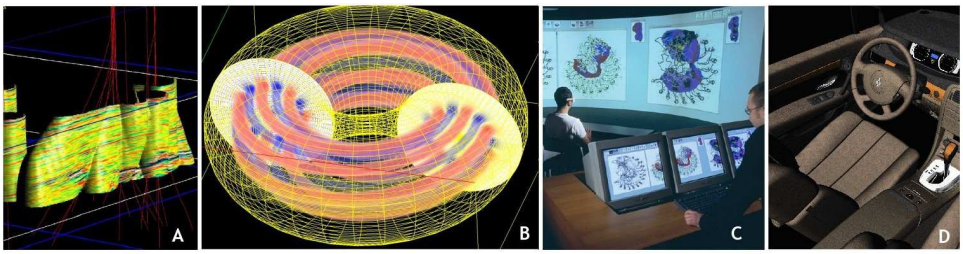


Figure 1.1: *Applications de nos méthodes à différents domaines, développées en coopération avec nos partenaires académiques et industriels. A: exploration pétrolière; B: physique des plasmas; C: biologie moléculaire; D: design industriel.*

CAO/CFAO : L'entreprise d'avionique Dassault a développé son nouvel avion "Falcon F7X" en utilisant une plate-forme de maquette numérique commune, utilisée tout au long du cycle de vie de l'avion, depuis la conception jusqu'à la maintenance, ce qui inclut également les processus de simulation numérique. L'AIRBUS A380 a été développé en utilisant une technologie similaire (c.f. http://www.aviation-civile.gouv.fr/html/publicat/av_civil/319/conception319.pdf)[1].

Energie : L'industrie pétrolière utilise des infrastructures complexes, à très grande échelle (multi-nationale). Modéliser le comportement de ces infrastructures nécessite des modèles multi-échelle, qui couplent des phénomènes physiques multiples. Ceci requiert d'importantes ressources de calcul. Ainsi, la compagnie Statoil (pays bas) possède le 89ème plus gros calculateur, et TotalFinaElf le 187ème (c.f. http://www.top500.org)[2].

Santé : Les avancées en techniques d'acquisition et d'imagerie médicale (imagerie ultrasonique, scanners et résonance magnétique nucléaire) peuvent à présent extraire des images 3D haute-résolution d'une manière non-intrusive. Une combinaison judicieuse de logiciel et de matériel permet de reconstruire l'information à partir de ces données et de la présenter au praticien. La mise au point de médicaments est également un autre secteur – moins connu – très consommateur de modèles informatique, prenant en compte notamment les interactions géométriques et physiques entre molécules complexes.

Ces différents domaines d'applications ont pour point commun de tous mettre en jeu une modélisation très fine, servant soit à étudier l'existant, soit à concevoir de nouveaux dispositifs, prédire leur comportement, et ainsi optimiser leur conception. Pour servir de support à cette modélisation, les représentations à base de maillages sont devenues omniprésentes, grâce en particulier à la grande souplesse qu'elles offrent au processus de modélisation. Pour cette raison, nous nous sommes intéressés à ces supports géométriques, et plus particulièrement à ce que nous avons appellé la *géométrie numérique*, évoquée dans la section suivante.

1.1.2 Géométrie numérique

Mon programme de recherche a pour objectif de développer la *géométrie numérique*, à savoir de nouveaux formalismes pour la représentation et la manipulation d'ensembles de points de $\mathbb{R}^n$ (surfaces, volumes, ...). Je m'intéresse particulièrement aux objets discrétisés sous forme de complexes cellulaires, tels que les surfaces triangulées et polygonales, les grilles 3D structurées et non-structurées. Pour définir de nouveaux opérateurs de modélisation agissant sur ces objets, mon approche consiste à développer les points suivants:

- ◇ Formaliser l'opérateur de modélisation, le plus souvent sous forme d'une équation aux dérivées partielles (EDP). A ce stade, certaines notions de physique peuvent guider l'intuition. Par exemple, une approximation de l'énergie de flexion d'une plaque mince peut être utilisée pour le lissage géométrique et l'extrapolation de surfaces.
- ◇ Discrétiser cet opérateur sous une forme compatible avec la représentation de l'objet, en utilisant par exemple des méthodes de type différences finies ou éléments finis.
- ◇ Montrer les "bonnes propriétés" de cette forme discrète de l'opérateur, telles que l'existence et l'unicité de la solution, ou encore l'indépendance de la solution par rapport à la discrétisation du maillage.
- ◇ Minimiser le critère ainsi défini, en utilisant des outils d'optimisation numérique (gradient conjugué, Gauss-Newton, ...). Le problème de minimisation prend place dans un

[1] qui n'est toutefois pas parvenue à prédire les retards de livraisons qui ont récemment côuté cher à l'entreprise.
[2] Les premières places du top500 sont utilisées par des calculateurs militaires, pour la simulation d'explosions nucléaires.

espace de très grande dimension (de l'ordre du nombre de sommets de l'objet, pouvant atteindre plusieurs millions), il convient donc de développer des outils permettant de calculer la solution en un temps raisonable, tels que des pré-conditioneurs spécialisés, ou encore des méthodes multi-échelle calculant la solution sous une forme répartie sur plusieurs harmoniques.

Les problèmes que j'étudie actuellement comprennent la paramétrisation des ensembles simpliciaux, l'extrapolation des surfaces triangulées, le lissage discret, la combinatoire des complexes cellulaires, l'ajustement aux données, la modélisation et la visualisation volumique.

Le formalisme que j'utilise combine des aspects de la géométrie différentielle, de l'analyse complexe, de l'algèbre lineaire, de l'analyse numérique, de la topologie discrète et de la théorie des graphes.

L'originalité de cette approche réside dans la transformation du problème de modélisation en une structure algébrique reflétant les propriétés géométriques du problème. Par exemple, les nombres complexes correspondent à la fois à des points et à des similitudes dans $\mathbb{R}^2$, ce que j'ai utilisé pour construire un équivalent discret de la notion de paramétrisation conforme et pour en démontrer les propriétés (c.f. section 3.3.3). Transformer le problème géométrique en un problème d'optimisation numérique permet ainsi de remplacer les relaxations locales communément utilisées par des approches d'optimisation globale, bien plus efficaces. Ceci permet de mettre au point des algorithmes appliquables à des jeux de données de taille industrielle.

Après une introduction des fondements scientifiques (Chapitre 2), je décris dans ce mémoire mes contributions principales, structurées par cette approche particulière, dans le domaine de la paramétrisation de surfaces, tout d'abord dans le cas d'objets homéomorphes à des disques (Chapitre 3), puis dans le cas d'objets de genre arbitraire (Chapitre 4). Enfin, en guise de conclusion, mon programme de recherche donne des directions pour aller plus loin, en se laissant guider en particulier par la notion d'abstraction (Chapitre 5).

Chapitre 2

Fondements Scientifiques

2.1 Introduction

Ce chapitre introduit les notions fondamentales et outils de bases que j'utilise dans mon travail de recherche. Pour décrire ces outils, j'ai choisi de suivre une approche du bas vers le haut. Ainsi, nous verrons tout d'abord dans la Section 2.2 les algorithmes d'optimisation numérique servant de "briques élémentaires" sur lesquelles s'appuient tout le reste. Nous verrons ensuite les quelques notions d'analyse fonctionelles à la base des méthodes de type éléments finis (Section 2.3). Finalement, nous introduisons rapidement le domaine scientifique du traitement numérique de la géométrie (Section 2.4), en insistant sur les problèmes ouverts difficiles. En se fondant sur ces bases, les chapitres suivants décrirons mes contributions.

2.2 Méthodes numériques

Ce chapitre se fonde sur la section *How to make it work in practice* de mes notes de cours pour le cours *Mesh Parameterization, Theory and Practice*, que j'ai co-organisé à la conférence Siggraph en 2007, avec Alla Sheffer et Kai Hormann [HLS+07].

Les méthodes numériques sont un formalisme particulièrement efficace pour traiter des problèmes d'optimisation géométriques faisant intervenir de grands maillages. L'approche consiste à modéliser le problème sous la forme d'une fonction de plusieurs variables à optimiser, les variables correspondant à des valeurs (par exemple les coordonnées) attachées aux sommets du maillage. Ces problèmes d'optimisation numériques mettent en jeu de grandes matrices, définies à partir des maillages et des valeurs stockées sur les sommets et/ou arêtes et/ou facettes de ces maillages. Dans la plupart des cas, ces matrices sont creuses (et ont plus précisément une structure identique ou dérivée de la matrice d'adjacence du graphe formé par le maillage). Nous débuterons donc l'étude de ces méthodes numériques par quelques aspects liés au structures de données efficaces pour stocker des matrices creuses.

2.2.1 Structures de données pour les matrices creuses

Stockage TRIAD

Pour optimiser le stockage des matrices creuses, la première idée qui peut venir à l'esprit consiste à ne stocker que les entrées non-nulles de la matrice ainsi que les indices (i, j) associés. Ainsi, on ne stocke que NNZ coefficients (NNZ: Number of Non-Zero à savoir

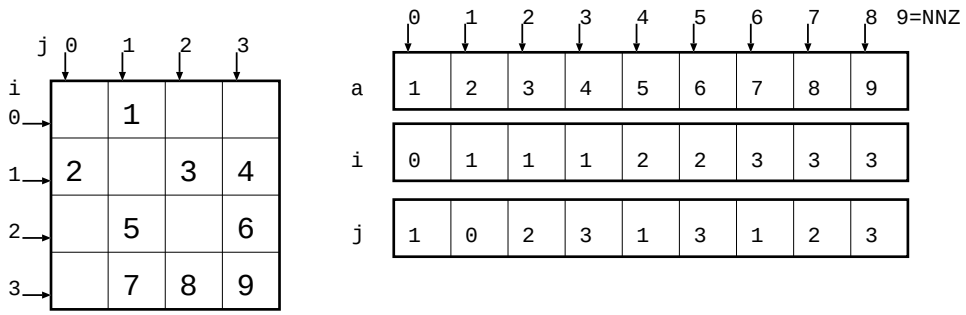


Figure 2.1: *Exemple de matrice creuse stockée en format TRIAD*

Algorithme 1 Structure de donnée TRIAD

```
struct TRIADMatrix {
        int N ; // dimension de la matrice
        int NNZ ; // nombre de coefficients non nuls
        double* a ; // coefficients non nuls (tableau de taille NNZ)
        int* I ; // indices de lignes (tableau de taille NNZ)
        int* J ; // indices de colonnes (tableau de taille NNZ)
} ;
```

Algorithme 2 Produit matrice TRIAD $\times$ vecteur

```
void mult(double* y, TRIADMatrix* M, double* x) {
   for(int i=0; i<M->N; i++) {
      y[i] = 0.0 ;
   }
   for(int k=0; k<M->NNZ; k++) {
      y[M->I[k]] += M->a[k] * x[M->J[k]] ;
   }
}
```

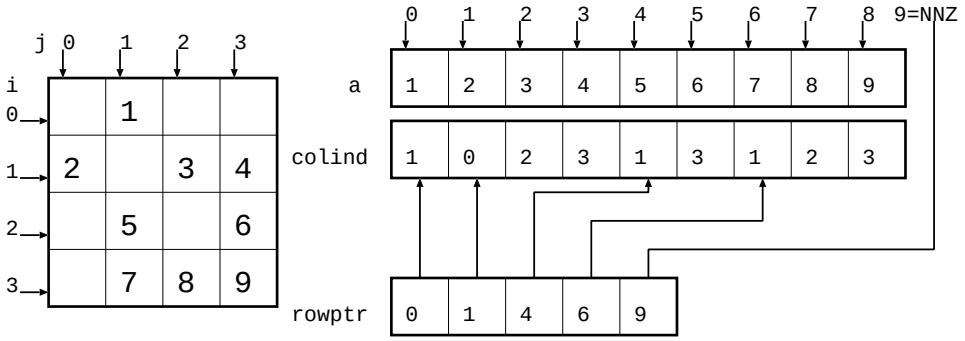


Figure 2.2: *Exemple de matrice creuse stockée en format CRS*

nombre d'entrées non-nulles) ainsi que les indices associés. Ce mode de stockage de matrices creuses, intitulé *format TRIAD*, est illustré sur la Figure 2.1, et une implantation en langage C est donnée à titre d'exemple (Algorithme 1). Nous donnons également la fonction permettant de calculer le produit entre une matrice stockée au format TRIAD et un vecteur (Algorithme 2). Le calcul de produits matrice-vecteur est par exemple particulièrement utile pour implanter un solveur fondé sur la méthode du gradient conjugué (voir plus loin, Section 2.2.2).

Bien que le mode de stockage TRIAD permette de ne pas utiliser inutilement de l'espace de stockage pour les entrées non-nulles, il occasione une forte redondance des données dans le stockage des coefficients i qui encodent le numéro de la ligne associé à chaque entrée. Pour cette raison, ce format reste peu utilisé en pratique dans les grandes libraries de résolution numérique (il se limite à la définition de formats de données, car sa grande simplicité en favorise la compréhension et l'utilisation). En pratique, les développeurs de grandes libraries de résolution numérique préfèrent utiliser le format CRS (Compressed Row Storage, pour stockage de lignes compressées), bien plus compact en mémoire. Nous détaillons ce dernier format dans la sous-section suivante.

Algorithme 3 Structure de donnée CRS (Compressed Row Storage)

```
struct CRSMatrix {
        int N ; // dimension de la matrice
        int NNZ ; // nombre de coefficients non nuls
        double* a ; // coefficients non nuls (tableau de taille NNZ)
        int* colind ; // indices de colonnes (tableau de taille NNZ)
        int* rowptr ; // pointeurs de lignes (tableau de taille N+1)
        double* diag ; // elements diagonaux (tableau de taille N)
} ;
```

Stockage par lignes compressées

Le format CRS (ou Compressed Row Storage pour stockage par lignes compressées) est la représentation la plus commune pour les matrices creuses, utilisée par les grandes librairies de calcul scientifique. La variante transposée CCS (ou Compressed Column Storage pour stockage par colonnes compresée) se rencontre également (suivant les types d'algorithmes et choix d'implantation des librairies). Nous présentons ici le format CRS. L'algorithme 3 décrit l'implantation (en langage C), et la Figure 2.2 en montre un exemple. La structure de données CRS utilise trois tableaux pour représenter les coefficients non-nuls et leurs indices. Le tableau `a` stocke l'ensemble des entrées non-nulles de la matrice, le tableau `colind`s indique pour chaque entrée l'indice de colonne qui lui correspond. Les lignes sont encodées d'une manière différente, par le tableau `rowptr`, qui indique pour chaque

ligne son début et sa fin dans les tableaux `a` et `colind`. Afin de faciliter l'écriture des algorithmes (en évitant un test), il est d'usage de compléter le tableau `colind` par une entrée supplémentaire, pointant une case plus loin que la dernière entrée de la matrice. Cette entrée est appelée communément une *sentinelle*. Nous verrons plus loin comment celle-ci facilite l'écriture de l'algorithme calculant le produit entre la matrice creuse et un vecteur donné.

Algorithme 4 Produit matrice CRS $\times$ vecteur

```
void mult(double* y, CRSMatrix* M, double* x) {
   for(int i=0; i<M->N; i++) {
      y[i] = 0.0 ;
      for(int jj=M->rowptr[i]; jj<M->rowptr[i+1]; jj++) {
         y[i] += M->a[jj] * x[M->colind[jj]] ;
      }
   }
}
```

L'Algorithme 4 montre comment implanter le calcul du produit entre une matrice creuse `M` stockée en format CRS et un vecteur donné `x`. Comme nous pouvons le voir, l'utilisation de la sentinelle `rowptr[N] = NNZ` permet de ne pas avoir de cas particulier pour traiter la dernière ligne de la matrice.

Différentes variantes de la structure CRS existent. Ainsi, si la matrice est symétrique, il est possible d'économiser de l'espace de stockage en ne représentant que la moitié triangulaire inférieure de la matrice. Suivant les besoins, d'autres variantes stockent la diagonale dans un vecteur séparé, par exemple pour faciliter le calcul d'un préconditioneur de Jacobi (voir plus loin 2.2.2). Des variantes stockent les colonnes (CCS pour Compressed Column Storage), ce qui peut faciliter certains types de calculs. Finalement, d'autres variantes stockent des blocs de coefficients (format BCRS pour Bloc Compressed Row Storage), ce qui permet à la fois de diminuer la quantité mémoire utilisée, d'accélérer les accès à la mémoire et de favoriser l'utilisation des instructions vectorielles (jeux d'instructions SSE sur les processeurs Intel, ou ARBfp sur les cartes graphiques). Nous avons expérimenté ce dernier format pour implanter notre solveur CNC (*Concurrent Number Cruncher*, pour "mangeur" de nombres parallère) [BCL07].

Toutefois, dans notre cas particulier, impliquant des problèmes numériques issus de la modélisation 3D à l'aide de maillages, la structure CRS et ses variantes peut s'avérer trop rigide. En effet, lors d'un problème de modélisation géométrique, il est naturel de construire la matrice du ou des systèmes linéaires mis en jeu par un parcours du graphe correspondant au maillage. Lors du parcours de ce graphe, des coefficients seront accumulés dans la matrice, générés par des voisinages élémentaires (appelés *stencil*). Comme le nombre de coefficients non-nuls dans une matrice CRS est fixé à l'avance, dans notre cas, construire la représentation de la matrice va nécessiter deux parcours du graphe: le premier parcours va identifier la structure de la matrice, à savoir, l'ensemble des couples (i, j) tels que $a_{i,j}$ est non-nul, suite à quoi la structure de données est allouée en mémoire. Ensuite, un deuxième parcours calcule les valeurs numériques des coefficients $a_{i,j}$. En pratique, cette manière de procéder est particulièrement pénible à implanter. Pour cette raison, j'ai implanté une structure de matrice dynamique, dont le nombre de coefficients non-nuls peut varier au cours de l'execution.

Matrices dynamiques

La structure de matrice dynamique que j'ai développée, à la base des expérimentations liés à mes travaux de recherche en géométrie numérique, se fonde sur l'utilisation de la classe `std::vector` de la librairie C++ standard (STL pour Standard Template Library). Cette

Algorithme 5 Structure de matrice creuse dynamique

```
class SparseMatrix {
public:
    struct Coeff {
        Coeff() { }
        Coeff(int j, double val) : index(j), a(val) { }
        int index ;
        double a ;
    } ;

    class Row : public std::vector<Coeff> {
    public:
        void add(int index, double val) {
            for(int i=0; i<size(); i++) {
                if((*this)[i].index == index) {
                    (*this)[i].a += val ;
                    return ;
                }
            }
            std::vector<Coeff>::push_back(Coeff(index, val)) ;
        }
    } ;

    SparseMatrix(int dim) : dimension(dim) {
        row = new Row[dim] ;
    }

    ~SparseMatrix() { delete[] row;  }

    // aij <- aij + val
    void add(int i, int j, double val) {
        row[i].add(j, val) ;
    }

    // A <- 0
    void clear() {
        for(int i=0; i<dimension; i++) {
            row[i].clear() ;
        }
    }

   // number of non-zero coefficients
   int nnz() const {
      int result = 0 ;
      for(int i=0; i<dimension; i++) {
          result += row[i].size() ;
      }
      return result ;
   }
    int dimension ;
    Row* row ;
} ;
```

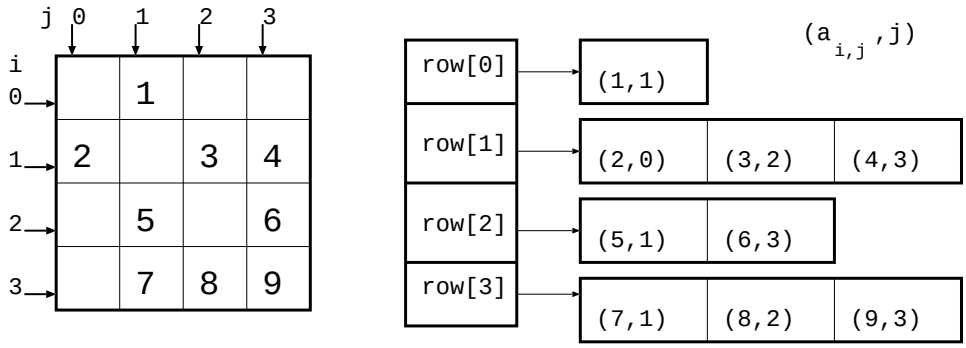


Figure 2.3: *Structure de matrice creuse dynamique*

Algorithme 6 Produit matrice dynamique × vecteur

```
    // y <- Mx
    void mul(double* y, const SparseMatrix& M, const double* x) {
        for(int i=0; i<M.dimension; i++) {
           y[i] = 0 ;
            const Row& R = M.row[i] ;
            for(int jj=0; jj<R.size(); jj++) {
                y[i] += R[jj].a * x[ R[jj].index ]  ;
            }
        }
    }
```

Algorithme 7 Conversion d'une matrice dynamique vers le format CRS

```
void convert_to_CRS(
   const SparseMatrix& M,
   CRSMatrix& M_CRS,
   int array_base // 0 pour le C, 1 pour le Fortran
) {
   M_CRS.N = M.dimension ;
   M_CRS.NNZ = M.nnz() ;
   M_CRS.A = new double[M_CRS.NNZ] ;
   M_CRS.col_ind = new int[M_CRS.NNZ] ;
   M_CRS.row_ptr = new int[M_CRS.N+1] ;
   int count = 0 ;
   for(int i=0; i<M_CRS.N; i++) {
        const SparseMatrix::Row& R = M.row[i] ;
        M_CRS.row_ptr[i] = count + array_base ;
        for(int jj=0; jj<R.size(); jj++) {
            M_CRS.a[count] = R[jj].a ;
            M_CRS.col_ind[count] = R[jj].index + array_base ;
            count++ ;
        }
    }
    M_CRS.col_ptr[M_CRS.N] = M_CRS.NNZ + array_base ;
}
```

classe conteneur a la propriété intéressante de pouvoir croitre dynamiquement tout en minimisant le nombre de recopies mémoire[1]. Ainsi, notre structure de matrice creuse, symbolisée sur la Figure 2.3, et détaillée dans l'Algorithme 5, est constituée d'un tableau `row` de lignes. Chaque ligne est implantée par un `std::vector` de paires valeur-indice,une telle paire étant représentée par la structure `Coeff`. Cette structure consomme un espace de stockage du même ordre de grandeur que celui d'une matrice CRS, tout en offrant une plus grande souplesse. Cette souplesse se traduit par la fonction `add(i,j,val)`, qui permet à tout moment d'ajoute la valeur `val` au coefficient $a_{i,j}$.

Le produit matrice-vecteur reste relativement simple à implanter avec cette structure. Le code source correspondant est donné dans l'Algorithme 6. Toutefois, il convient de noter que la plus grande souplesse offerte par cette structure de données dynamique se paye par une moins grande efficacité des calculs de produits matrice-vecteur (environ d'un facteur 2 d'après nos expérimentations). Ceci s'explique sans doute par la moins bonne localité des données, ne permettant pas une aussi bonne utilisation du cache que dans le cas du format CRS. Pour cette raison, dans les cas où un grand nombre de produits matrice-vecteur sont susceptibles d'être calculés (e.g. dans un solveur itératif de type gradient conjugué, décrit plus loin), il peut être plus efficace de convertir la matrice dynamique dans le format CRS. Le code source correspondant à cette conversion est donné par l'Algorithme 7. Cette fonction ne présente pas de difficulté particulière, si ce n'est le paramètre `array_base` qui mérite quelques explications supplémentaires: les langages C et FORTRAN n'utilisent pas la même convention pour l'indiçage des tableaux, dont le premier indice est 0 en C, et 1 en FORTRAN. Pour cette raison, si la matrice dynamique (structure de données C++) est utilisée ensuite par une routine FORTRAN, il est nécessaire de positionner ce paramètre à 1 pour respecter la convention d'indiçage de ce langage.

Maintenant que nous avons pu voir les structures de données élémentaires pour les matrices creuses, les algorithmes permettant de les manipuler et leur implantation en C et C++, nous allons voir quelques algorithmes de résolution de systèmes linéaires.

2.2.2 Résolution de systèmes linéaires

Un grand nombre de problèmes en modélisation géométrique nécessitent de résoudre un grand système linéaire, à savoir une équation de la forme $Ax = b$, avec A une matrice carrée $n \times n$ inversible, b un vecteur de dimension n, et x le vecteur inconnu (de dimension n également). Dans cette section, nous décrivons plusieurs méthodes permettant de résoudre ces systèmes linéaires.

Relaxation

La méthode de relaxation est la plus simple, à la fois du point de vue conceptuel, et du point de vue de l'implantation. Cette méthode a beaucoup été utilisée dans les années 90 pour implanter des algorithmes de traitement numérique de la géométrie. Elle a plus tard été remplacée par des méthodes plus sophistiquées, évoquées plus loin.

La méthode de relaxation peut se comprendre facilement en considérant le problème à résoudre $Ax = b$ comme un système linéaire:

$$\left\{\begin{array}{lllll} a_{1,1}x_1 & +a_{1,2}x_2 & +\dots & +a_{1,n}x_n & = & b_1 \\ & & \vdots & & & \\ a_{i,1}x_1 & +a_{i,2}x_2 & +\dots & +a_{i,n}x_n & = & b_i \\ & & \vdots & & & \\ a_{n,1}x_1 & +a_{n,2}x_2 & +\dots & +a_{n,n}x_n & = & b_n \end{array}\right.$$

[1] ceci est réalisé en doublant la taille de l'espace mémoire alloué au conteneur à chaque fois qu'une recopie est nécessaire, ce qui fait décroitre le nombre de recopies en fonction du log de la taille finale demandée.

La méthode consiste alors à parcourir les équations une par une, et à calculer pour l'équation i la valeur de x_i obtenue en faisant comme si tous les autres x_j pour $j \neq i$ étaient connus, ce qui donne le schéma de mise à jour suivant pour la valeur x_i :

$$x_i \leftarrow \frac{1}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right)$$

L'algorithme de résolution de système linéaire par la méthode de relaxation s'écrit alors:

tant que $\|Ax - b\| < \varepsilon$
  pour i **de** 1**à** n
    $x_i \leftarrow \frac{1}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right)$
  fin$//$*pour*
fin$//$*tant que*

avec ε la précision souhaitée par l'utilisateur. Il est également possible de spécifier un nombre d'itération maximum, afin d'arrêter l'algorithme lorsqu'il ne parvient pas à converger.

Comme nous pouvons l'observer, cette algorithme ne peut pas s'appliquer à des matrices ayant des valeurs nulles sur la diagonale. D'une manière plus générale, il est possible de démontrer une condition suffisante de convergence de l'algorithme: si la matrice est diagonale dominante, à savoir:

$$\forall i, |a_{i,i}| > \sum_{j \neq i} |a_{i,j}|$$

alors l'algorithme converge.

Il est possible d'accélérer la méthode de relaxation, en utilisant le fait que la mise à jour effectuée pour chaque variable x_i "allait dans la bonne direction". Intuitivement, en allant "un peu plus loin" dans cette direction, à savoir en multipliant le déplacement par un facteur ω, il sera possible d'accélérer la convergence. Le schéma de mise à jour modifié s'écrit alors:

$$\begin{array}{lcl} x_{prev} & \leftarrow & x_i \\ x_i & \leftarrow & \frac{1}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right) \\ x_i & \leftarrow & x_{prev} + \omega (x_i - x_{prev}) \end{array}$$

Ce schéma calcule tout d'abord la mise à jour de x_i comme précédement, puis augmente le déplacement effectué par rapport à l'ancienne valeur x_{prev} d'un facteur ω. Il est possible de démontrer que l'algorithme converge sous les mêmes conditions que pour la relaxation (matrice A diagonale dominante), et pour $\omega \in [1, 2[$. L'algorithme ainsi modifié est connu sous le nom de *sur-relaxations successives* (ou SOR pour *successive over-relaxation* dans la littérature anglophone). En ré-écrivant sous forme plus compacte le schéma de mise à jour, l'algorithme SOR s'écrit alors de la manière suivante:

tant que $\|Ax - b\| < \varepsilon$
  pour i **de** 1 **à** n
    $x_i \leftarrow (1 - \omega) x_i + \frac{\omega}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right)$
  fin$//$*pour*
fin$//$*tant que*

Le choix optimum du paramètre ω est déterminé par les valeurs propres de A. Comme calculer ces valeurs propres est en général plus difficile que de résoudre le système linéaire, le paramètre ω est en général déterminé soit par une étude théorique des valeurs propres (pour des problèmes particuliers), soit de manière empirique.

Le principal avantage de la méthode SOR est sa grande simplicité d'implantation. Cette simplicité a favorisé son utilisation dans la communauté "traitement numérique de

la géométrie"[Tau95a]. Toutefois, comme nous le verrons par la suite, des méthodes plus sophistiquées, telles que la méthode du gradient conjugué, permettent de résoudre plus efficacement ce type de problèmes.

Le gradient conjugué

L'algorithme du gradient conjugué est bien plus efficace, et à peine plus compliqué à implanter. Cet algorithme de résolution de systèmes symétriques se fonde sur l'équivalence entre la résolution du système $Ax = b$ et la minimisation de la forme quadratique $F(x) = 1/2x^tAx - b^tx$. De manière plus précise, l'algorithme calcule une base de vecteurs orthogonaux dans l'espace de la matrice (à savoir une base de vecteurs conjugés), en appliquant l'algorithme d'orthogonalisation de Schmidt. Les calculs se simplifient de manière remarquable, et permettent de calculer les vecteurs un par un, en ne gardant en mémoire que le dernier vecteur. Le suivant est alors obtenu sous forme d'une combinaison linéaire entre le précédent et le gradient de F au point courant. L'algorithme complet s'écrit de la manière suivante:

Algorithme 8 Gradient conjugué pré-conditionné

$i \leftarrow 0$; $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}$; $\mathbf{d} \leftarrow \mathbf{M}^{-1}\mathbf{r}$;
$\delta_{new} \leftarrow \mathbf{r}^T\mathbf{d}$; $\delta_0 \leftarrow \delta_{new}$;
tant que $i < i_{max}$ **et** $\delta_{new} > \varepsilon^2\delta_0$
    $\mathbf{q} \leftarrow \mathbf{A}\mathbf{d}$; $\alpha \leftarrow \frac{\delta_{new}}{\mathbf{d}^T\mathbf{q}}$;
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d}$; $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}$;
    $\mathbf{s} \leftarrow \mathbf{M}^{-1}\mathbf{r}$; $\delta_{old} \leftarrow \delta_{new}$;
    $\delta_{new} \leftarrow \mathbf{r}^T\mathbf{s}$; $\beta \leftarrow \frac{\delta_{new}}{\delta_{old}}$;
    $\mathbf{d} \leftarrow \mathbf{r} + \beta\mathbf{d}$; $i \leftarrow i + 1$;
fin

Dans cet algorithme, la matrice M, appelée un préconditioneur, permet d'améliorer la vitesse de convergence de l'algorithme. Le préconditioneur de Jacobi définit M comme la matrice composée des éléments diagonaux de A. Comme nous pouvons le remarquer, les seules opérations effectuées par cet algorithme, mise à part de simples opérations sur des vecteurs, sont des produits matrice-vecteur.

La méthode du gradient conjugué ne peut s'appliquer qu'à des matrices symétriques. Dans le cas où la matrice A n'est pas symétrique, il est possible de dériver du système $Ax = b$ un système symétrique équivalent, de la manière suivante:

$$\begin{pmatrix} Id & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

Il est alors possible d'appliquer la méthode du gradient conjugué à ce système. Ceci définit l'algorithme du *gradient bi-conjugué*, ou encore biCG. Une variante nommée biCGSTAB (gradient bi-conjugué stabilisé) permet d'améliorer la vitesse de convergence en stabilisant les calculs.

Le lecteur souhaitant en apprendre plus sur la théorie du gradient conjugué pourra se réferrer à l'excellent texte de Jonathan Shewchuck sur le sujet[She94].

[Tau95a] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH Conference Proceedings*, pages 351–358. ACM, 1995.

[She94] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizingpain. Technical report, CMU School of Computer Science, 1994. ftp://warp.cs.cmu.edu/quake-papers/painless-conjugate-gradient.ps.

méthode	avantages	inconvénients
relaxation - SOR	facile à implanter faible coût mémoire	pas très efficace
gradient conjugué	relativement facile à implanter faible coût mémoire	efficacité moyenne
méthodes directes	les plus efficaces codes publics disponibles	coût mémoire important algorithmes très complexes

Table 2.1: *Bilan sur les méthodes de résolution de systèmes linéaires*

Les méthodes directes creuses

Une méthode possible pour résoudre un système linéaire consiste à factoriser la matrice, sous forme d'un produit de matrices simples à inverser. Par exemple, la factorisation LU consiste à trouver la matrice triangulaire inférieure L et la matrice triangulaire supérieure U dont le produit est égal à la matrice M du système (si la matrice est symétrique, U correspond à la transposée de L). Une fois cette factorisation obtenue, il devient très facile de résoudre des systèmes linéaires impliquant $M = LU$, de la manière suivante:

$$LUx = b \quad \rightarrow \left\{ \begin{array}{lcl} Lx_1 & = & b \\ Ux & = & x_1 \end{array} \right.$$

Ainsi, l'algorithme résout le système linéaire en résolvant deux systèmes triangulaires (par substitutions successives). D'une manière conceptuelle, la méthode du pivot de Gauss, enseignée à l'école pour résoudre des systèmes linéaires à la main, correspond à cet algorithme (exprimé d'une manière légèrement différente).

Dans le cas de matrices creuses, différentes méthodes permettent de calculer des facteurs L et U représentés également par des matrices creuses. Différentes librairies, disponibles sur Internet en OpenSource, implante ces algorithmes complexes, à savoir SuperLU, TAUCS, MUMPS. Comme l'ont remarqué Botsch *et. al*[BBK05], ces solveurs directs sont particulièrement efficaces pour des problèmes de géométrie numérique.

Bilan sur les méthodes de résolution de systèmes linéaires

Nous concluons cette sous-section par un bilan rapide de ces méthodes de résolution de systèmes linéaires, résumé dans la Table 2.1. En résumé, les méthodes de type SOR ont l'avantage d'être extrêmement faciles à implanter, mais ont du mal à converger pour des maillages comportant plus de dix mille sommets. Les méthodes de type gradient conjugué réalisent un bon compromis entre efficacité et difficulté d'implantation. Les méthodes directes creuses sont les plus rapides, mais ont parfois l'inconvénient de consommer une très grande quantité de mémoire. Enfin, les méthodes directes creuses en mémoire externe apparues récemment[MIT06] permettent de bénéficier de l'efficacité des méthodes directes, sans présenter le risque de dépasser les ressources RAM disponible.

2.2.3 Fonctions de plusieurs variables

Nous nous intéressons à présent à des fonctions $F : x \mapsto F(x)$ de $\mathbb{R}^n$ dans $\mathbb{R}$. Ces fonctions permettent par exemple de formaliser des critères géométriques devant être satisfait par des surfaces. Dans ce cadre, la fonction prend en argument un vecteur x qui regroupe toutes les coordonnées des sommets du maillage, et renvoie un réel qui mesure la "qualité" de

[BBK05] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for mesh processing. In *IMA Mathematics of Surfaces XI, Lecture Notes in Computer Science*, 2005.

[MIT06] Omer Meshar, Dror Irony, and Sivan Toledo. An out-of-core sparse symmetric indefinite factorization method. *ACM Transactions on Mathematical Software*, 32:445–471, 2006.

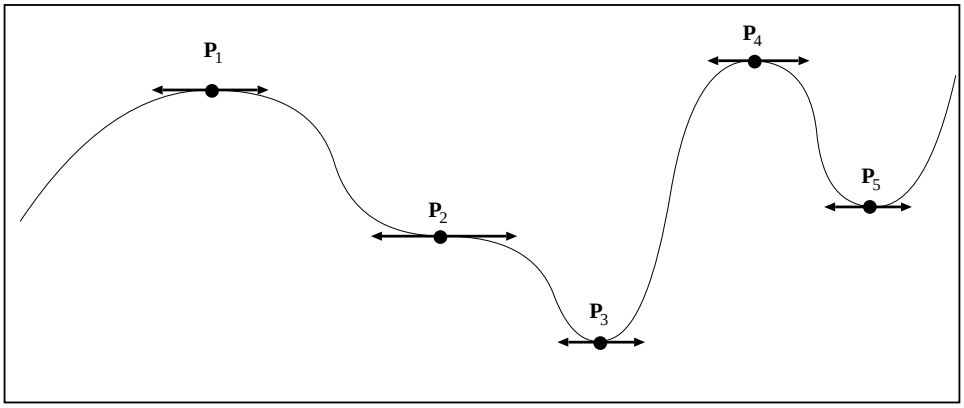


Figure 2.4: *Points stationaires. Les points* $\mathbf{P}_1$ *et* $\mathbf{P}_4$ *sont des maxima locaux,* $\mathbf{P}_2$ *est un point d'inflexion,* $\mathbf{P}_5$ *un minimum local et* $\mathbf{P}_3$ *le minimum global.*

la géométrie ainsi décrite (les petites valeurs de cette fonction correspondant en général à la meilleure qualité). L'algorithme d'optimisation géométrique va alors rechercher les paramètres de cette fonction (à savoir les coordonnées des sommets) qui minimisent la fonction. Dans ce cadre, une telle fonction est appelée "énergie", ou encore "fonction objectif". Les méthodes *variationelles* permettent de minimiser ces énergies en étudiant les variations (à savoir les dérivées) de $F(x)$ en fonction des paramètres $x = [x_1, \ldots x_n]$. Nous étudierons des méthodes à l'ordre 2, qui nécessitent de définir la notion correspondant à la dérivée première (à savoir le *gradient*) et celle correspondant à la dérivée seconde (à savoir le *Hessien*). Nous étudierons ensuite quelques petits théorèmes utiles permettant de faciliter les calculs faisant intervenir les dérivées de fonctions de plusieurs variables.

Le gradient

L'information différentielle à l'ordre 1 est représentée par le vecteur de toutes les dérivées par rapport à toutes les variables, appellé gradient de F, et noté ∇F:

$$\nabla F = \begin{bmatrix} \partial F/\partial x_1 \\ \vdots \\ \partial F/\partial x_i \\ \vdots \\ \partial F/\partial x_n \end{bmatrix}$$

La notion de gradient est très utile dans les calculs de minima, caractérisés par le fait que le gradient s'annule en un minimum de la fonction (si celle-ci est dérivable). Toutefois, l'ensemble des points qui annulent le gradient (appelés *point stationaires*) ne contient pas que des minima, comme le montre la Figure 2.4, dans le cas des fonctions d'une variable. D'une part, un minimum peut être local, mais d'autre part, afin de savoir si nous avons affaire à un minimum, un maximum ou un point d'inflexion, il convient d'étudier également le signe de la dérivée seconde.

Quelques petits théorèmes utiles facilitent le calcul avec les gradients:

$$\begin{array}{llllll} (1) & \|x\|^2 & = & x^t x & & \\ (2) & \nabla(b^t x) & = & \nabla(x^t b) & = & b \\ (3) & \nabla(x^t A x) & = & (A + A^t)x & = & 2Ax \text{ si } A \text{ est symétrique} \end{array}$$

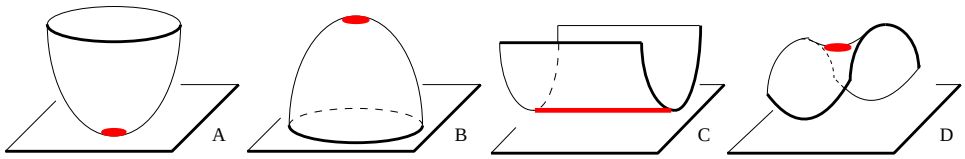


Figure 2.5: *Points stationaires. A: minimum local (Hessien défini positif); B: maximum local (Hessien défini négatif); C: goutière (Hessien positif mais non défini); D: point selle*

(1) est juste une facilité d'écriture, permettant de traiter de manière unifiée la norme au carré d'un vecteur et le produit de matrices (nous en verrons des utilisations par la suite). Il est également trivial de vérifier (2). Pour vérifier (3), écrivons $x^t Ax$ et calculons ses dérivées par rapport aux x_i:

$$\begin{aligned}
x^t Ax &= \sum_{k_1}\sum_{k_2} a_{k_1,k_2} x_{k_1} x_{k_2} \\
\frac{\partial x^t Ax}{\partial x_i} &= \frac{\partial}{\partial x_i}\left(a_{i,i}x_i^2 + x_i \sum_{k_2 \neq i} a_{i,k_2} x_{k_2} + x_i \sum_{k_1 \neq i} a_{k_1,i} x_{k_1} \right) \\
&= 2a_{i,i}x_i + \sum_{k_2 \neq i} a_{i,k_2} x_{k_2} + \sum_{k_1 \neq i} a_{k_1,i} x_{k_1} \\
&= \sum_{k_2} a_{i,k_2} x_{k_2} + \sum_{k_1} a_{k_1,i} x_{k_1} \\
&= \sum_j (a_{i,j} + a_{j,i}) x_j
\end{aligned}$$

Cette dernière expression correspond bien à la i-ième ligne de $(A + A^t)x$ □

Le Hessien

L'information différentielle à l'ordre 2 est représentée par la matrice de toutes les dérivées secondes par rapport à tous les couples de variables (x_i, x_j), appellée Hessien de F, et notée $\nabla^2 F$:

$$\nabla^2 F = \begin{bmatrix} \partial^2/\partial_{x_1,x_1} & \dots & \partial^2/\partial_{x_1,x_n} \\ \vdots & & \vdots \\ \partial^2/\partial_{x_n,x_1} & \dots & \partial^2/\partial_{x_n,x_n} \end{bmatrix}$$

De même que pour les fonctions uni-variées, il est possible de calculer un développement limité (i.e. série de Taylor) pour une fonction de plusieurs variables, autour d'un point x_0, donnant une bonne approximation de $F(x_0 + p)$ en fonction du vecteur déplacement p autour de x_0:

$$F(x_0 + p) \simeq F(x) + p^t \left(\nabla F(x_0)\right) + 1/2 p^t \left(\nabla^2 F(x_0)\right) p$$

Comme nous l'avons évoqué à propos du gradient, le Hessien, à savoir l'équivalent de la dérivée seconde pour les fonctions de plusieurs variables, permet de caractériser les points stationaires. La Figure 2.5 montre les configurations pouvant être rencontrées dans le cas de fonctions de deux variables. Suivant le signe des valeurs propres du Hessien, nous sommes en présence d'un minimum local (A), d'un maximum local (B), d'une goutière (C) ou d'un point selle (D). Par conséquent, pour trouver le minimum d'une fonction, annuler

le gradient ne suffit pas. Dans le cas général, même si le Hessien présente des valeurs propres positives, nous pouvons très bien être en présence d'un minimum local.

2.2.4 Optimisation quadratique

Formes quadratiques

Une forme quadratique est une fonction polynomiale dont le degré maximum des termes ne dépasse pas 2. Une forme quadratique peut toujours s'écrire sous la forme suivante:

$$F(x) = 1/2x^t Gx + x^t c + \alpha$$

avec G une matrice $n \times n$ symétrique, c un vecteur de $\mathbb{R}^n$, et α un scalaire. Il est relativement facile de trouver le minimum d'une forme quadratique, en annulant son gradient:

$$\nabla F(x) = 0 \Leftrightarrow Gx + c = 0$$

Ainsi, trouver le minimum d'une forme quadratique ou résoudre un système linéaire symétrique sont deux problèmes équivalents.

Moindres carrés

Supposons que l'on souhaite résoudre un système d'équation pour lequel le nombre d'équations m est supérieur au nombre d'inconnues n:

$$\begin{cases} a_{1,1}x_1 + \ldots + a_{1,n}x_n & = & b_1 \\ & \vdots & \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n & = & b_m \end{cases}$$

soit $Ax = b$. Dans le cas général, le système n'a pas de solution. Il est alors possible de tenter de trouver la "moins mauvaise" solution, à savoir celle qui minimise la somme des résidus au carré de chaque équation:

$$F(x) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{i,j}x_j - b_i \right)^2 = \|Ax - b\|^2$$

La fonction F est une forme quadratique, qui peut également s'écrire (théorème (1)):

$$F(x) = \|Ax - b\|^2 = (Ax - b)^t (Ax - b) = x^t A^t Ax - 2x^t A^t b + b^t b$$

(les deux termes $x^t A^t b$ et $b^t Ax$ sont égaux, car il s'agit de scalaires, autrement dit de matrices 1×1 qui sont symétriques). Le vecteur x qui minimise F annule le gradient de F, d'où (théorèmes (2) et (3)):

$$\nabla F(x) = 2A^t Ax - 2A^t b = 0$$

Nous retrouvons ainsi la formule classique des *moindres carrés*, appelée également *régression linéaire*, à savoir, le vecteur x qui minimise $\|Ax - b\|^2$ satisfait $A^t Ax = A^t b$.

Moindres carrés avec suppression de degrés de libertés

Pour certains algorithmes, il peut être utile de restreindre les degrés de libertés du système, en fixant certains paramètres de la fonction objectif. En termes formels, ceci revient à partitionner le vecteur de paramètres x en deux sous-vecteur $x = [x_f | x_l]$, dont les nf premières composantes $x_f = [x_1 \ldots x_f]$ correspondent aux paramètres libres, et les $n - nf$ dernières composantes $x_l = [x_{nf+1} \ldots x_n]$ correspondent aux paramètres fixés. La fonction F ne dépend à présent que du vecteur x_f, et s'exprime ainsi:

$$F(x_f) = \|Ax - b\|^2 = \left\| [A_f \mid A_l] \begin{bmatrix} x_f \\ \hline x_l \end{bmatrix} - b \right\|^2$$

Le partitionnement du vecteur x en deux sous-vecteur $[x_f, x_l]$ induit un partitionnement de la matrice A en deux sous-matrices A_f, A_l (dans le produit Ax, les coefficients de A pondérant des x_f sont dans A_f, et ceux qui pondèrent des x_l sont dans A_l). Il est alors possible de séparer les termes en x_f:

$$F(x_f) \quad = \quad \|A_f x_f + A_l x_l - b\|^2$$

En notant $b' = A_l x_l - b$ et en utilisant la formule des moindres carrés démontrée dans la section précédente, nous trouvons l'équation satisfaite par la valeur de x_f qui minimise F:

$$A_f^t A_f x_f = A_f^t b' \quad \text{soit} \quad A_f^t A_f x_f = A_f^t b - A_f^t A_l x_l$$

2.2.5 Optimisation non-linéaire

Nous nous intéressons à présent au problème plus difficile de la minimisation de fonctions non-linéaires. Dans cette section, nous présentons la méthode de Newton, permettant de calculer un point fixe d'une telle fonction non-linéaire. D'autres algorithmes existent, le lecteur pourra se réferrer aux ouvrages de références[NW00].

Fonctions de une seule variable

Considérons tout d'abord le cas des fonctions d'une seule variable. L'algorithme suivant calcule un point fixe de la fonction f de manière itérative:

tant que $|f'(x)| > \varepsilon$
  $p \leftarrow -f'(x)/f''(x)$
  $x \leftarrow x + p$
fin // *tant que*

L'algorithme fonctionne de la manière suivante: à chaque étape, autour du point x courant, la fonction f est approximée par un développement limité à l'ordre 2, appelé *la fonction modèle* car elle donne une bonne approximation (un bon modèle) des variations de f autour du point x en fonction d'un déplacement p:

$$f(x+p) \simeq \tilde{f}_x(p) = f(x) + pf'(x) + p^2/2f''(x)$$

Ensuite, l'algorithme calcule le déplacement p qui permet d'atteindre le minimum de la fonction modèle, à savoir:

$$\begin{array}{lcl} \tilde{f}'_x(p) & = & 0 \\ f'(x) + pf''(x) & = & 0 \\ p & = & -f'(x)/f''(x) \end{array}$$

L'algorithme se place alors au point $x+p$, qui minimise la fonction modèle, et réévalue la fonction modèle au nouveau point x ainsi obtenu.

[NW00] Nocedal and Wright. *Numerical Optimization*. Springer, 2000.

Fonctions de plusieurs variables

Il est relativement facile d'adapter cet algorithme au cas des fonctions de plusieurs variables. Comme nous l'avons vu dans la section précédente, le développement limité à l'ordre 2 d'une fonction F de plusieurs variables est donné par:

$$F(x+p) \simeq \tilde{F}_x(p) = F(x) + p^t\,(\nabla F(x)) + 1/2 p^t\left(\nabla^2 F(x)\right) p$$

Il s'agit bien sûr d'une forme quadratique. Trouver le déplacement p qui la minimise s'exprime alors ainsi:

$$\begin{array}{lcl} \nabla \tilde{F}_x(p) & = & 0 \\ (\nabla F(x)) + \left(\nabla^2 F(x)\right) p & = & 0 \\ p & = & -\left(\nabla^2 F(x)\right)^{-1} (\nabla F(x)) \end{array}$$

L'algorithme de Newton pour les fonctions de plusieurs variables s'écrit alors:

tant que $\|\nabla F(x)\| > \varepsilon$
  résoudre $(\nabla^2 F(x))p = -(\nabla F(x))$
  $x \leftarrow x + p$
fin // *tant que*

Autrement dit, pour minimiser une fonction non-linéaire, l'algorithme minimise une série de formes quadratique (en résolvant une série de systèmes linéaires).

Cas particulier d'une somme de carrés

Nous nous intéressons à présent à la méthode de Newton, dans le cas particulier où la fonction F à minimiser s'exprime sous la forme d'une somme de carrés:

$$F(x) = 1/2 \sum_{i=1}^{m} f_i(x)^2$$

Dans ce cas particulier, il est possible de simplifier légèrement les calculs, en trouvant une expression du gradient ∇F et du Hessien $\nabla^2 F$ (qui interviennent dans l'algorithme de Newton), en fonction de la matrice Jacobienne de la fonction f à valeurs dans $\mathbb{R}^m$ définie par $f = [f_1 f_2 \ldots f_m]$. La matrice Jacobienne J est une matrice $m \times n$ qui contient toutes les dérivées premières des f_i par rapport à toutes les variables x_j:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Le gradient ∇F et le Hessien $\nabla^2 F$ sont alors donné par:

$$\begin{array}{lcl} \nabla F & = & J^t F \\ \nabla^2 F & = & J^t J + Q \\ \text{avec } Q & = & \sum\limits_{i=1}^{m} f_i(x) G_i(x) \end{array}$$

où G_i dénote le Hessien de la fonction f_i.

Dans la plupart des cas, le terme Q qui regroupe les termes d'ordre plus grand que 2, peut être négligé (sauf pour les problèmes dits *à grands résidus*). En négligeant ce terme Q, on obtient alors l'algorithme de Gauss-Newton :

tant que $\|\nabla F(x)\| > \varepsilon$
  résoudre $J^t J p = -J^t F$
  $x \leftarrow x + p$
fin // *tant que*

Dans certain cas, le Hessien $J^t J$ peut devenir singulier (ce qui implique que la matrice du système linéaire à résoudre n'est pas inversible). Toutefois, le problème provient de la présence d'une infinité de solutions (plutôt que de l'absence de solution). En conséquence, il est possible d'utiliser *l'inverse généralisée de Moore-Penrose* afin de trouver une des solutions du systèmes:

$$(J^t J)^\dagger = \lim_{\varepsilon \to 0} (J^t J + \varepsilon I)^{-1} J$$

où I dénote la matrice identité.

En pratique, on utilise une faible valeur de ε (par exemple un petit pourcentage de la trace de $J^t J$). Ceci définit l'algorithme de Levenberg-Marquardt:

tant que $\|\nabla F(x)\| > \varepsilon$
  résoudre $(J^t J + \varepsilon I) p = -J^t F$
  $x \leftarrow x + p$
fin // *tant que*

2.2.6 Optimisation contrainte - méthode de Lagrange

Nous nous intéressons à présent au problème de l'optimisation sous contrainte, à savoir des problèmes du type:

$$\text{minimiser } F(x)$$

$$\text{avec} \begin{cases} G_1(x) & = & 0 \\ G_2(x) & = & 0 \\ & \vdots & \\ G_m(x) & = & 0 \end{cases}$$

où F et $G_1, G_2 \ldots G_m$ sont des fonctions de $\mathbb{R}^n$ dans $\mathbb{R}$. Les fonctions G_i sont appelées des *contraintes*. Le théorème KKT de Karush, Kuhn et Tucker [NW00] indique que le minimum de F qui satisfait les contraintes G_i est également un point fixe de la fonction L, définie ainsi:

$$L(x, \lambda) = F(x) + \sum_{i=1}^{m} \lambda_i G_i(x)$$

La fonction L dépend de m paramètres λ_i additionels, associés aux m contraintes G_i.

La fonction L est une fonction non-linéaire (de x et de λ), dont les points stationaires peuvent être trouvés par la méthode de Newton, présentée dans la section précédente.

2.3 Analyse fonctionelle

Les méthodes d'optimisation que nous avons vues dans la section précédente permettent de résoudre des systèmes linéaires et d'optimiser des fonctions de plusieurs variables. Toutefois, les problèmes de modélisation géométrique ne se formalisent en général pas directement sous cette forme, mais plutôt sous la forme d'équation aux dérivées partielles et d'optimisation variationelle. Pour cette raison, nous verrons dans cette section quelques notions d'analyse fonctionelle et de théorie des éléments finis permettant de discrétiser de

[NW00] Nocedal and Wright. *Numerical Optimization*. Springer, 2000.

telles équations différentielle et de les traduire en problèmes d'optimisation numériaue. Le lecteur désireux d'en apprendre plus sur le sujet pourra se réferrer à l'article d'Arvo[Arv95], qui explique l'importance de ces notions dans le domaine de la simulation numérique de la lumière. Dans notre contexte, nous utiliserons également ces notions comme formalisme de base pour le traitement numérique de la géométrie.

2.3.1 Espaces de Hilbert

Soit une surface S, et soit X l'espace des fonctions à valeurs dans $\mathbb{R}$ définies sur S. Etant donnée une norme $\|.\|$, l'espace de fonctions X est dit *complet* par rapport à cette norme si les suites de Cauchy convergent dans X. Une suite de Cauchy est une suite de fonctions $f_1, f_2 \ldots$ telle que $\lim_{n,m->\infty} \|f_n - f_m\| = 0$. Un espace vectoriel complet est appelé un *espace de Banach*.

L'espace X est qualifié d'*espace de Hilbert* dans le cas spécifique où la norme est définie par $\|f\| = \sqrt{< f, f >}$, où $< .,. >$ dénote un produit scalaire. Une définition possible de produit scalaire est donnée par $< f, g >= \int_S f(x)g(x)dx$, ce qui définit la norme L_2.

L'une des possibilités intéressantes offertes par ce niveau de structure mathématique est la possibilité de définir des *bases de fonctions* et de projeter d'autres fonctions sur elles, en utilisant le produit scalaire (exactement de la même manière que l'on projette un vecteur sur une base de vecteurs).

Dans une base de fonctions (Φ_i), une fonction f peut s'exprimer sous la forme $f = \sum \alpha_i \Phi_i$. De même qu'en géométrie, une base de fonction (Φ_i) est dite *orthonormale* si $\|\Phi_i\| = 1$ pour tout i et si $< \Phi_i, \Phi_j >= 0$ pour tout $i \neq j$. Si nous poursuivons l'analogie avec le cas géométrique, étant donnée une fonction f, il est très facile de calculer ses "coordonnées" α_i dans une base orthonormale (Φ_i), en *projetant* la fonction f sur cette base, ce qui donne $\alpha_i =< f, \Phi_i >$.

2.3.2 Opérateurs

Nous pouvons à présent donner quelques définitions de base concernant les *opérateurs*. D'une manière simple, un opérateur est une fonction de fonctions (i.e. de X dans X). Un opérateur L appliqué à une fonction f se note Lf (ce qui définit une autre fonction de X). Un operateur L est dit *linéaire* si $L(\lambda F) = \lambda Lf$ pour tout $f \in X, \lambda \in \mathbb{R}$. Une *fonction propre* d'un opérateur L est une fonction f telle que $Lf = \lambda f$. Le scalaire λ est appelé une *valeur propre* de L. En d'autres termes, appliquer un opérateur L à l'une de ses fonctions propre résulte en une version "amplifiée" de la fonction (le facteur d'amplification étant donné par la valeur propre λ).

Un opérateur L est dit *Hermitien* (ou à symétrie Hermitienne)[2] si $< Lf, g >=< f, Lg >$ pour toute $f, g \in X$. Une propriété importante des opérateurs Hermitiens est que les fonctions propres associées à des valeurs propres différentes sont orthogonale. Il est très facile de démontrer cette propriété, en considérant deux fonctions propres f, g associées à deux valeurs propres différentes λ, μ:

$$\begin{aligned} < Lf, g > &= < f, Lg > \\ < \lambda f, g > &= < f, \mu g > \\ \lambda < f, g > &= \mu < f, g > \end{aligned}$$

ce qui donne le résultat $(< f, g >= 0)$ étant donné que $\lambda \neq \mu$ $\square$

[2] la définition générale des opérateurs Hermitiens implique des fonctions de $\mathbb{C}$ dans $\mathbb{C}$. Dans le cadre de ce mémoire, nous considérerons le cas particulier des fonctions de $\mathbb{R}$ dans $\mathbb{R}$.

[Arv95] James Arvo. The Role of Functional Analysis in Global Illumination. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 115–126, New York, NY, 1995. Springer-Verlag.

Nous verrons plus loin (Section 4.5) comment l'utilisation de cette propriété est susceptible de permettre la définition de bases de fonctions sur des surfaces (ou plus généralement sur des variétés) de genre arbitraires.

2.3.3 Modélisation par élements Finis

Considérons à présent une équation aux dérivées partielles linéaire, à savoir qui s'exprime sous la forme $Lf = g$, où L est un opérateur linéaire, f est la fonction inconnue recherchée, et la fonction g est le second membre. La formulation par élements finis classique (Galerkin) projette cette équation sur une base de fonctions (ϕ^k). Dans notre cas, cette base de fonction (ϕ^k) est également utilisée pour représenter la solution, ce qui conduit à la formulation suivante:

$$\begin{aligned} f &= \sum_{k=1}^{n} \alpha_k \phi^k \\ g &= \sum_{k=1}^{n} \beta_k \phi^k \\ \forall j, < Lf, \phi^j > &= < g, \phi^j > \end{aligned}$$

qui peut s'exprime sous forme matricielle:

$$Q\alpha = B\beta$$

avec $Q_{i,j} =< L\phi^i, \phi^j >$, $B_{i,j} =< \phi^i, \phi^j >$ et où α dénote le vecteur $[\alpha_1, \alpha_2, \ldots \alpha_n]$. La matrice Q est appelée *matrice de rigidité*, et B est appelée *matrice de masse*.

2.4 Traitement numérique de la géométrie

Les outils mathématiques de base que nous avons vu dans les deux sections précédentes, à savoir l'optimisation numérique (Section 2.2) et l'analyse fonctionelle (Section 2.3.3) nous permettent à présent d'aborder le vif du sujet, à savoir le traitement numérique de la géométrie (geometry processing). Nous en dressons ici un état de l'art rapide, pour en exhiber quelques problèmes ouverts particulièrement intéressants. Les chapitres suivants présenterons alors nos contributions pour tenter de "casser" ces problèmes.

Le traitement numérique de la géométrie est récemment apparu (dans le milieu des années 90) comme une stratégie prometteuse pour résoudre les problèmes particuliers rencontrés lors de la manipulation de maillage pouvant comporter des millions d'éléments. Comme un maillage peut être considéré comme un *échantillonage* d'une surface - en d'autres termes un échantillonage d'un certain signal - le formalisme du *traitement du signal numérique* s'est révélé être un choix naturel pour établir les bases théoriques de cette discipline (voir par exemple [Tau95a]). Cette discipline a ensuite étudié différents aspects de ce formalisme, appliqué à des problèmes de modélisation géométrique. Nous introduirons rapidement les aspects principaux de cette discipline, à savoir l'analyse multi-résolution, le lissage discret et la paramétrisation de maillages.

La modélisation multi-résolution : En étudiant les aspects liés à l'échantillonage des objets, la notion de *résolution* est naturellement apparue comme jouant un rôle central. Pour capturer les variations d'un signal, la loi de Shannon-Nyquist impose que la fréquence d'échantillonage soit au moins double de celle des plus hautes fréquences apparaissant dans le spectre de ce signal. Dans le cas d'un signal présentant un large spectre, ceci résulte en

[Tau95a] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH Conference Proceedings*, pages 351–358. ACM, 1995.

une perte d'espace de stockage et de temps de calcul. Les méthodes dites multi-résolution fournissent une réponse naturelle à ce problème, en subdivisant le signal en un ensemble de composantes (appelées des harmoniques). Ceci donne la possibilité d'utiliser une fréquence d'échantillonage différente pour chaque harmonique. Peter Schroeder a été l'un des pionniers qui ont appliqué ce formalisme (en utilisant en particulier une famille de fonctions appelées des ondelettes) aux problèmes d'illumination globale [Sch94], et Steven Gortler les a appliquées à des problèmes de modélisation géométrique. Toutefois, alors qu'il est relativement aisé de construire une représentation hiérarchique pour un signal dépendant d'un seul paramètre (tel que le son qui dépend du temps), contruire une représentation multi-résolution d'un objet géométrique (par exemple une surface 3D) s'est révélé être un problème non-trivial. Motivé par des problématiques de simplification des données et de visualisation fondées sur des niveaux de détails, Huggues Hoppe a développé la structure de données appelée *maillage progressif* (Progressive Mesh)[Hop96]. Un maillage progressif se compose d'un maillage de base (simplifié à l'extrème) et d'une suite d'opérations de raffinement (permettant de ré-introduire les détails géométriques dans le maillage de base). La transformation d'un objet en un maillage progressif sépare naturellement les différentes fréquences géométriques apparaissant dans l'objet initial. En se fondant sur cette simple observation, les maillages progressifs (ainsi que d'autres structures de données similaires) ont été utilisées pour construire d'une part des représentations géométriques multi-résolution[EDD+95], et d'autre part les outils de modélisation et de traitement multi-résolution permettant de les modifier[KCVS98], [GSS99a].

Le lissage discret : Un autre défi de la discipline "traitement du signal géométrique" concerne l'adaptation aux représentations maillées des outils disponibles pour les représentations classiques de type "courbes et surfaces" : Dans ces représentations, la géométrie est représentée par un ensemble de surfaces paramétriques. De nombreux travaux se sont consacrés à la définission d'outils permettant d'optimiser la forme d'une surface, en optimisant un critère de "lissage" (fairness). Ces critères sont définis à partir de notions de géométrie différentielle (courbure moyenne, courbure Gaussienne ...), ou à partir d'approximations de grandeurs physiques (énergie "plaques minces"). D'une manière générale, optimiser ce critère revient à résoudre une équation aux dérivées partielles[BW90]. L'adaptation de ce formalisme au cas de maillages (cas discret) a été un domaine de recherche très actif. Kobbelt a proposé le terme *lissage discret* (discrete fairing)[Kob97] pour qualifier cette famille d'approches.

La paramétrisation des surfaces maillées: La paramétrisation des maillages est un autre problème pour lequel la communauté "traitement du signal géométrique" a consacré beaucoup d'activité pendant ces dernières années. Dans le cas classique des représentations de type "courbes et surfaces", les modèles géométriques sont représentés par des ensembles de fonctions de deux paramètres (représentation paramétrique). Cette représentation est utile pour plusieurs domaines d'applications. En particulier, il est facile d'attacher des propriétés physiques à ces surfaces, en les représentant par des structures de données stockées dans l'espace paramétrique. D'autre part, certains problèmes, tels que les problèmes

[Sch94] P. Schroeder. *Wavelet Methods for Global Illumination*. PhD thesis, Princeton University, 1994.

[Hop96] H. Hoppe. Progressive meshes. In *SIGGRAPH Conf. Proc.*, pages 99–108. ACM, 1996.

[EDD+95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH Conference Proceedings*, pages 173–182. ACM, 1995.

[KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH Conference Proceedings*, pages 105–114, 1998.

[GSS99a] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *SIGGRAPH Conference Proceedings*, pages 325–334. ACM, 1999.

[BW90] M.I.G. Bloor and M.J. Wilson. Using PDEs to generate free-form surfaces. *CAD*, 22, 1990.

[Kob97] L. Kobbelt. Discrete fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1997.

de re-maillage, trouvent une formulation plus simple dans cet espace. Pour cette raison, la communauté à cherché à développer des méthodes permettant de construire une représentation paramétrique à partir d'un maillage. Michael Floater a été l'un des pionniers de cette discipline. Motivé par des problèmes d'ajustement de Splines à des données, il a eu l'idée d'appliquer le théorème de Tutte caractérisant les plongements de cartes planaires, afin de construire une paramétrisation linéaire par morceaux à partir d'un maillage triangulé homéomorphe à un disque. Un grand nombre d'articles traitant de ce problème spécifique ont été publié. Par exemple, certaines méthodes relachent la contrainte de fixer le bord sur un polygone convexe dans l'espace paramétrique. D'autres méthodes utilisent différents critères de déformations à minimiser, adaptés à différents domaines d'applications. A présent, le congrès SIGGRAPH, principale conférence du domaine, dévoue une session entière à ce problème spécifique, et Michael Floater identifie plus de 20 articles dédiés à ce problème, publiés chaque années dans les principaux journaux et conférences. Son état de l'art[FH04] liste les avancées les plus significatives dans ce domaine.

Problèmes ouverts : Malgrès les nombreuses avancées réalisées dans le domaine du traitement du signal géométrique, d'importants problèmes restent ouvert. Comme nous l'avons mentionné dans la Section 2.4, même si l'acquisition et le filtrage de données géométriques est devenu bien plus facile qu'il y a 30 ans, un maillage scanné composé de 10 millions de triangles ne peut pas être utilisé directement dans des applications de visualisation temps réel ou dans des processus de simulation numérique. Pour cette raison, il est nécessaire de développer de nouvelles méthodes automatiques, permettant de convertir ces grands maillages en des représentations de plus haut niveau. Ces méthodes automatiques n'existent pas encore. Par exemple, le pionnier Henri Gouraud mentionne souvent dans ses conférences que le problème de l'acquisition des données est toujours ouvert. Malcolm Sabin, un autre pionnier, renommé pour ses travaux dans le domaine des "courbes et surfaces" et dans le domaine des "surfaces de subdivision" identifie plusieurs défis à relever en traitement du signal géométrique. L'un de ces défis concerne la mise au point d'une méthode permettant de construire un maillage de contrôle optimum afin d'approximer une surface donnée. De manière plus générale, convertir un maillage en une représentation de plus haut niveau, à savoir un ensemble d'équation, est un problème difficile n'ayant pas encore reçu de solution satisfaisante. C'est l'un des objectifs à long terme d'initiatives internationales, telles que le réseau d'excellence Européen `AIM@Shape`[3] dont l'équipe ALICE fait partie.

[3] http://www.aim-at-shape.net

[FH04] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In M. S. Floater N. Dodgson and M. Sabin, editors, *Advances on Multiresolution in Geometric Modelling*. Springer-Verlag, 2004.

Chapitre 3

Paramétrisation

Ce chapitre est une combinaision de la section *Setting the boundary free* de mes notes de cours pour le cours *Mesh Parameterization, Theory and Practice*, que j'ai co-organisé à la conférence Siggraph en 2007, avec Alla Sheffer et Kai Hormann [HLS+07], et de la section *Mesh parameterization* que j'ai rédigée pour le cours Siggraph *3D Modeling with Mesh Models*, organisé par Mario Botsch et Leif Kobbelt [BPK+07]. Ce dernier sera publié l'année prochaine (Juillet 2008) sous la forme d'un ouvrage (*Polygonal Mesh Processing*, aux éditions AK Peters).

Dans ce chapitre, nous introduisons la notion de paramétrisation de surfaces, et en particulier l'analyse des déformations en utilisant comme outil la première forme fondamentale. Nous nous pencherons ensuite sur le cas des surfaces triangulées, et montrerons comment en transposant cette analyse des déformations dans le cas discret, il est possible de définir de nouvelles méthodes de paramétrisation de surfaces. Dans ce contexte, nous présenterons nos contributions suivantes:

- ◇ Paramétrisation de surfaces discontinues [LM98], [LJL00]
- ◇ Plaquage de textures sous contraintes [Lé01]
- ◇ Applications conformes au sens des moindres carrés [LPRM02]
- ◇ Paramétrisation à base d'angles [SLMB04]
- ◇ Application au re-maillage [ACSD+03]
- ◇ Application à l'extrapolation de surfaces [Lé03]
- ◇ Application à la simulation numérique de la lumière [LLAP05]

3.1 Notion de paramétrisation

Cette section donne les définitions de base liées à la notion de paramétrisation, ainsi que quelques exemples. Le lecteur déjà familier avec cette notion pourra se rediriger directement vers la section suivante. Une paramétrisation d'une surface 3D est une fonction qui met cette surface en correspondance bijective avec un domaine 2D. Cette notion joue un rôle fondamental en géométrie numérique, car elle permet de transformer des problèmes 3D difficiles dans un espace 2D où leur résolution devient bien plus simple. La prochaine section a pour objectif de rendre cette idée plus intuitive.

3.1.1 La carte du monde et les coordonnées sphériques

Considérons le problème posé par le dessin d'une carte du monde. Comme nous pouvons le voir sur la Figure 3.1, le problème consiste à trouver une bonne manière de "déplier" la surface du monde, afin d'obtenir une surface 2D plate. Comme la surface du monde

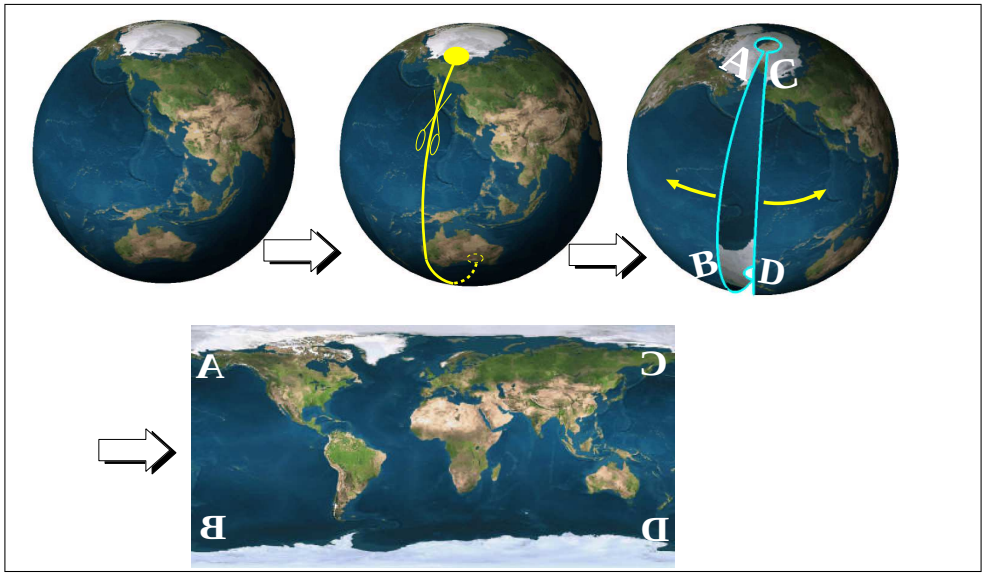


Figure 3.1: *Découpez moi un méridien, et je déplierai le monde ...*

est fermée, pour la déplier, il sera nécessaire de la découper. Par exemple, il est possible de la découper le long d'un méridien, à savoir une courbe qui joint les deux poles. Lors du dépliage, nous pouvons remarquer que les deux pôles sont très déformés, et deviennent deux courbes. Le pôle Nord devient le segment $[A-C]$, et le pôle Sud devient le segment $[B-D]$[1]. Nous remarquons également que le méridien le long duquel la sphère a été découpée correspond à deux courbes différentes: les segments $[A-B]$ et $[C-D]$. En d'autres termes, si une ville était située exactement sur ce méridien, elle apparaitrait deux fois sur la carte. Comme nous le montrons sur la Figure 3.2, il est possible de donner à chaque point de la carte deux coordonnées (θ,ϕ). Dans le cas de la projection utilisée sur la Figure 3.1, les coordonnées (x,y,z) dans l'espace 3D et les coordonnées (θ,ϕ) sur la carte sont liées par l'équation suivante, appelée une équation *paramétrique* de la sphère :

$$\begin{array}{rcl} \theta & \in & [0\ldots 2.\pi], \\ \phi & \in & [-\pi\ldots\pi] \end{array} \mapsto \left\{ \begin{array}{rcl} x(\theta,\phi) & = & R.\cos(\theta).\cos(\phi) \\ y(\theta,\phi) & = & R.\sin(\theta).\cos(\phi) \\ z(\theta,\phi) & = & R.\sin(\phi) \end{array} \right. \tag{3.1}$$

où R dénote le rayon de la sphère. Nous pouvons voir que cette équation est assez différente de l'équation habituelle d'une sphère $x^2+y^2+z^2=R^2$, également appelée équation *implicite*. L'équation implicite fournit un moyen de tester si un point donné appartient à une sphère, alors que l'équation paramétrique fournit un moyen de transformer le rectangle $[0\ldots 2.\pi]\times[-\pi\ldots\pi]$ en une sphère.

Nous donnons les définitions suivantes, liées à l'équation paramétrique:

- ◇ Les coordonnées (θ,ϕ) en un point $\mathbf{p}=(x,y,z)$ sont appelées *coordonnées sphériques* au point $\mathbf{p}$.
- ◇ Les lignes verticales de la carte, définies par $\theta=$ Constante, correspondent chacune à une courbe sur la surface 3D, appelée une courbe *iso-*θ. Dans notre cas, les courbes iso-θ sont des cercles qui traversent les deux pôles de la sphère (à savoir les *méridiens* du globe).

[1] A noter que le dépliage utilisé dans une vraie planisphère peut être différent. Dans notre exemple, nous gardons un dépliage qui a une équation très simple, à savoir qui correspond à une paramétrisation plus simple que dans le cas d'une vraie carte du monde. L'article de Michael Floater [FH04] liste quelques unes des méthodes de projection utilisées par les cartographes.

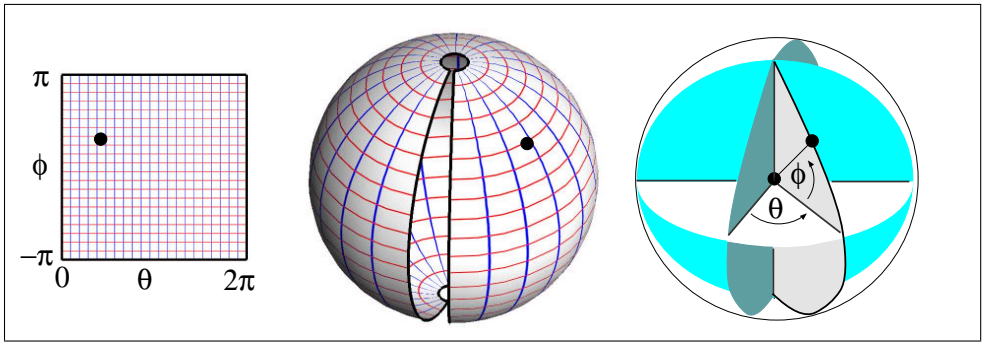


Figure 3.2: *Coordonnées sphériques*

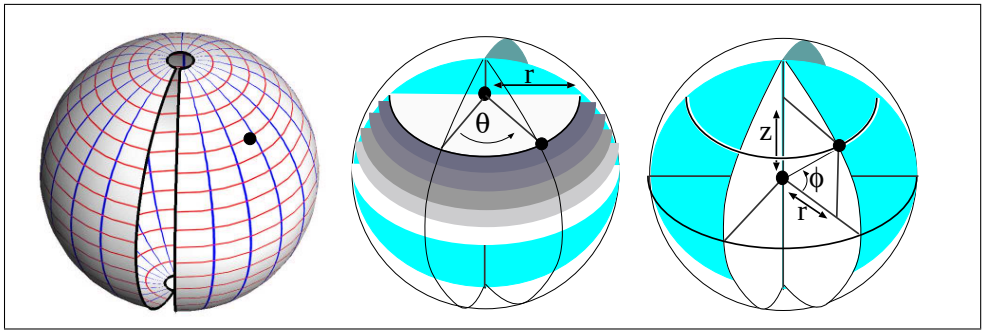


Figure 3.3: *Construire une sphère en empilant des disques*

◇ Les lignes horizontales de la carte, définies par $\phi =$ Constante correspondent chacune à une courbe *iso-*ϕ. Dans notre cas, les courbes iso-ϕ correspondent aux *parallèles* du globe, et la courbe iso-ϕ caractérisée par $\phi = 0$ correspond à l'*équateur*.

Comme nous pouvons le voir sur la Figure 3.2, dessiner les courbes iso-θ et iso-ϕ aide à comprendre comment la carte est *déformée* quand elle est appliquée sur la surface. Sur la carte, les courbes iso-θ et iso-ϕ sont respectivement des lignes verticales et horizontales, qui forment une grille régulière. Visualiser ce que cette grille devient en appliquant la carte sur la surface permet de voir par exemple que ceci cause de grandes déformations près des pôles. En effet, près des pôles, la forme des carrés de la grille est très nettement altérée. Nous verrons dans la section suivante comment caractériser et mesurer ces déformations. D'autre part, comme le lecteur pourrait légitimement se demander d'où cette équation de la sphère provient, nous tentons à présent de donner une explication intuitive des coordonnées sphériques.

Comme nous le montrons sur la Figure 3.3, nous pouvons considérer une sphère comme un "empilement" de cercles de rayons variables. Un point sur l'un de ces cercles à une hauteur donnée z peut alors être repéré par ses coordonnées $x = r\cos(\theta)$, $y = r\sin(\theta)$, où r dénote le rayon du cercle considéré (à savoir le rayon de la "tranche"), et où $\theta \in [0, 2.\pi]$ est un paramètre qui permet de "tourner autour" de la tranche. Comme nous pouvons le remarquer, r dépend de z (le rayon de la tranche dépend de l'endroit où la tranche est découpée). Comme nous le montrons sur la Figure 3.3, il est possible d'exprimer à la fois r et z en fonction d'un paramètre supplémentaire ϕ, sous la forme $r = R.\cos(\phi)$,$z = R.\sin(\phi)$, où R dénote le rayon de la sphère, et où $\phi \in [-\pi, \pi]$ permet de sélectionner la tranche. Ainsi, en remplaçant r et z par leur expression, nous obtenons l'équation paramétrique de la sphère (que nous avons déjà vue précédemment):

$$\begin{array}{rcl} \theta & \in & [0\ldots 2.\pi], \\ \phi & \in & [-\pi\ldots\pi] \end{array} \mapsto \left\{ \begin{array}{rcl} x(\theta,\phi) & = & R.\cos(\theta).\cos(\phi) \\ y(\theta,\phi) & = & R.\sin(\theta).\cos(\phi) \\ z(\theta,\phi) & = & R.\sin(\phi) \end{array} \right. \tag{3.2}$$

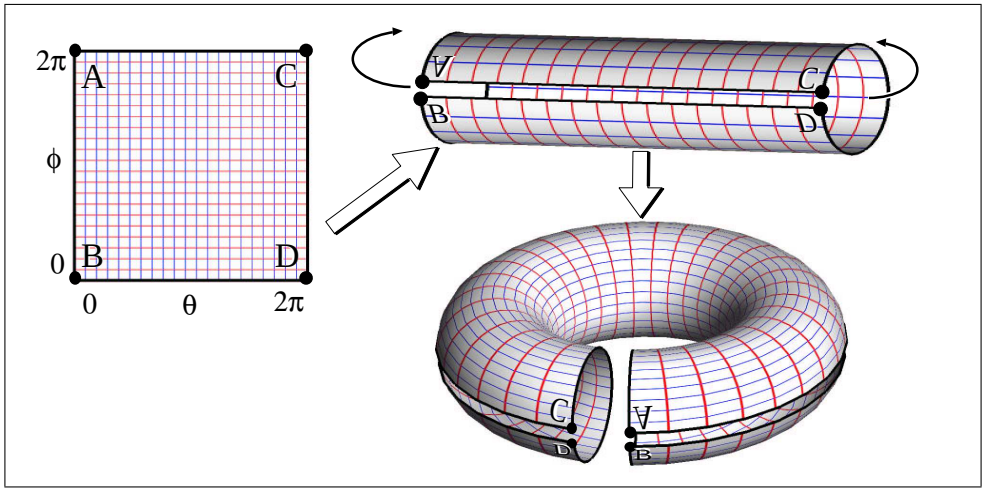


Figure 3.4: *Comment construire un tore à partir d'une feuille de papier*

Pour résumer, l'équation paramétrique de la sphère explicite "comment construire une sphère", alors que l'équation usuelle (implicite) permet de tester si un point donné appartient à la sphère. Chaque point de la sphère est repéré par un couple unique[2] de coordonnées θ et ϕ, ainsi, la paramétrisation définit un système de coordonnées (curvilignes) sur la sphère. Nous pouvons également remarquer que bien que chaque point de la sphère ait trois coordonnées (x,y,z), deux coordonnées (θ,ϕ) suffisent pour le repérer. En réalité, une surface (même plongée dans un espace 3D) est un objet 2D, dont la nature bi-dimentionelle peut être révélée en exhibant une paramétrisation. L'espace paramétrique peut alors être considéré comme une "carte" de la surface.

Avant d'aller plus loin, considérons un autre exemple. Comme nous le montrons sur la Figure 3.1.1, il est facile de transformer un domaine 2D carré en un tore. Une équation paramétrique du tore peut alors s'écrire:

$$\begin{array}{rcl} \theta & \in & [0\ldots 2.\pi], \\ \phi & \in & [0\ldots 2.\pi] \end{array} \mapsto \left\{ \begin{array}{rcl} x(\theta,\phi) & = & (R_2+R_1.\cos(\theta)).\cos(\phi) \\ y(\theta,\phi) & = & (R_2+R_1.\cos(\theta)).\sin(\phi) \\ z(\theta,\phi) & = & R_1.\sin(\theta) \end{array} \right. \tag{3.3}$$

3.1.2 Analyse des déformations et anisotropie

La matrice Jacobienne et la 1ère forme fondamentale

La section précédente a présenté quelques exemples simples de paramétrisation, et a mentionné le fait que cette paramétrisation déformait l'espace paramétrique (en particulier près des pôles de la sphère). Cette section considère à présent une surface pour laquelle une paramétrisation est connue, et décrit des moyens pour quantifier la manière dont l'espace paramétrique est déformé quand celui-ci est projeté sur la surface. Nous verrons ensuite comment ces notions permettent de construire une paramétrisation d'une surface tout en minimisant ces déformations.

Les dérivées premières de la paramétrisation jouent un rôle important dans l'analyse des déformations, il est donc important d'en saisir de manière intuitive leur interprétation géométrique. En physique, la mécanique des points matériels étudie le mouvement d'un objet soumis à des forces, et approximé par un unique point $\mathbf{p}$. La *trajectoire* est la courbe décrite par le point $\mathbf{p}$ quand t varie de t_0 à t_1, où t dénote le temps. La fonction qui met un instant t donné en correspondance avec la position $\mathbf{p}(t) = \{x(t),y(t),z(t)\}$ du point $\mathbf{p}$

[2] si nous exclueons les deux pôles et la ligne de découpe.

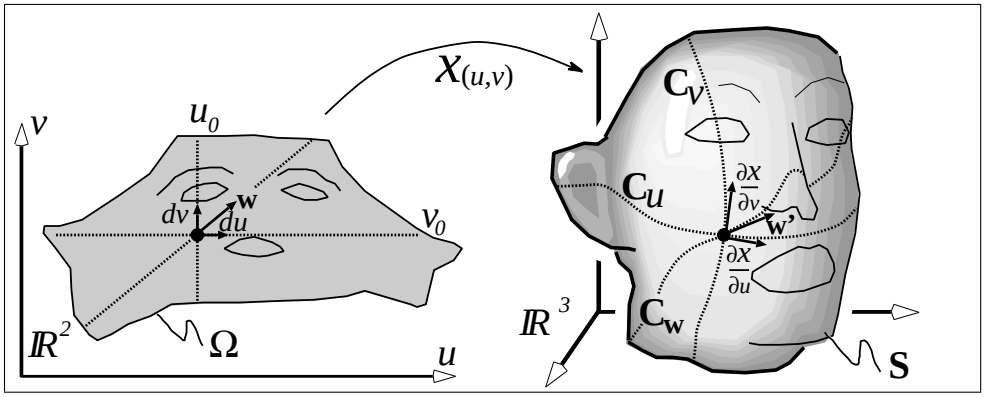


Figure 3.5: *Des déplacement élémentaires autour d'un point (u,v) de Ω le long des axes u et v sont transformés en des vecteurs tangents aux courbes iso-u et iso-v qui passent par le point $X(u,v)$*

est une paramétrisation de la trajectoire, i.e. une paramétrisation d'une *courbe*. Il est bien connu que le vecteur formé par les dérivées $\mathbf{v}(t) = \partial\mathbf{p}/\partial t = \{\partial x/\partial t, \partial y/\partial t, \partial z/\partial t\}$ correspond à la *vitesse* de $\mathbf{p}$ à l'instant t, et que ce vecteur est tangent à la trajectoire.

Comme nous le montrons sur la Figure 3.5, nous considérons à présent une fonction $X : (u,v) \mapsto (x,y,z)$, mettant un sous-espace Ω de $\mathbb{R}^2$ en correspondance bijective avec une surface $\mathbf{S}$ de $\mathbb{R}^3$. Les scalaires (u,v) sont les coordonnées dans l'espace paramétrique[3]. Dans le cas de la paramétrisation d'une courbe, la courbe est décrite par un seul paramètre t. Dans notre cas, nous considérons une paramétrisation de surface donnée par $X(u,v) = \{x(u,v), y(u,v), z(u,v)\}$, et nous avons donc **deux** paramètres, u et v. Ainsi, en en point donné (u_0,v_0) de l'espace paramétrique Ω, nous pouvons considérer deux vecteurs "vitesse": $(\partial X/\partial u)(u_0,v_0)$ et $(\partial X/\partial v)(u_0,v_0)$. Il est facile de vérifier que $(\partial X/\partial u)(u_0,v_0)$ correspond au vecteur "vitesse" de la trajectoire $\mathbf{C}_u : t \mapsto X(u_0+t, v_0)$ en $X(u_0,v_0)$ et que $(\partial X/\partial v)(u_0,v_0)$ correspond au vecteur "vitesse" de la trajectoire $\mathbf{C}_v : t \mapsto X(u_0, v_0+t)$. La courbe $\mathbf{C}_u$ (resp. $\mathbf{C}_v$) est l'iso-u (resp. l' iso-v) qui passe par $X(u_0,v_0)$, autrement dit l'image à travers la paramétrisation X de la ligne d'équation $u = u_0$ (resp. $v = v_0$).

A cette étape, nous pourrions penser que l'information fournie par les deux vecteurs "vitesse"$(\partial X/\partial u)(u_0,v_0)$ and $(\partial X/\partial v)(u_0,v_0)$ est largement insuffisante pour caractériser les déformations entre Ω et $\mathbf{S}$, dans les voisinnages de (u_0,v_0) et de son image $X(u_0,v_0)$. En réalité, ces deux vecteurs vitesse permettent de calculer comment dans le voisinage de (u_0,v_0), un vecteur arbitraire $w = (a,b)$ de l'espace paramétrique est transformé en un vecteur w' de la surface.

En d'autres termes, nous souhaitons calculer le vecteur "vitesse" $w' = \partial X(u_0+t.a, v_0+t.b)/\partial t$ de la courbe qui correspond à l'image d'une ligne droite $(u,v) = (u_0,v_0)+t.w$. Le vecteur w', à savoir la tangente à la courbe $\mathbf{C_w}$, peut être simplement calculé par la règle de composition des fonctions $((g \circ f)' = g' \circ f \times f')$. Ainsi, il est facile de vérifier que le vecteur w' s'exprime $w' = a.(\partial X/\partial u)(u_0,v_0) + b.(\partial X/\partial v)(u_0,v_0)$. Le vecteur $\mathbf{w}'$ est appelé *dérivée directionelle* de X en (u_0,v_0) par rapport à la direction $\mathbf{w}$.

Sous forme matricielle, $\mathbf{w}'$ s'exprime $\mathbf{w}' = J(u_0,v_0).\mathbf{w}$, où $J(u_0,v_0)$ est la matrice for-

[3]Comme les notations θ et ϕ utilisées dans la section précédente évoquent des angles, elles ne sont pas toujours appropriées dans le cas général, et nous préférons utilisées les notations plus neutres u et v pour dénoter ces paramètres.

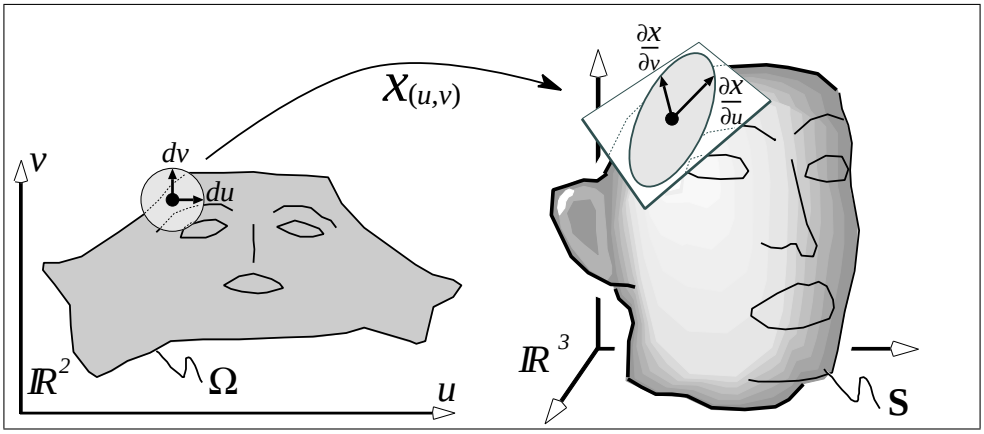


Figure 3.6: *Anisotropie: un cercle élémentaire est transformé en une ellipse.*

mée par l'ensemble des dérivées premières de X:

$$J(u_0,v_0) = \begin{bmatrix} \frac{\partial x}{\partial u}(u_0,v_0) & \frac{\partial x}{\partial v}(u_0,v_0) \\ \frac{\partial y}{\partial u}(u_0,v_0) & \frac{\partial y}{\partial v}(u_0,v_0) \\ \frac{\partial z}{\partial u}(u_0,v_0) & \frac{\partial z}{\partial v}(u_0,v_0) \end{bmatrix} = \begin{bmatrix} & \vdots & \\ \frac{\partial X}{\partial u}(u_0,v_0) & \vdots & \frac{\partial X}{\partial v}(u_0,v_0) \\ & \vdots & \end{bmatrix} \tag{3.4}$$

La matrice $J(u_0,v_0)$ est appelée *matrice Jacobienne* de X en (u_0,v_0) (nous avons déjà rencontré cette notion dans la Section 2.2 traitant des méthodes numériques).

La notion de dérivée directionelle permet de savoir comment un déplacement élémentaire $\mathbf{w}$ depuis le point (u_0,v_0) de l'espace paramétrique se comporte lorsqu'il est transformé par une fonction X. Nous souhaitons à présent comprendre comment cette transformation affecte les distance et les angles. Les distances et les angles peuvent être mesurées à l'aide du produit scalaire (le produit scalaire de deux vecteurs fournit le cosinus de l'angle entre ces deux vecteur, et le produit scalaire d'un vecteur avec lui-même nous fournit le carré de sa norme).

Nous allons donc nous intéresser au produit scalaire $\mathbf{w}'_1.\mathbf{w}'_2$ entre les images $\mathbf{w}'_1$ et $\mathbf{w}'_2$ de deux vecteurs $\mathbf{w}_1$ et $\mathbf{w}_2$. En réalisant les substitutions $\mathbf{w}'_1 = J.\mathbf{w}_1$ et $\mathbf{w}'_2 = J.\mathbf{w}_2$, le produit scalaire $\mathbf{w}'_1.\mathbf{w}'_2$ est alors donné par $\mathbf{w}'_1.\mathbf{w}'_2 = (J.\mathbf{w}_1)^t.(J.\mathbf{w}_2) = \mathbf{w}^t_1.J^t.J.\mathbf{w}_2$. La matrice $J^t.J$, appelée *1ère forme fondamentale* de X, notée G, s'exprime ainsi:

$$G(u_0,v_0) = J^t.J = \begin{bmatrix} \frac{\partial X}{\partial u}\cdot\frac{\partial X}{\partial u} & \frac{\partial X}{\partial u}\cdot\frac{\partial X}{\partial v} \\ \frac{\partial X}{\partial v}\cdot\frac{\partial X}{\partial u} & \frac{\partial X}{\partial v}\cdot\frac{\partial X}{\partial v} \end{bmatrix} \tag{3.5}$$

La 1ère forme fondamentale $G(u_0,v_0)$ est également appelée le *tenseur métrique* de X, car il permet de mesurer la manière dont les distances et les angles sont déformés par la paramétrisation dans le voisinage de (u_0,v_0). Le carré de la norme de l'image $\mathbf{w}'$ d'un vecteur $\mathbf{w}$ est donné par $||\mathbf{w}'||^2 = \mathbf{w}^t.G.\mathbf{w}$, et le produit scalaire $\mathbf{w}'_1.\mathbf{w}'_2 = \mathbf{w}^t_1.G.\mathbf{w}_2$ détermine ce que devient l'angle entre $\mathbf{w}_1$ et $\mathbf{w}_2$ à travers la paramétrisation[4]. La section suivante donne une interpretation géométrique de la 1ère forme fondamentale et de ses valeurs propres, qui déterminent l'*ellipse d'anisotropie*.

L'ellipse d'anisotropie

La section précédente a étudié la manière dont est transformé le *déplacement* élémentaire en partant de (u_0, v_0) dans la direction $\mathbf{w}$. Comme nous le montrons sur la Figure 3.6, notre objectif est à présent de déterminer ce que devient un *cercle* élémentaire.

Considérons à présent les deux valeurs propres λ_1, λ_2 de la matrice G, ainsi que les deux vecteurs propres associés $\mathbf{w}_1, \mathbf{w}_2$. Notons que puisque G est symétrique, ses vecteurs propres $\mathbf{w}_1$ et $\mathbf{w}_2$ sont orthogonaux[5]. Un vecteur unitaire $\mathbf{w}$ peut alors s'écrire sous la forme $\mathbf{w} = \cos(\theta).\mathbf{w}_1 + \sin(\theta).\mathbf{w}_2$. Le carré de la norme du vecteur transformé $\mathbf{w}' = J.\mathbf{w}$ s'écrit alors:

$$\begin{aligned} ||\mathbf{w}'||^2 &= \mathbf{w}^t.G.\mathbf{w} \\ &= (\cos(\theta).\mathbf{w}_1 + \sin(\theta).\mathbf{w}_2)^t.G.(\cos(\theta).\mathbf{w}_1 + \sin(\theta).\mathbf{w}_2) \\ &= \cos^2(\theta).||\mathbf{w}_1||^2.\lambda_1 + \sin^2(\theta).||\mathbf{w}_1||^2.\lambda_2 + \\ &\quad \sin(\theta).\cos(\theta)(\lambda_1.\mathbf{w}_2^t.\mathbf{w}_1 + \lambda_2.\mathbf{w}_1^t.\mathbf{w}_2) \\ &= \cos^2(\theta).\lambda_1 + \sin^2(\theta).\lambda_2 \end{aligned} \tag{3.6}$$

Dans l'Equation 3.6, les termes $\mathbf{w}_1^t.\mathbf{w}_2$ et $\mathbf{w}_2^t.\mathbf{w}_1$ s'annulent, car $\mathbf{w}_1$ et $\mathbf{w}_2$ sont orthogonaux. Calculons à présent les extrema de $||\mathbf{w}'||^2$ en fonction de θ:

$$\begin{aligned} \frac{\partial ||\mathbf{w}'(\theta)||^2}{\partial \theta} &= 2.\sin(\theta).\cos(\theta).(\lambda_2 - \lambda_1) \\ &= \sin(2\theta).(\lambda_2 - \lambda_1) \end{aligned} \tag{3.7}$$

Les extrema de $||\mathbf{w}'(\theta)||^2$ sont alors atteints pour $\theta \in \{0, \pi/2, \pi, 3\pi/2\}$, soit pour $\mathbf{w} = \mathbf{w}_1$ ou $\mathbf{w} = \mathbf{w}_2$. Par conséquent, les valeurs minimum et maximum de $||\mathbf{w}'(\theta)||^2$ sont les valeurs propres λ_1 et λ_2 de G. Ainsi:

- ◇ Les axes de l'ellipse d'anisotropie sont $J.\mathbf{w}_1$ et $J.\mathbf{w}_2$;
- ◇ Les longueurs de ces axes sont $\sqrt{\lambda_1}$ et $\sqrt{\lambda_2}$.

Note: les longueurs des axes $\sqrt{\lambda_1}$ et $\sqrt{\lambda_2}$ correspondent également aux valeurs singulières de la matrice J. Nous rappelons que la décomposition en valeurs singulière (SVD) de la matrice J s'écrit:

$$J = U\Sigma V^t = U \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix} V^t$$

avec $U : 3 \times 3$ et $V : 2 \times 2$ des matrices dont les vecteurs colonnes forment une base orthonormée (matrices dites également *unitaires*), et Σ est une matrice dont seul les éléments diagonaux σ_1, σ_2 sont non-nuls. Les réels σ_1, σ_2 sont appelés les *valeurs singulières* de J. Dans notre cas, en substituant la matrice Jacobinenne J par sa SVD, on obtient:

$$\begin{aligned} G &= J^t J \\ &= (U\Sigma V^t)^t (U\Sigma V^t) \\ &= V\Sigma^t U^t U \Sigma V^t \quad = \quad V\Sigma^t \Sigma V^t \\ &= V \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} V^t \end{aligned}$$

Puisque la matrice U est unitaire, le terme central $U^t U$ de la troisième ligne est égal à la matrice unité et se simplifie donc dans les calculs. Nous obtenons ainsi la SVD de la matrice G, qui est également une diagonalisation de G. Par unicité de la SVD, nous en

[4] Petit rappel sur le produit scalaire de deux vecteurs: $\mathbf{a}.\mathbf{b} = ||\mathbf{a}||.||\mathbf{b}||.\cos((\hat{a,b}))$

[5] la preuve de ce théorème est donnée sous forme continue, pour les opérateurs, en Section 2.3.

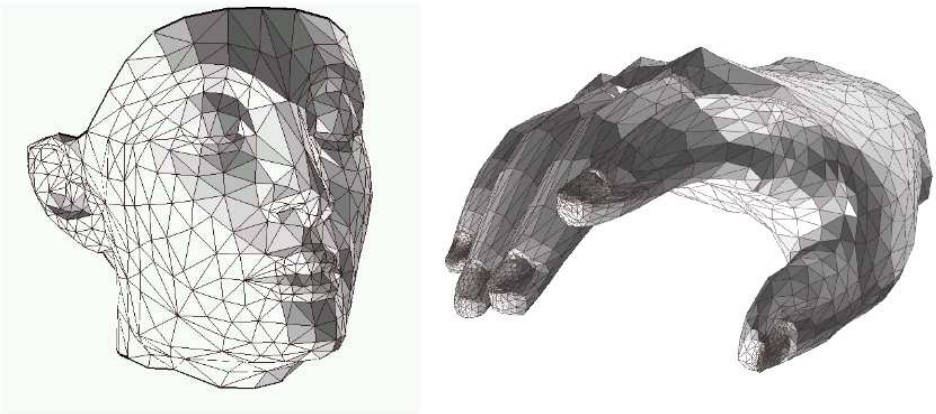


Figure 3.7: *Deux exemples de surfaces triangulées*

déduisons la relation entre les valeurs propres λ_1, λ_2 de G et les valeurs singulières σ_1, σ_2 de J, à savoir $\lambda_1 = \sigma_1^2$ et $\lambda_2 = \sigma_2^2$.

En résumé, un cercle élémentaire se transforme en une ellipse, dont les axes correspondent aux vecteurs propres du tenseur métrique G, et dont les longueurs des axes correspondent aux racines carrées des valeurs propres de G (ou encore aux valeurs singulières de la matrice Jacobienne J).

3.2 Surfaces triangulées

3.2.1 Définition

Une surface triangulée est un ensemble V de sommets $\mathbf{p}_i, i = 1 \ldots n$, reliés par un ensemble F de triangles. Chaque triangle est défini par le triplet (i, j, k) dénotant les indices de ses sommets. Il est parfois également utile de définir l'ensemble E des arêtes (i, j) présentes dans le maillage. Ainsi, une surface triangulée est définie par le triplet (V, E, F) de sommets V, d'arêtes E et de triangles (ou facettes) F. La Figure 3.7 montre deux exemples de surfaces triangulées.

3.2.2 Paramétrisation d'une surface triangulée

Une idée naturelle pour définir une paramétrisation d'une surface triangulée consiste à utiliser des fonctions linéaire par morceaux (les morceaux correspondant aux triangles). Ainsi, il est possible de représenter la paramétrisation par la donnée des coordonnées (u_i, v_i) associées à chaque sommet (x_i, y_i, z_i). La Figure 3.8 montre l'aspect d'une surface dans l'espace 3D et dans l'espace paramétrique (u, v).

Note: la section précédente considérait l'étude d'une paramétrisation existante, alors que cette section considère le problème de la construction d'une paramétrisation pour une surface existante. Pour cette raison, contraitement aux conventions de la géométrie différentielles que nous avons utilisées dans la section précédente, il est plus naturel de placer l'espace 3D (connu) à gauche des schemas, et l'espace paramétrique (que l'on cherche à construire) à droite des schemas. Nous verrons d'autres conséquences plus profondes de cette inversion, expliquant la formulation et le comportement des méthodes classiques.

En un point (u, v) arbitraire du domaine paramétrique Ω, la paramétrisation X est donnée par:

$$X(u, v) = \lambda_1 \mathrm{p}_i + \lambda_2 \mathrm{p}_j + \lambda_3 \mathrm{p}_k$$

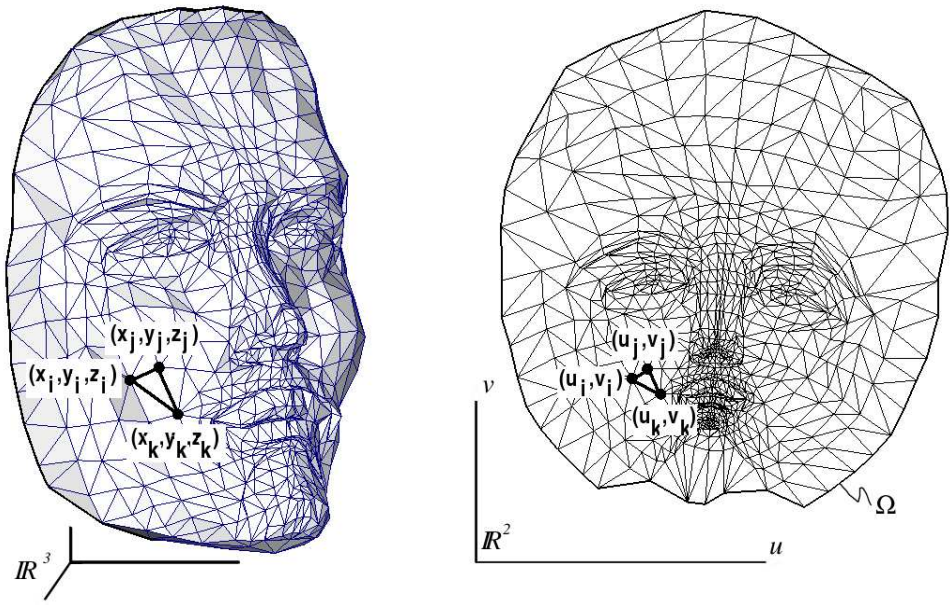


Figure 3.8: *Une paramétrisation d'une surface triangulée peut être définie sous la forme d'une fonction linéaire par morceaux, déterminée par les coordonnées* (u_i, v_i) *de chaque sommet* (x_i, y_i, z_i).

où (i, j, k) dénote le triplet d'indices tels que le triangle de l'espace paramétrique (u_i, v_i), (u_j, v_j), (u_k, v_k) contient (u, v), et $(\lambda_1, \lambda_2, \lambda_3)$ dénote les coordonnées barycentriques du point de coordonnées (u, v) dans ce triangle.

Les coordonnées barycentriques $(\lambda_1, \lambda_2, \lambda_3)$ sont solution du système suivant:

$$\begin{pmatrix} u_i & u_j & u_k \\ v_i & v_j & v_k \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_i \\ \lambda_j \\ \lambda_k \end{pmatrix} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Ce système peut se résoudre relativement facilement[6]:

$$\begin{pmatrix} \lambda_i \\ \lambda_j \\ \lambda_k \end{pmatrix} = \frac{1}{2|T|_{u,v}} \begin{pmatrix} v_j - v_k & u_k - u_j & u_j v_k - u_k v_j \\ v_k - v_i & u_i - u_k & u_k v_i - u_i v_k \\ v_i - v_j & u_j - u_i & u_i v_j - u_j v_i \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

où $2|T|_{u,v} = (u_i v_j - v_i u_j) + (u_j v_k - v_j u_k) + (u_k v_i - v_k u_i)$ dénote le double de l'aire du triangle (u_i, v_i), (u_j, v_j), (u_k, v_k) de l'espace paramétrique. Les cartes graphiques sont capables de calculer très rapidement ces équations[7], ce qui permet en "peignant" l'espace paramétrique avec une image de plaquer celle-ci sur l'objet 3D.

En résumé, construire une paramétrisation d'une surface triangulée consiste à chercher un ensemble de couples de coordonnées (u_i, v_i), associées à chaque sommet i. De plus, ces coordonnées doivent être telles que l'image de la surface dans l'espace paramétrique ne se recouvre pas. Nous verrons dans la suite de cette section plusieurs méthodes (issues de la littérature et résultant de nos travaux) pour calculer ces coordonnées.

[6] par exemple, en utilisant la formule de Cramer pour l'inverse d'une matrice, à savoir $M^{-1} = (1/|M|)Co(M)^t$, où $|M|$ dénote le déterminant de M, et $Co(M)$ la co-matrice de M.

[7] en procédent différement, les coordonnées barycentriques sont en général calculées incrémentalement lors de la visite successive de tous les pixels d'un triangle.

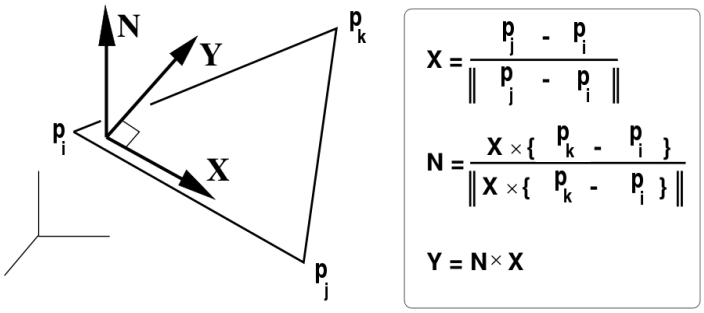


Figure 3.9: *Base locale X, Y dans un triangle.*

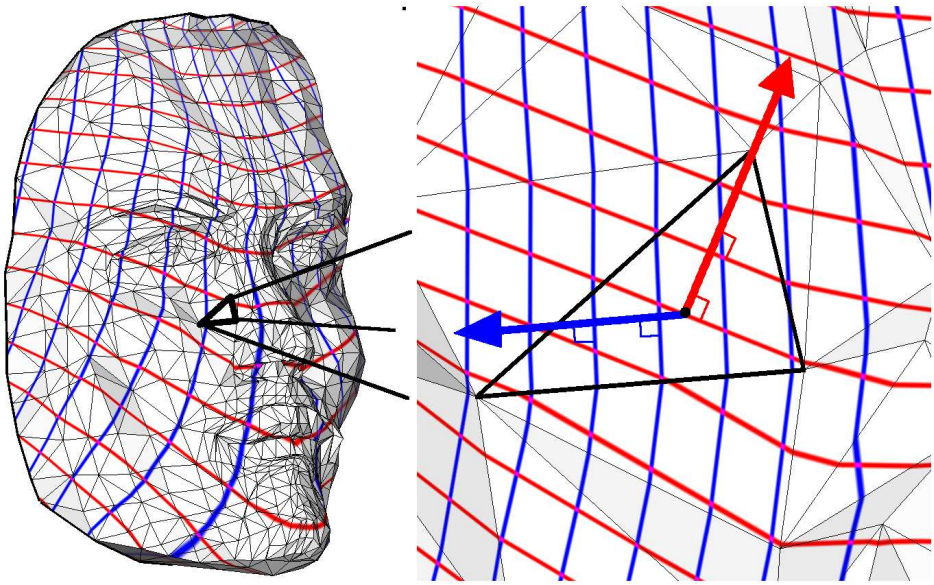


Figure 3.10: *Courbes iso-u,v et gradients associés.*

3.2.3 Gradient dans un triangle

L'analyse des déformations induites par la paramétrisation, présentée dans la section précédente, fait appel au calcul des gradients de la paramétrisation en fonction des deux paramètres u et v. Dans le cas d'une surface triangulée, la paramétrisation est une fonction linéaire par morceaux. Les gradients sont donc constants sur chaque triangle.

Avant d'aborder le calcul de ces gradients, il convient de préciser que le contexte est légèrement différent de celui de la section précédente. En effet, comme nous l'avons déjà évoqué, la surface 3D est donnée, et nous recherchons à construire une paramétrisation. Dans cette configuration, il est plus naturel de caractériser l'inverse de la paramétrisation, à savoir la fonction qui va de la surface 3D (connue) vers l'espace paramétrique (inconnu). Cette fonction est également linéaire par morceaux. Afin de donner un sens à l'analyse des déformations de cette fonction, il convient de munir chaque triangle d'une base orthonormée X,Y, explicitée sur la Figure 3.9 (et nous pouvons par exemple utiliser l'un des sommets $\mathbf{p}_i$ du triangle comme origine). Dans cette base, nous pouvons nous intéresser à l'inverse de la paramétrisation, à savoir la fonction qui met un point (X,Y) du triangle en correspondance avec un point (u,v) de l'espace paramétrique, et qui s'écrit:

$$\begin{cases} u(X,Y) &= \lambda_1 u_i + \lambda_2 u_j + \lambda_3 u_k \\ v(X,Y) &= \lambda_1 v_i + \lambda_2 v_j + \lambda_3 v_k \end{cases}$$

où $(\lambda_1,\lambda_2,\lambda_3)$ dénotent les coordonnées barycentriques du point (X,Y) dans le triangle, calculées comme précédemment:

$$\begin{pmatrix} \lambda_i \\ \lambda_j \\ \lambda_k \end{pmatrix} = \frac{1}{2|T|_{X,Y}} \begin{pmatrix} Y_j - Y_k & X_k - X_j & X_j Y_k - X_k Y_j \\ Y_k - Y_i & X_i - X_k & X_k Y_i - X_i Y_k \\ Y_i - Y_j & X_j - X_i & X_i Y_j - X_j Y_i \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

où $2|T|_{X,Y} = (X_i Y_j - Y_i X_j) + (X_j Y_k - Y_j X_k) + (X_k Y_i - Y_k X_i)$ dénote le double de l'aire du triangle (dans l'espace 3D cette fois).

En substituant la valeur de λ_1, λ_2 et λ_3 dans $u = \lambda_1 u_i + \lambda_2 u_j + \lambda_3 u_k$ (resp. v), nous obtenons:

$$\begin{pmatrix} \partial u/\partial X \\ \partial u/\partial Y \end{pmatrix} = \frac{1}{2|T|_{X,Y}} \begin{pmatrix} Y_j - Y_k & Y_k - Y_i & Y_i - Y_j \\ X_k - X_j & X_i - X_k & X_j - X_i \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix}$$

Comme nous le montrons sur la Figure 3.10, ces gradients sont différents (bien que fortement liés) aux gradients calculés dans la section précédente. Le gradient de u (resp. de v) coupe les iso-u (resp. les iso-v) à angle droit au lieu de lui être tangent, et sa norme est l'inverse de celle calculée dans la section précédente.

Dans le cas particulier où la base locale a son origine en $\mathbf{p}_i$ et l'axe des X le long de $\overrightarrow{\mathbf{p}_i\mathbf{p}_j}$ (c.f. Figure 3.9), des simplifications peuvent être réalisées:

$$\begin{array}{lcl} X_i = 0 & ; & Y_i = 0 \\ X_j = d_{i,j} & ; & Y_j = 0 \\ 2|T| = X_j \times Y_k & & \end{array}$$

On obtient ainsi l'expression suivante du gradient:

$$\begin{pmatrix} \partial u/\partial X \\ \partial u/\partial Y \end{pmatrix} = \begin{pmatrix} -1/d_{i,j} & 1/d_{i,j} & 0 \\ X_k/(d_{i,j} \times Y_k) - 1/Y_k & -X_k/(d_{i,j} \times Y_k) & 1/Y_k \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \tag{3.8}$$

avec $d_{i,j} = \|\overrightarrow{\mathbf{p_i}\mathbf{p_j}}\|$.

Avant de voir comment utiliser l'expression de ces gradients pour minimiser les déformations, nous allons voir une méthode classique de paramétrisation.

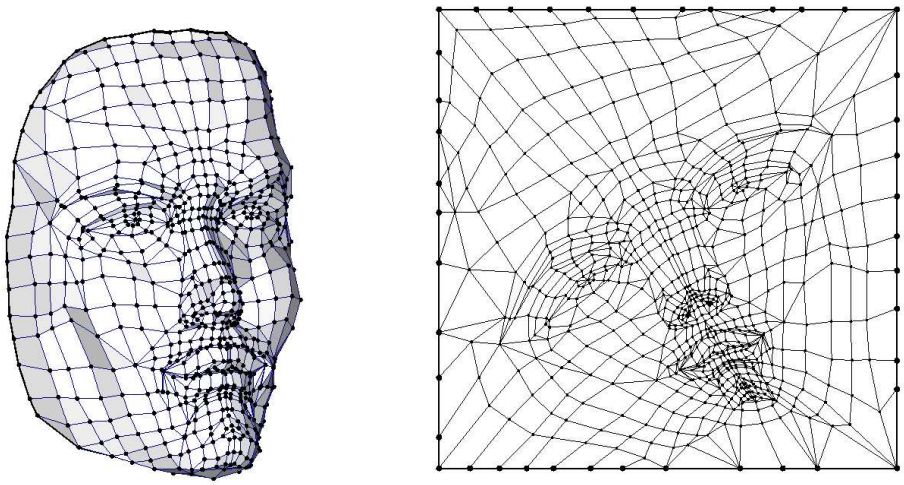


Figure 3.11: *Paramétrisation par la méthode de Tutte-Floater. Les coordonnées paramétriques du bord de la surface sont fixées sur un polygone convexe, et les autres sont obtenues en résolvant un système linéaire.*

3.2.4 Cartes barycentriques

Les cartes barycentriques représentent la méthode la plus répandue pour construire une paramétrisation d'une surface triangulée. Cette méthode se fonde sur le théorème de Tutte [Tut60], emprunté à la théorie des graphes, qui s'énonce de la manière suivante:

Etant donnée une surface triangulée homéomorphe à un disque, si les coordonnées des noeuds du bord forment un polygone convexe, et si les coordonnées des noeuds internes forment une combinaison barycentrique de leurs voisins, alors les coordonnées définissent une paramétrisation valide (à savoir sans recouvrement). La deuxième condition se formalise ainsi:

$$\forall i, -a_{i,i}\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \sum_{j \in N_i} a_{i,j}\begin{pmatrix} u_j \\ v_j \end{pmatrix}$$

où N_i dénote l'ensemble des noeuds connectés au noeud i par une arête, et où les coefficients $a_{i,j}$ satisfont:

$$\forall i \text{ noeud interne }, \begin{cases} a_{i,j} > 0 \text{ si } i \neq j \\ a_{i,i} = -\sum_{j \neq i} a_{i,j} \end{cases} \tag{3.9}$$

Michael Floater [FH04] a eu l'idée d'utiliser ce théorème – qui *caractérise* une famille de paramétrisations valides – comme une méthode pour *construire* une paramétrisation. L'idée consiste à tout d'abord fixer les noeuds du bord sur un polygône convexe, puis à résoudre l'équation 3.9 afin de trouver les coordonnées des noeuds internes. Ceci revient à résoudre deux systèmes linéaires $A\mathbf{u} = \mathbf{u}_0$ et $A\mathbf{v} = \mathbf{v}_0$, où les vecteurs $\mathbf{u}$ et $\mathbf{v}$ contiennent toutes les coordonnées u (resp. v) des noeuds internes, et où le membre de droite $\mathbf{u}_0$ (resp. $\mathbf{v}_0$) contient les coordonnées des noeuds du bord. Ces deux systèmes linéaires peuvent être résolus par l'une des méthodes présentée dans la section 2.2. La figure 3.11 montre un exemple de paramétrisation calculée par cette méthode (et des poids $a_{i,j}$ explicités plus loin).

[Tut60] W. Tutte. Convex representation of graphs. In *Proc. London Math. Soc.*, volume 10, 1960.

[FH04] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In M. S. Floater N. Dodgson and M. Sabin, editors, *Advances on Multiresolution in Geometric Modelling*. Springer-Verlag, 2004.

La preuve initiale du théorème de Tutte fait appel à des outils sophistiqués de théorie des graphes [Tut60]. Plus récemment, une preuve plus simple à été établie par Colin de Verdière[dV90]. Enfin, une preuve fondée sur la notion de forme différentielle discrète a également été construite[GGT05]. Se fondant sur de simples arguments de comptage d'éléments, cette dernière preuve a la propriété intéressante d'être compréhensible sans nécessiter le bagage important de théorie des graphes utilisé par les deux précédentes.

Un choix possible pour les coefficients $a_{i,j}$ est donné par $a_{i,j} = 1$ si $i \neq j$ et $a_{i,i} = -|N_i|$. Toutefois, ce choix introduit des déformations, non souhaitable dans la plupart des applications. Pour cette raison, la sous-section suivante introduit une méthode de calcul des coefficients $a_{i,j}$, permettant de minimiser les déformations.

3.2.5 Laplacien discret

Le Laplacien, ou opérateur de Laplace, est une généralisation de la dérivée seconde à des fonctions de plusieurs variables. En deux dimensions, cette opérateur s'écrit:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Intuitivement, le Laplacien définit la régularité (ou plutôt l'irrégularité) d'une fonction. Par exemple, les fonctions linéaires annulent le Laplacien. Nous nous intéressons donc à cet opérateur avec l'idée qu'il va nous aider à choisir les coefficients $a_{i,j}$ de l'équation de Tutte, de manière à optimiser la régularité de la paramétrisation ainsi construite.

Pour calculer ces coefficients, il est possible de considérer l'équation de Tutte comme un système de deux équations de Laplace avec conditions aux limites de Dirichlet, à savoir:

$$\begin{array}{lcl} \Delta u = 0 \text{ sur } \mathbf{S} & ; & u = u_0 \text{ sur } \partial\mathbf{S} \\ \Delta v = 0 \text{ sur } \mathbf{S} & ; & v = v_0 \text{ sur } \partial\mathbf{S} \end{array} \tag{3.10}$$

Plusieurs méthodes permettent de discrétiser cette équation. La plupart de ces méthodes se fondent sur l'identité bien connue:

$$\int_S \Delta f g = -\int_S \nabla f \nabla g + \int_{\partial S} g \nabla f N \tag{3.11}$$

où N dénote la normale au bord du domaine d'intégration. Cette équation s'obtient par intégration par parties. Remarquons que le deuxième terme disparait quand la surface est fermée.

Ainsi, en utilisant l'équation des gradients vue dans la section précédente, Pinkall et Polthier [PP93a] ont obtenu une discrétisation du Laplacien. Il est également possible de procéder en calculant des grandeurs différentielles, et en les intégrant sur de petits voisinnages [DMSB99], ce qui permet d'obtenir la même formule que Pinkall et Polthier. Dans le cadre de ce mémoire, nous préférons utiliser la méthode des éléments finis, présentée brièvement en section 2.3.3, car elle permet de mieux justifier la présence du terme de normalisation par l'aire, comme nous le verrons par la suite.

[dV90] Y. Colin de Verdiere. Sur un nouvel invariant des graphes et un critere de planarite. *J. of Combinatorial Theory*, 50, 1990.

[GGT05] S. J. Gortler, C. Gotsman, and D. Thurston. One-forms on meshes and applications to 3d mesh parameterization. *Journal of CAGD*, 2005.

[PP93a] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Math.*, 2(15), 1993.

[DMSB99] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH Conference Proceedings*, pages 317–324. ACM, 1999.

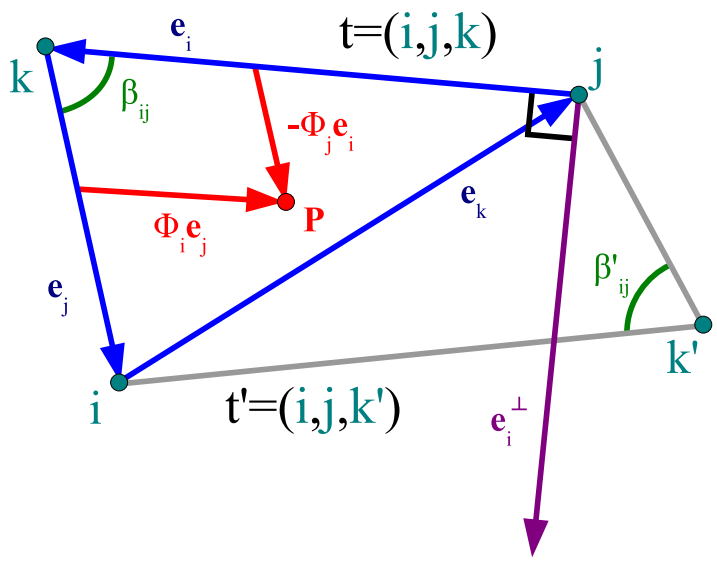


Figure 3.12: *Notations pour le calcul des coefficients de B et de Q.*

Il existe plusieurs manières d'aborder la discrétisation de l'équation 3.10, nous pouvons directement appliquer la méthode des éléments finis décrite en Section 2.3.3, ou alors tenter de définir un *Laplacien discret*, sous forme d'une matrice. Nous détaillons ce deuxième point de vue. Nous considérons une base de fonction de tests (Φ_i), par exemple les fonctions P1. Etant donnée une fonction $g = \sum g_i \Phi_i$, définir un Laplacien discret revient à chercher une fonction $f = \sum f_i \Phi_i$ telle que:

$$f = \Delta g$$

En projetant cette équation sur la base des fonctions de tests (Φ_i), nous obtenons:

$$\forall i, < f, \Phi_i > = < \Delta g, \Phi_i >$$

soit:

$$\forall i, \sum_j f_j < \Phi_j, \Phi_i > = \sum_j g_j < \Delta \Phi_j, \Phi_i >$$

ou encore, en appliquant la formule de Stokes (Equation 3.11):

$$\forall i, \sum_j f_j < \Phi_j, \Phi_i > = -\sum_j g_j < \nabla \Phi_j, \nabla \Phi_i >$$

Sous forme matricielle, cette équation s'écrit:

$$B \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \quad = \quad Q \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix}$$

avec:

$$B_{i,j} = < \Phi_i, \Phi_j > \quad ; \quad Q_{i,j} = - < \nabla \Phi_i, \nabla \Phi_j >$$

Nous détaillons à présent le calcul des coefficients de la matrice de rigidité Q et de la matrice de masse B. Pour ce faire, nous commençons par paramétrer le triangle $t = (i, j, k)$ par les coordonnées barycentriques de chaque point $P \in t$ relatives aux sommets i et j. Nous remarquons que ces coordonnées correspondent aux fonctions "chapeau" P1 Φ_i et Φ_j. Cette remarque nous permet d'écrire $P = k + \Phi_i \mathbf{e}_j - \Phi_j \mathbf{e}_i$ avec Φ_i, $\Phi_j \geq 0$ et $\Phi_i + \Phi_j \leq 1$

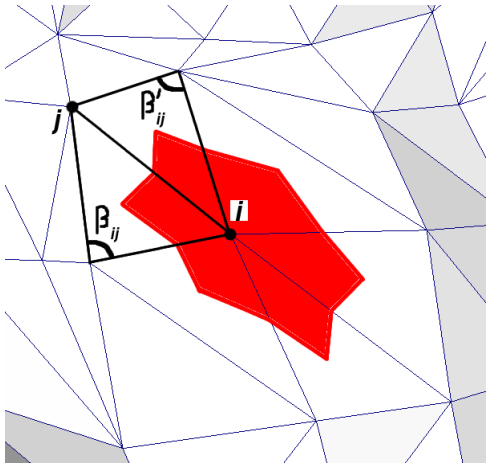


Figure 3.13: *Coefficients du Laplacien discret.*

(voir Figure 3.12). Ceci définit un élément différentiel d'aire $dA(P) = \mathbf{e}_i \wedge \mathbf{e}_j d\Phi_i d\Phi_j = 2|t| d\Phi_i d\Phi_j$, où $|t|$ correspond à l'aire du triangle t. Ainsi, nous obtenons l'intégrale:

$$\int_{P\in t} \Phi_i \Phi_j dA = 2|t| \int_{\Phi_i=0}^{1} \int_{\Phi_j=0}^{1-\Phi_i} \Phi_i \Phi_j d\Phi_i d\Phi_j =$$

$$|t| \int_{\Phi_i=0}^{1} \Phi_i (1-\Phi_i)^2 d\Phi_i = |t| \left(\frac{1}{2} - \frac{2}{3} + \frac{1}{4} \right) = \frac{|t|}{12}$$

que nous sommons sur les deux triangles qui partagent l'arête (i, j), de manière à calculer le coefficient de la matrice de masse $B_{i,j} = (|t| + |t'|/12$.

Les termes diagonaux sont donnés par:

$$\int_{P\in t} \Phi_i^2 dA = 2|t| \int_{\Phi_i=0}^{1} \int_{\Phi_j=0}^{1-\Phi_i} \Phi_i^2 d\Phi_i d\Phi_j =$$

$$2|t| \int_{\Phi_i=0}^{1} \Phi_i^2 (1-\Phi_i) d\Phi_i = 2|t| \left(\frac{1}{3} - \frac{1}{4} \right) = \frac{|t|}{6}$$

que nous sommons sur l'ensemble $St(i)$ des triangles incidents au sommet i. Ainsi, nous obtenons: $B_{i,i} = (\sum_{t\in St(i)} |t|)/6$.

Pour calculer les coefficients de la matrice de rigidité Q, nous utilisons la formule de Stokes et l'intégration par parties (Equation 3.11), ce qui permet de réduire l'ordre des dérivations et fournit une formule plus symétrique:

$$Q_{i,j} =< \Delta\Phi^i, \Phi^j >=< \delta d\Phi^i, \Phi^j >=< d\Phi^i, d\Phi^j >= -\int_S \nabla\Phi^i . \nabla\Phi^j$$

Dans t, les gradients des coordonnées barycentriques sont des vecteurs constants:

$$\nabla\Phi^i = \frac{-\mathbf{e}_i^{\perp}}{2|t|} \qquad \nabla\Phi^i . \nabla\Phi^j = \frac{\mathbf{e}_i . \mathbf{e}_j}{4|t|^2}$$

où $\mathbf{e}_i^{\perp}$ dénote $\mathbf{e}_i$ tourné de $\pi/2$ autour de la normale au triangle t.

En intégrant ce terme sur t, nous obtenons:

$$\int_t \nabla\Phi^i . \nabla\Phi^j dA = \frac{\mathbf{e}_i . \mathbf{e}_j}{4|t|} = \frac{||\mathbf{e}_i||.||\mathbf{e}_j|| \cos(\beta_{ij})}{2||\mathbf{e}_i||.||\mathbf{e}_j|| \sin(\beta_{ij})} = \frac{\cot(\beta_{ij})}{2}$$

En sommant ces expressions, nous obtenons finalement les coefficients de la matrice de rigidité Q:

$$Q_{i,i} = \sum_{t \in St(i)} \nabla\Phi^i.\nabla\Phi^i = \sum_{t \in St(i)} \frac{\mathbf{e}_i^2}{4|t|}$$

$$Q_{i,j} = \int_{t \cup t'} \nabla\Phi^i.\nabla\Phi^j = \frac{1}{2}\left(\cot(\beta_{ij}) + \cot(\beta'_{ij})\right)$$

En utilisant ces coefficients, et en rappellant le lien entre $f = \Delta g$ et les matrices B et Q:

$$B \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \quad = \quad Q \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix}$$

nous pouvons définir le Laplacien discret $\Delta^d = B^{-1}Q$. En effet, cette matrice permet à partir d'une fonction g exprimée dans la base des fonctions de test Φ_i de calculer une fonction f égale au Laplacien de g en projection sur les fonctions Φ_i:

$$f = \Delta g \quad \rightarrow \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \Delta^d \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix}$$

Toutefois, il convient de noter que la matrice B n'est pas triviale à inverser (son schéma de valeurs non-nulles correspond à la connectivité du graphe). Pour cette raison, la communauté des éléments finis utilise communément une approximation appellée *masse condensée*[8], qui consiste à remplacer la matrice B par une matrice diagonale D (donc bien plus facile à inverser), définie par:

$$D_{i,i} = \sum_j B_{i,j}$$

Le Laplacien discret est alors une matrice dont les coefficients sont donnés par:

$$\Delta^d = D^{-1}Q \quad ; \quad \Delta^d_{i,j} = \frac{3(\text{cotan}\beta_{i,j} + \text{cotan}\beta'_{i,j})}{2 \sum\limits_{t \in St(i)} |t|} \quad ; \quad \Delta^d_{i,i} = -\sum_{j \neq i} \Delta^d_{i,j}$$

La Figure 3.13 rappelle les notations utilisées pour calculer le coefficient $\Delta^d_{i,j}$. L'aire indiquée en rouge correspond au coefficient de la matrice de masse condensée D (à savoir le tiers de l'aire des triangles incidents au sommet i). Nous rappelons également la notation utilisée pour les deux angles $\beta_{i,j}$ et $\beta'_{i,j}$.

Comme nous pouvons le remarquer, nous retrouvons la même formule que Pinkall *et. al* [PP93a] ou Meyer *et. al* [DMSB99], avec une meilleure justification de la présence du terme de correction d'aire, qui provient dans notre cas de la matrice de masse condensée.

Nous pouvons donc à présent utiliser notre Laplacien discret pour définir les coefficients $a_{i,j}$ utilisés dans la méthode de Tutte-Floater. Toutefois, il convient de préciser que pour certains maillages présentant des angles suffisament obtus, les coefficients du Laplacien discret peuvent devenir négatif. En conséquence, les conditions du théorème de Tutte ne sont plus satisfaites, et nous pouvons effectivement observer des recouvrement locaux dans la solution. Floater a plus récemment mis au point un système de pondération (*Mean Value Coordinates*)[Flo03] qui ne soufre pas de ce problème (les coefficients sont toujours

[8] en mécanique, les coefficients de la matrice B correspondent souvent à la masse du système considéré. L'approximation consiste à "condenser" toute la masse sur les termes diagonaux de la matrice

[PP93a] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Math.*, 2(15), 1993.

[DMSB99] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH Conference Proceedings*, pages 317–324. ACM, 1999.

[Flo03] M. S. Floater. Mean value coordinates. *CAGD*, 20:19–27, 2003.

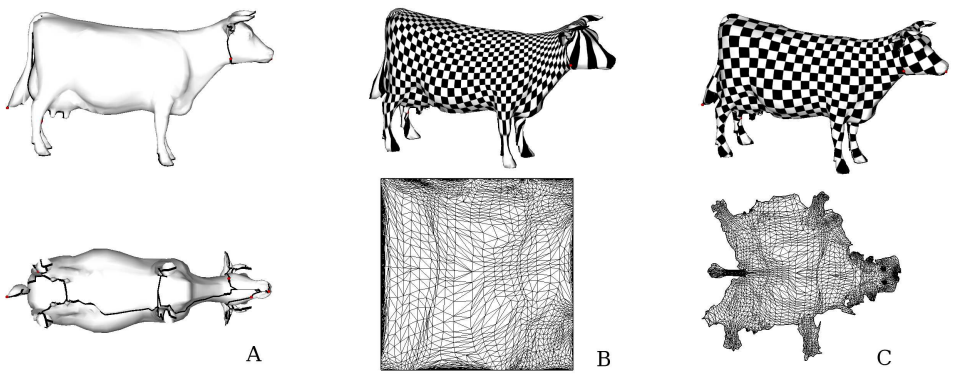


Figure 3.14: *A: un maillage découpé, rendu homéomorphe à un disque grâce à la méthode* seamster[SH02]*; B: paramétrisation Tutte-Floater obtenue en fixant le bord sur un carré; C: paramétrisation obtenue par une méthode à bord libre*[SdS01].

positifs), et qui donne un résultat très proche de celui obtenu avec un Laplacien discret. Toutefois, étant fondée sur le théorème de Tutte, cette dernière méthode requiert toujours de fixer le bord de la surface sur un polygone convexe (c.f. Figure 3.14). Ceci présente deux inconvénient: (1) il est en général assez difficile de trouver des coordonnées "naturelles" pour le bord, et (2) pour certaines surfaces, la contrainte de fixer le bord sur un polygone *convexe* induit une paramétrisation peu naturelle, avec de fortes déformations, si par exemple notre objet de départ a lui-même une forme fortement non-convexe. Même si dans l'exemple montré sur la figure il serait sans doute possible de trouver une meilleure forme pour le bord, engendrant moins de déformations, la paramétrisation ainsi obtenue sera probablement moins bonne que le résultat de la Figure 3.14-C, plus conforme à ce qu'un taneur attendrait. Pour cette raison, nous nous sommes intéressé à des méthodes permettant de libérer le bord de la surface, en définissant des fonctionelles d'énergies ayant un meilleur comportement pour l'extrapolation. Nous détaillons ces méthodes dans la section suivante.

3.3 Libérer le bord

Dans la deuxième moitié des années 90, la méthode de Floater[Flo97], fondée sur le théorème de Tutte[Tut60] était bien connue de la communauté, et appliquée à une grande classe de problèmes. Toutefois, la nécessité de contraindre le bord sur un polygone convexe limitait l'efficacité de certaines applications. Pour cette raison, la communauté a cherché à mettre en place des méthodes de soufrant pas de cette limitation, tout en continuant de chercher à minimiser les déformations. Cette section établit tout d'abord un bref état de l'art de ces méthodes, en les exprimant dans le formalisme de l'analyse des déformations, mis en place dans la Section 3.1.2. Toutefois, avant d'aller plus loin, une précaution s'impose:

- ◇ la moitié de ces méthodes étudie la fonction allant de la surface vers le domaine paramétrique (comme dans la Section précédente). En effet, les coordonnées (u,v) étant inconnues, il est plus naturel d'étudier la fonction allant de l'espace connu (la surface) vers l'espace inconnu (l'espace paramétrique);
- ◇ l'autre moitié la fonction inverse, allant du domaine paramétrique vers la surface (comme dans notre Section 3.1.2). Ce choix se justifie car il permet de mieux se mettre en re-

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

[Tut60] W. Tutte. Convex representation of graphs. In *Proc. London Math. Soc.*, volume 10, 1960.

lation avec les ouvrages classiques de géométrie différentielle[do 76] qui utilisent cette convention.

Comme les deux conventions se justifient, différents auteurs utilisent soit l'une, soit l'autre. Afin de marquer la différence, dans ce chapitre, nous utiliserons les notations suivantes:

Notations pour la convention $(x,y) \mapsto (u,v)$

Les symboles J, G, σ_1, σ_2 se rapportent à la fonction allant de la surface vers l'espace paramétrique, et dénotent respectivement la matrice Jacobienne, la première forme fondamentale, et les longueurs des deux axes de l'ellipse d'anisotropie. Dans le cas d'un triangle T, dont les coordonnées des sommets exprimées dans une base orthonormée de son plan support sont notées (x_1,y_1), (x_2,y_2), (x_3,y_3) et dont les coordonnées paramétriques sont notées (u_1,v_1), (u_2,v_2), (u_3,v_3), ces grandeurs s'expriment de la manière suivante:

$$\begin{aligned} J_T &= \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{pmatrix} \\ &= \frac{1}{2|T|_{x,y}} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_3-x_2 & x_1-x_3 & x_2-x_1 \\ y_3-y_2 & y_1-y_3 & y_2-y_1 \end{pmatrix} \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \end{pmatrix} \\ &= \frac{1}{2|T|_{x,y}} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \end{pmatrix} \end{aligned} \tag{3.12}$$

$$G_T = J_T^t J_T = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$\text{avec:} \begin{cases} a &= \frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2 \\ b &= \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial y} \\ c &= \frac{\partial v}{\partial x}^2 + \frac{\partial v}{\partial y}^2 \end{cases} \tag{3.13}$$

Comme nous l'avons vu dans la Section 3.1.2, les longueurs des axes de l'ellipse d'anisotropie σ_1 et σ_2 correspondent aux valeurs singulières de J_T, ou aux racines carrées des valeurs propres de G_T. Leur expression s'obtient en calculant les racines carrées des racines du polynôme caractéristique $|G_T - \sigma Id|$:

$$\begin{aligned} \sigma_1 &= \sqrt{1/2(a+c) + \sqrt{(a-c)^2 + 4b^2}} \\ \sigma_2 &= \sqrt{1/2(a+c) - \sqrt{(a-c)^2 + 4b^2}} \end{aligned} \tag{3.14}$$

Notations pour la convention $(u,v) \mapsto (x,y)$

Les symboles $J', G', \sigma_1', \sigma_2'$ se rapportent à la fonction allant de l'espace paramétrique vers la surface, et dénotent respectivement la matrice Jacobienne, la première forme fondamentale, et les longueurs des deux axes de l'ellipse d'anisotropie.

[do 76] M.F. do Carmo. *Differential geometry of curves and surfaces*. Prentice Hall, Englewood Cliffs, Inc., 1976.

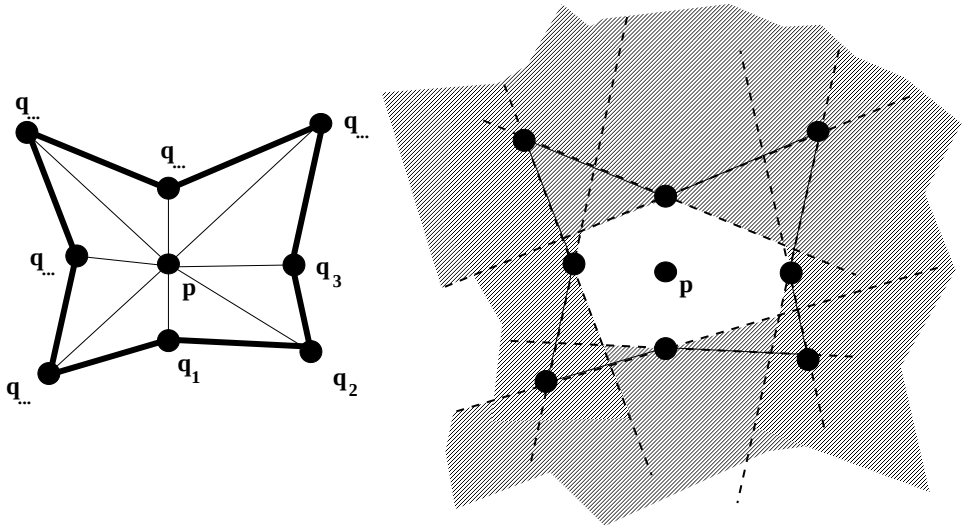


Figure 3.15: *A gauche: pour éviter les retournements de triangles, chaque sommet* **p** *est contraint à rester dans le noyau du polygone défini par ses voisins* $\mathbf{q}_i$*; A droite: le noyau d'un polygone (en blanc) est défini comme l'intersection des demi-plans définis par les droites supports de ses côtés (en grisé).*

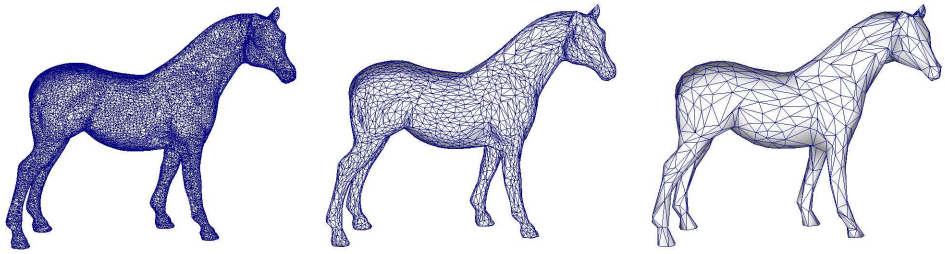


Figure 3.16: *Les méthodes de paramétrisation fondées sur l'analyse des déformations utilisent des fonctions objectif souvent non-linéaires. Pour accéler les calculs, ces méthodes utilisent en général une approche multi-résolution, fondée sur la structure de maillage progressif.*

Nous étudions les relations entre ces grandeurs et celles associées à la fonction inverse. La matrice Jacobienne de l'inverse d'une fonction f est égale à l'inverse de la matrice Jacobienne de f, d'où $J' = J^{-1}$. De plus, il est facile de vérifier que si la SVD de J s'écrit $U\Sigma V^t$, avec $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2)$ et U, V des matrices unitaires, à savoir telles que $U^tU = Id$ et $V^tV = Id$, alors la SVD de J' s'écrit $V\Sigma^{-1}U^t$ (le produit des deux donne bien la matrice identité). Les longueurs du plus petit et du plus grand axe de l'ellipse d'anisotropie de la fonction inverse sont donc donnés par:

$$\sigma_1' = \frac{1}{\sigma_2} \quad ; \quad \sigma_2' = \frac{1}{\sigma_1}$$

3.3.1 Méthodes fondées sur l'analyse des déformations

Ces quelques définitions nous permettent à présent de considérer plusieurs méthodes de la littérature et de les exprimer dans un formalisme commun, en prenant la précaution de bien examiner quel est l'espace de départ et l'espace d'arrivée, à savoir la surface ou l'espace paramétrique. Avant d'évoquer ces méthodes, nous précisons deux aspects importants:

- ◇ afin d'éviter les retournements de triangles, plusieurs de ces méthodes contraignent chaque sommets $\mathbf{p}$ à rester dans le noyau du polygône défini par ses voisins $\mathbf{q}_i$. Cette notion est illustrée sur la Figure 3.15. Pour calculer le noyau d'un polygône, il est par exemple possible d'appliquer l'algorithme de fenêtrage (clipping) ré-entrant de Sutherland et Hogdman au polygône (fenétré par lui-même). Cet algorithme est décrit dans tous les bons ouvrages généraux sur le graphisme[FvFH90];
- ◇ comme elles mettent en jeu les valeurs propres de la première forme fondamentale, les fonctions objectifs issues de l'analyse des déformations sont souvent non-linéaires, et donc difficile à minimiser efficacement. Pour accélerer les calculs, une technique courante consiste à utiliser une approche multi-résolution, fondée sur la structure de donnée appelée *maillage progressif*, introduite par Hoppe[Hop96]. L'algorithme optimise tout d'abord une version simplifiée de l'objet, puis introduit les sommets supplémentaire et les optimise par raffinements successifs.

Maintenant que nous avons vu les notions générales liées à l'analyse des déformations ainsi que les aspects particuliers liés à l'optimisation des fonctionelles d'énergie issue de cette analyse des déformations, nous pouvons aborder plusieurs méthodes classiques appartenant à cette famille.

Tenseur de Green Lagrange

Historiquement, pour minimiser les déformations d'une paramétrisation, l'une des premières méthodes a été développée par Maillot, Yahia et Verroust[MYV93]. L'idée de cette méthode consiste à minimiser une norme du tenseur des déformations de Green-Lagrange. Cette notion est définie en mécanique des milieux continus pour mesurer les déformations d'un matériau. Intuitivement, nous savons que si le tenseur métrique G est égal à la matrice unité, alors nous sommes en présence d'une paramétrisation isometrique. Le tenseur des déformations de Green-Lagrange est égal à $G - Id$, et mesure donc le caractère non-isométrique de la paramétrisation. L'approche de Maillot *et.al* consiste ainsi à minimiser une norme matricielle de $G - Id$.

Paramétrisation Quasi-Isométrique (MIPS)

La méthode MIPS de Hormann et Greiner[HG00] (Mostly Isometric Parameterization of Surfaces, ou paramétrisation de surfaces quasi-isométrique) a été – à notre connaissance – la première méthode de paramétrisation de surfaces permettant de laisser le bord évoluer naturellement. Cette méthode se fonde sur la minimisation du rapport entre les longueurs des axes de l'ellipse d'anisotropie, correspondant à la 2-norme de la matrice Jacobienne:

$$K_2(J_T) = \|J_T\|_2 \|J_T^{-1}\|_2 = \sigma_1/\sigma_2$$

Comme la minimisation de cette énergie ne se comporte pas bien numériquement, Hormann et Greiner ont remplacé la 2-norme $\|.\|_2$ par la norme de Froebenius $\|.\|_F$, à savoir la

[FvFH90] Foley, vanDam, Feiner, and Hughes. *Computer Graphics, principles and practice*. Addisson Welsey, 1990.

[Hop96] H. Hoppe. Progressive meshes. In *SIGGRAPH Conf. Proc.*, pages 99–108. ACM, 1996.

[MYV93] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *SIGGRAPH 93 Conf. Proc.*, pages 27–34. Addison Wesley, 1993.

[HG00] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In P.-J. Laurent, P. Sablonniere, and L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.

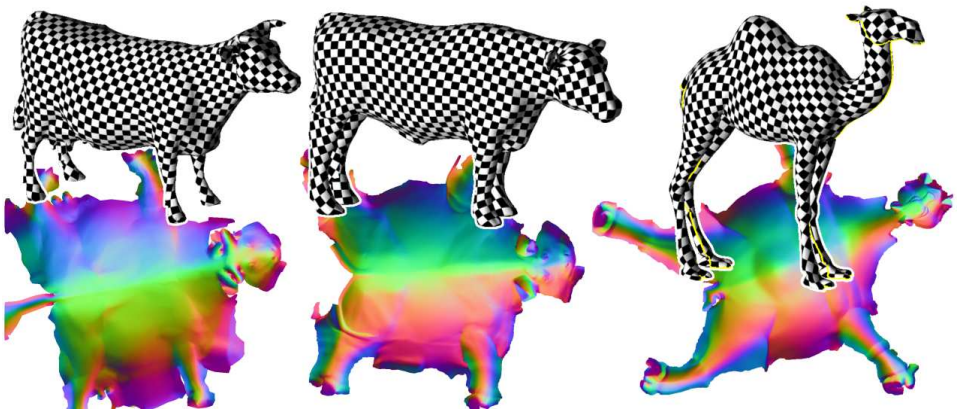


Figure 3.17: *Quelques résultats calculés par minimisation de la dérivée directionelle (stretch L_2), données fournies par Pedro Sander.*

racine carrée de la some des carrés des valeurs singulières:

$$\begin{aligned} K_F(J_T) &= \|J_T\|_F \|J_T^{-1}\|_F \\ &= \sqrt{\sigma_1^2+\sigma_2^2}\sqrt{\left(\frac{1}{\sigma_1}\right)^2+\left(\frac{1}{\sigma_2}\right)^2} \\ &= \frac{\sigma_1^2+\sigma_2^2}{\sigma_1\sigma_2} \\ &= \frac{\text{trace}(G_T)}{\det(J_T)} \end{aligned}$$

Comme nous pouvons le constanter, les calculs se simplifient relativement bien, pour aboutir au rapport entre la trace du tenseur métrique et le déterminant de la matrice Jacobienne. Cette grandeur peut s'interpreter comme l'*énergie de Dirichlet par unité d'aire dans l'espace paramétrique*: le terme trace(G) correspond à l'énergie de Dirichlet, et le Jacobien $\det(J)$ au demi-rapport des aires des triangles dans l'espace 3D et dans l'espace paramétrique.

Minimisation de la dérivée directionelle (Stretch L_2)

Motivés par des applications de plaquage de textures, Sander *et. al*[SSGH01] ont étudié la manière dont un signal stocké dans l'espace paramétrique se déforme lorsqu'il est projeté sur la surface (en lui appliquant la paramétrisation). Pour cette raison, leur formalisme utilise la fonction inverse, qui va de l'espace paramétrique vers la surface (et nous utiliserons donc les notations J', G', σ_1', σ_2', comme nous l'avons expliqué au début de la section).

Une manière possible de caractériser les déformations d'une texture est de considérer en un point de l'espace paramétrique et pour une direction donnée la manière dont la texture ce déforme. Sander *et. al* ont nommé cette grandeur l'*élongation* ("stretch" en Anglais). Ceci correspond précisément à la notion de dérivée directionelle que nous avons abordée dans la Section 3.1.2. Pour un triangle T, ils définissent alors deux énergie, correspondant à la moyenne de l'élongation pour toutes les directions (stretch $L_2(T)$), et à l'élongation

[SSGH01] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Computer Graphics (SIGGRAPH)*, pages 409–416. ACM, 2001.

maximale (stretch $L_\infty(T)$) :

$$L_2(T) = \sqrt{(\sigma_1'^2 + \sigma_2'^2)/2}$$

$$L_\infty(T) = \sigma_2'$$

où l'expression de $\sigma_1' = 1/\sigma_2$ et $\sigma_2' = 1/\sigma_1$ est donnée en début de section. Ces énergies locales de chaque triangle T se combinent alors en énergies globales $L_2(S)$ et $L_\infty(S)$ sur une surface S de la manière suivante:

$$L_2(S) = \sqrt{\frac{\sum_T |T| L_2(T)}{\sum_T |T|}}$$

$$L_\infty(S) = \mathrm{Max}_T L_\infty(T)$$

La Figure 3.17 montre quelques résultats calculés avec cette méthode. Ce formalisme est particulièrement bien adapté au plaquage de texture, car il mesure précisément les déformations responsables des artéfacts visuels que ce type d'application cherche à éviter. De plus, une modification simple de cette méthode permet de prendre en compte le contenu de la texture, et ainsi de calculer une paramétrisation adaptée au signal[SGSH02].

Energie combinée

Afin d'introduire plus de souplesse dans ces méthodes, Degener *et. al* ont proposé une énergie combinée[DMK03], avec un terme $\sigma_1'\sigma_2'$ pénalisant les déformations d'aires[9], et un terme σ_1'/σ_2' pénalisant les déformations angulaire. Pour faciliter l'optimisation numérique, chaque terme x est remplacé par l'expression $x + 1/x$, ce qui donne:

$$E_{combined} = E_{angle} \times (E_{area})^\theta$$

avec:

$$E_{area} = \sigma_1'\sigma_2' + \frac{1}{\sigma_1'\sigma_2'} = \det(J_T') + \frac{1}{\det(J_T')}$$

$$E_{angle} = \frac{\sigma_1'}{\sigma_2'} + \frac{\sigma_2'}{\sigma_1'} = \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1}{\sigma_2} = E_{MIPS}$$

où le paramètre θ permet de choisir l'importance relative du critère d'angles et du critère d'aires. Nous remarquons que le critère d'angles correspond exactement à l'énergie minimisée par la méthode MIPS, abordée au début de la Section.

3.3.2 Plaquage de textures sous contraintes

Maintenant que nous avons vu l'analyse des déformations, son expression dans le cas des surfaces triangulées, et l'application de ces concepts dans l'état de l'art, je présente plusieurs de mes contributions, fondées sur les même concepts.

Pour "libérer le bord", ma première idée a été de définir une fonctionelle d'énergie minimisant les variations du gradient, avec pour intuition que ceci permettrait de définir des contraintes et d'"extrapoler" la paramétrisation au delà de ces contraintes. En effet,

[9] ceci correspond également au Jacobien de la paramétrisation, à savoir le déterminant de la matrice Jacobienne, qui définit l'élément différentiel d'aires

[SGSH02] P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Eurographics Workshop on Rendering*, 2002.

[DMK03] P. Degener, J. Meseth, and R. Klein. An adaptable parameterization method. 2003.

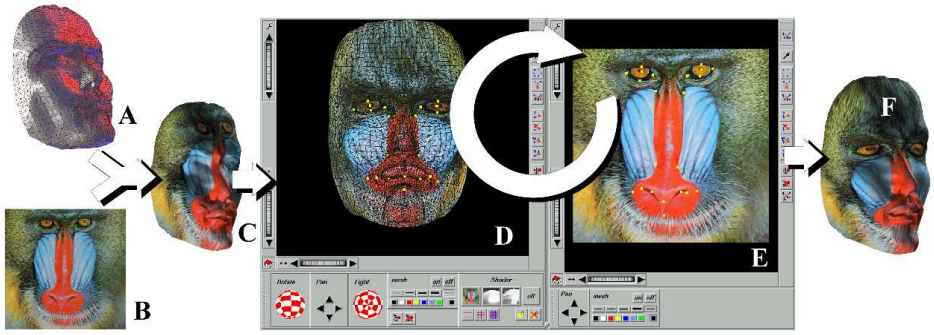


Figure 3.18: *En partant d'un modèle 3D (A) et d'une image (B), notre système plaque l'image sur le modèle automatiquement (C), puis laisse l'utilisateur installer un ensemble de contraintes établissant la correspondance entre des points en 3D (D) et des points de l'image (E). Notre énergie quadratique interpole ces contraintes de manière régulière (F).*

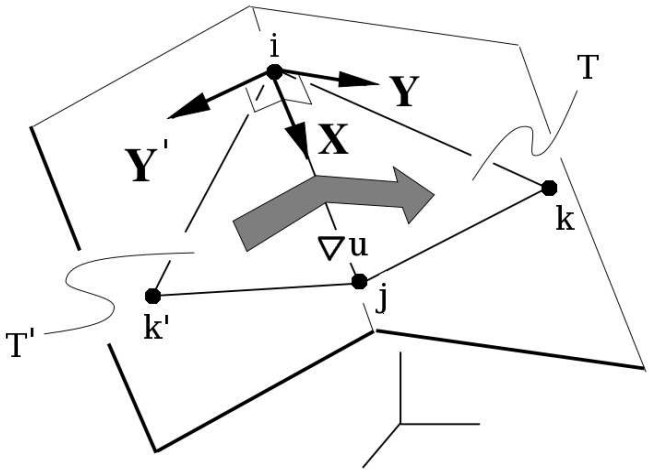


Figure 3.19: *Gradient continu à travers une paire de triangles.*

minimiser les variations du gradient (à savoir de la "vitesse") incitera la paramétrisation à continuer dans la même direction (les iso-u,v seront le plus rectiligne possible) et avec la même "vitesse" (leur espacement restera le plus constant possible). Ceci m'a permis de généraliser la notion de "morphing" (déformation d'images) à des images plaquées sur des objets 3D (c.f. Figure 3.18). Mon approche fonctionne de la manière suivante: en partant d'une surface 3D (Figure 3.18-A) et d'une image à plaquer sur cette surface (Figure 3.18-B), l'objectif est de permettre à l'utilisateur de mettre en correspondence les détails de l'image avec les détails du modèle. Pour ce faire, le système construit déjà automatiquement une première solution, en utilisant la méthode de Tutte-Floater évoquée dans la section précédente (Figure 3.18-C). L'utilisateur va ensuite établir un ensemble de correspondances (ou contraintes), à savoir des couples de points 3D (x_k, y_k, z_k) (Figure 3.18-D) associés à des coordonnées 2D sur l'image (u_k, v_k) (Figure 3.18-E). Le résultat final est visible sur la Figure 3.18-F. Le détail de cette méthode est décrit dans ma publication [Lé01]. Cette Section en reprend les aspects principaux, et établit le lien avec les autres aspects de mon programme de recherche.

Etant donné un ensemble de m contraintes (x_k, y_k, z_k) ; (u_k, v_k), le problème consiste maintenant à définir une fonction objectif devant être minimisée par la paramétrisation, exprimant à la fois le respect des contraintes et un critère de régularité, assurant des variations "naturelles" de la paramétrisation autour des contraintes. Ceci correspond à un problème classique d'ajustement aux données, souvent abordé en minimisant la fonctionelle $D(X)$ suivante, caractérisant une paramétrisation X:

$$D(X) \quad = \quad \sum_{i=1}^{m}(M_i - X(U_i)) + \varepsilon \int_{\Omega} \left(\frac{\partial^2 X}{\partial u^2}\right)^2 + \left(\frac{\partial^2 X}{\partial v^2}\right)^2 dudv$$

Plutôt que de minimiser les dérivées secondes, indéfinies dans le cas d'une surface triangulée, nous allons nous intéresser aux variations des gradients de la paramétrisation, dont les expressions ont été données dans la section précédente. Comme nous le montrons sur la Figure 3.19, nous allons évaluer la déviation $D_u(T, T')$ (resp. $D_v(T, T')$) du gradient de u (resp. v) entre deux triangles T, T' partageant l'arête (i, j):

$$D_u(T, T') = \left\| \begin{pmatrix} \frac{\partial u}{\partial X} \\ \\ \frac{\partial u}{\partial Y} \end{pmatrix} - \begin{pmatrix} \frac{\partial u}{\partial X'} \\ \\ \frac{\partial u}{\partial Y'} \end{pmatrix} \right\|^2 \tag{3.15}$$

Nous munissons la paire de triangle d'une base locale orthonormée (X, Y) et (X, Y'), l'axe des X étant commun aux deux triangles. Dans ces bases, les gradients de u dans T et T' sont donnés par:

$$\begin{aligned} \begin{pmatrix} \partial u/\partial X \\ \partial u/\partial Y \end{pmatrix} &= \begin{pmatrix} -1/d_{i,j} & 1/d_{i,j} & 0 \\ X_k/(d_{i,j} \times Y_k) - 1/Y_k & -X_k/(d_{i,j} \times Y_k) & 1/Y_k \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \\ \begin{pmatrix} \partial u/\partial X' \\ \partial u/\partial Y' \end{pmatrix} &= \begin{pmatrix} -1/d_{i,j} & 1/d_{i,j} & 0 \\ X_{k'}/(d_{i,j} \times Y_{k'}) - 1/Y_{k'} & -X_{k'}/(d_{i,j} \times Y_{k'}) & 1/Y_{k'} \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_{k'} \end{pmatrix} \end{aligned} \tag{3.16}$$

où $d_{i,j} = \|\mathbf{p_j} - \mathbf{p_i}\|$ dénote la longeur de l'arête (i, j).

Comme nous pouvons le remarquer, la coordonnée en X du gradient ne dépend que des valeurs u_i et u_j. Par conséquent, seule la coordonnée en Y joue un rôle dans le calcul de la déviation $D_u(T, T')$, qui est alors donnée par:

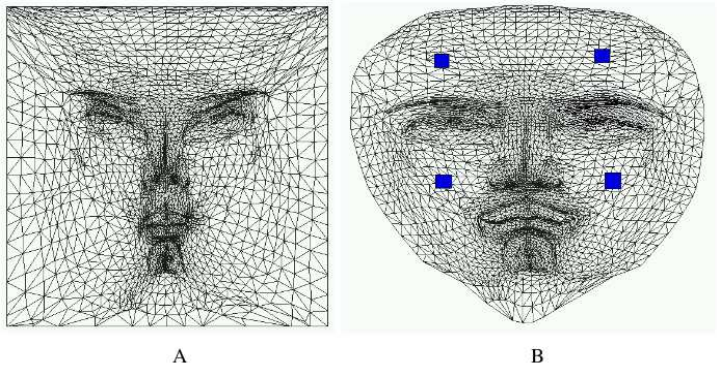


Figure 3.20: *Capacités d'extrapolation offertes par notre méthode. A: le résultat obtenu avec une méthode de type Tutte-Floater; B: le résultat de notre méthode. Notre fonctionelle d'énergie trouve une forme naturelle pour le bord (les 4 carrés bleus correspondent aux points contraints).*

$$
\begin{array}{lll}
D_u(T,T') = (& & \\
 & (X_k/(d_{i,j} \times Y_k) - 1/Y_k + X_{k'}/(d_{i,j} \times Y_{k'}) - 1/Y_{k'}) & u_i \quad + \\
 & (-X_k/(d_{i,j} \times Y_k) - X_{k'}/(d_{i,j} \times Y_{k'})) & u_j \quad + \\
 & 1/Y_k & u_k \quad + \\
 & 1/Y_{k'} & u_{k'} \\
)^2 & &
\end{array}
$$

Ceci permet de définir une fonction objectif, exprimant la régularité de la paramétrisation:

$$D = \sum_{T,T'} (|T| + |T'|) \left(D_u(T,T') + D_v(T,T') \right) \tag{3.17}$$

En utilisant la formulation de type moindres carrés avec réduction des degrés de libertés, détaillée en Section 2.2.4, nous obtenons un système linéaire à résoudre, dont la solution fournit les coordonnées u_i, v_i aux sommets de la triangulation. Cette formulation nous a permis de remplacer les solveurs de type relaxation, alors très largement utilisés dans la communauté du traitement numérique de la géométrie, par la méthode du gradient conjugué, bien plus efficace [Lé01]. A présent, les grands progrès réalisés par les solveurs creux directs[MIT06] permettent d'obtenir une efficacité encore plus grande, et de manipuler en temps réel la paramétrisation même sur des modèles de taille conséquente.

La Figure 3.20 montre les résultats de notre méthode en extrapolation (comparés avec une approche de type Tutte-Floater). Les 4 carrés bleus correspondent aux points fixés, à savoir leurs coordonnées (u,v) sont supprimées de l'ensemble des degrés de liberté du système, en utilisant la méthode expliquée en Section 2.2.4. Comme nous pouvons le constater, le bord de la surface trouve une forme naturelle. La Figure 3.21 montre le résultat de notre méthode appliquée à une texture qui diffère très fortement de la géométrie 3D de la surface. Malgrès cela, notre méthode parvient à générer un plaquage de texture qui semble naturel à l'utilisateur.

Toutefois, bien qu'utile dans un contexte interactif, notre méthode n'offre pas suffisament de garanties pour être utile dans un contexte automatique. En effet, rien ne garantit que la fonction que nous calculons sera bijective. Pour cette raison, nous nous sommes

[MIT06] Omer Meshar, Dror Irony, and Sivan Toledo. An out-of-core sparse symmetric indefinite factorization method. *ACM Transactions on Mathematical Software*, 32:445–471, 2006.

Figure 3.21: *Plaquage de textures sous contraintes.*

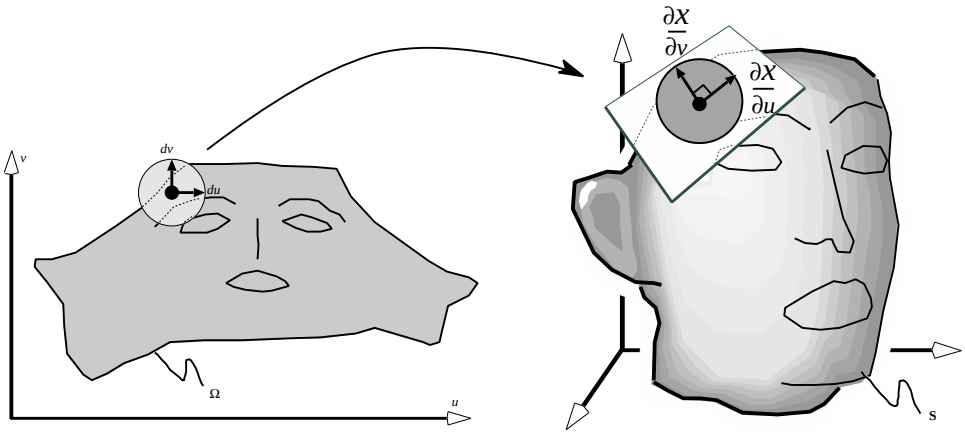


Figure 3.22: *Une paramétrisation conforme transforme un cercle élémentaire en un cercle élémentaire.*

intéressés au formalisme de l'analyse complexe, en particulier celui des applications conformes, qui offrent suffisament de "rigidité" pour offrir un bon comportement en extrapolation. La section suivante détaille ces notions (je recommande au lecteur intéressé par ses aspects de consulter l'excellent ouvrage de Stanley Needham[Nee94]), et présente le travail que j'ai réalisé avec Nicolas Ray, pendant sa thèse, intitulée "atlas de textures multi-résolution".

3.3.3 Cartes conformes au sens des moindres carrés (LSCM)

Comme nous l'avons vu tout au long de cette section, l'analyse des déformations introduite en Section 3.1.2 joue un rôle important dans la définition des méthodes de paramétrisation. Nous nous intéressons à présent à une famille de paramétrisations particulières, pour lesquelles l'ellipse d'anisotropie est un cercle en tout point de la surface. Comme nous le montrons sur la Figure 3.22, ceci signifie que les deux vecteurs gradients $\partial X/\partial u$ et $\partial X/\partial v$ sont orthogonaux et de même norme. Cette condition peut également s'écrire $\partial X/\partial v = N \wedge \partial X/\partial u$. Ceci implique également que la matrice Jacobienne corresponde à la composée d'une rotation et d'une homothétie (autrement dit une similitude). Les applications conformes sont donc localement des similitudes. Contrairement au cheminement exposé dans notre publication initiale [LPRM02], nous prendrons ici comme point de départ le simple calcul des gradients de la surface. Nous développerons plus loin le formalisme de l'analyse complexe.

[Nee94] Tristan Needham. *Visual Complex Analysis*. Oxford Press, 1994.

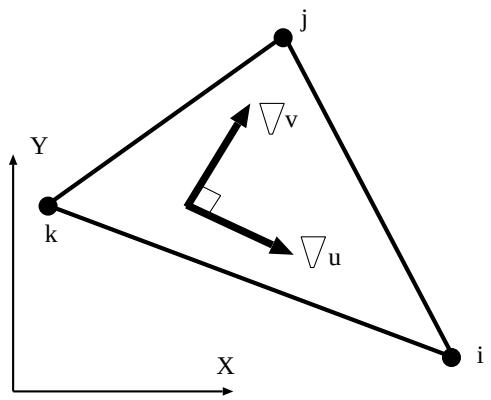


Figure 3.23: *Dans un triangle muni d'une base locale (X,Y), la condition caractérisant les paramétrisations conformes s'exprime relativement facilement.*

Dans le contexte particulier de la paramétrisation des surfaces triangulées, la géométrie est connue, et l'espace paramétrique est inconnu, nous nous intéressons donc à la fonction inverse, qui va de la surface vers l'espace paramétrique. Il convient de noter ici que la matrice Jacobienne de l'inverse d'une fonction X est égale à l'inverse de la matrice Jacobienne de X. Par conséquent, l'inverse d'une application conforme est également conforme, et le raisonnement sur les gradients $\partial X/\partial u$ et $\partial X/\partial v$ se transpose directement aux gradients ∇u et ∇v de la paramétrisation inverse.

Nous considérons à présent l'un des triangles de la surface, muni d'une base orthonormée (X,Y) de son plan support (c.f. Figure 3.23). La condition caractérisant les applications conformes s'exprime alors de la manière suivante:

$$\nabla v = \mathrm{rot}_{90}(\nabla u) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \nabla u \tag{3.18}$$

où rot_{90} dénote la rotation de 90 degrés dans le sens trigonométrique.

Nous rappelons ici l'expression du gradient, calculée en Section 3.2.3:

$$\nabla u = M_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = 1/2|T| \begin{pmatrix} Y_j - Y_k & Y_k - Y_i & Y_i - Y_j \\ X_k - X_j & X_i - X_k & X_j - X_i \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix}$$

où M_T dénote la matrice qui permet de calculer le gradient d'une application linéaire définie sur le triangle en fonction de ses trois valeurs aux sommets du triangle.

L'équation 3.18, caractérisant les applications conformes dans un triangle, se réécrit alors:

$$M_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} M_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Dans le cas continu, toute surface admet une paramétrisation conforme. Toutefois, dans notre cas de surface triangulée munie d'une paramétrisation linéaire par morceaux, seules les surfaces développables admettent une paramétrisation conforme. Ces surfaces développables sont caractérisées par le fait que pour tout sommet i interne, la somme des angles incident au sommet i est égale à 2π. Pour une surface générale (à savoir non développable), nous allons minimiser une énergie qui mesure le caractère non-conforme de la paramétrisation, défini ainsi:

$$E_{LSCM} = \sum_{T=(i,j,k)} |T| \left\| M_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} M_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \right\|^2 \tag{3.19}$$

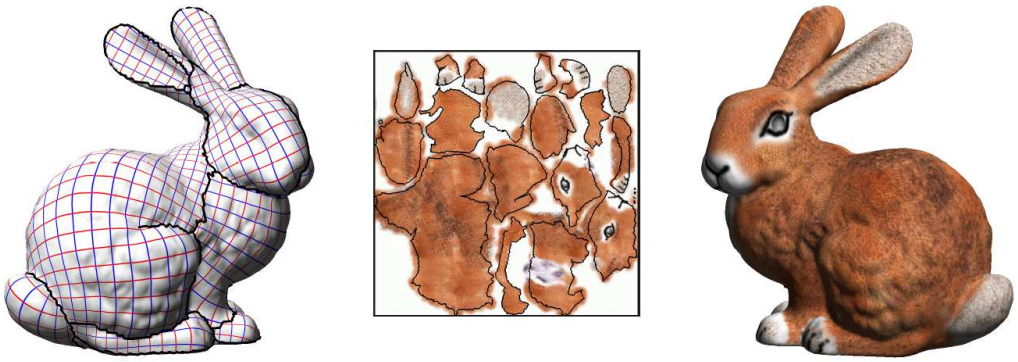


Figure 3.24: *Paramétrisation de surfaces triangulées avec notre méthode LSCM (Least Squares Conformal Maps).*

Cette fonction est une forme quadratique, dont le minimum est susceptible d'être facile à trouver. Toutefois, il convient de préciser que le caractère conforme d'une paramétrisation est invariant par similitude appliquée dans l'espace paramétrique. Par conséquent, la forme quadratique E_{LSCM} n'est pas définie-positive, et sa matrice est singulière. Il est toutefois facile de résoudre le problème, en fixant deux sommets dans l'espace paramétrique, ce qui revient à définir les 4 degrés de liberté d'une similitude. Ces 4 variables sont retirées de l'ensemble des degrés de liberté du système en utilisant la méthode exposée en Section 2.2.4. Nous avons baptisé cette méthode de calcul des coordonnées (u,v) *carte conforme au sens des moindres carrés*, ou encore LSCM[10].

Nous avons démontré avec Sylvain Petitjean que la forme quadratique ainsi obtenue était définie positive. Le lecteur pourra se référer à notre article [LPRM02] pour la preuve complète. Nous en donnons néanmoins l'idée générale: tout disque topologique peut être construit par deux opérations élémentaires, l'une ajoutant un sommet et un triangle au bord, et l'autre reliant trois sommets du bord pour former un nouveau triangle. Nous obtenons le résultat en vérifiant qu'aucune de ces deux opérations ne diminue le rang de la matrice.

Nous avons complété cette méthode avec différents outils, permettant de construire un *atlas de textures*. Un atlas de textures est une représentation efficace de la couleur pour les systèmes de dessins 3D (3D Paint Systems), voir Figure 3.24. Le modèle 3D est décomposé en un ensemble de parties (appelées cartes ou charts), homéomorphes à des disques. Chaque carte est munie d'une paramétrisation grâce à notre méthode LSCM, puis les cartes dépliées sont regroupées dans un espace 2D commun, grâce à un algorithme développé par Nicolas Ray [LPRM02]. Les méthodes existantes pour construire un atlas de textures souffrent de plusieurs limitations, qui causent la génération d'un nombre excessif de petites cartes ayant chacune un bord simple. Dans le contexte du plaquage de textures, les discontinuités entre ces cartes causent divers artéfacts, et compliquent l'application de motifs réguliers sur de grandes zones.

La capacité de LSCM de générer des cartes avec des bords arbitraires permet de réduire le nombre de cartes nécessaire à la construction d'un atlas pour un objet donné. Cette propriété combinée à la simplicité d'implantation de LSCM a facilité la diffusion de cette méthode, à présent intégrée à plusieurs logiciels commerciaux et domaine public (Gocad, Maya et Blender).

Nous avons abordé ici LSCM sous l'aspect des relations entre les gradients ∇u et ∇v, calculés dans chaque triangle. Dans la section suivante, nous montrons des liens plus profonds entre notre méthode et des notions d'analyse complexe. Ceci permet également de

[10] pour Least Squares Conformal Maps

mieux comprendre la relation avec la méthode de Tutte-Floater ainsi que ses généralisations récentes.

3.3.4 Cartes conformes et fonctions harmoniques

La notion d'application conforme joue un rôle particulier dans l'analyse complexe et dans la géométrie Riemannienne. Le système d'équations suivant caractérise les applications conformes:

$$\begin{aligned} \frac{\partial v}{\partial x} &= -\frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial y} &= \frac{\partial u}{\partial x} \end{aligned}$$

Ce système d'équation est connu sous le nom d'équations de Cauchy-Riemann. Ces équations jouent un rôle particulier dans l'analyse complexe, car elles définissent les fonctions complexes différentiables (appelées fonctions analytiques). On s'intéresse à la fonction complexe $U(X) = u(X) + iv(X)$ avec $X = x + iy$. Il est possible de vérifier que la dérivée $U'(X)$ de U en X définie par:

$$U'(X) = \lim_{A \to 0} \frac{U(X) - U(A)}{X - A}$$

n'existe que si les équations de Cauchy-Riemann sont satisfaites.

Une manière possible de comprendre cette relation est donnée par le développement limité à l'ordre 1. Dans le cas d'une fonction f de $\mathbb{R}$ dans $\mathbb{R}$, le développement limité de f s'écrit $f(x_0 + a) \simeq f(x_0) + f'(x_0)a$. Ceci permet de réaliser une approximation linéaire locale de la fonction f autour du point x_0.

Si nous nous intéressons à présent à une application complexe $U(X)$, quand la dérivée complexe est définie, le développement limité s'écrit $U(X_0 + A) \simeq U(X_0) + U'(X_0)A$. Si nous considérons la fonction complexe U comme une transformation du plan, et en écrivant $U'(X_0) = Re^{i\theta}$ sous forme polaire, nous obtenons $U(X_0 + A) \simeq U(X_0) + Re^{i\theta}A$. Autrement dit, les fonctions complexes dérivables se comportent localement comme une similitude du plan (composée d'une translation de vecteur $U(X_0)$, d'une rotation d'angle θ et d'une homothétie de rapport R). Ceci donne une autre explication des équations de Cauchy-Riemann: la rotation et l'homothétie appliquée aux deux gradients doit être la même.

Ces notations faisant appel aux nombres complexes permettent d'aboutir à une formulation plus élégante de l'énergie minimisée par notre méthode LSCM. Les deux composantes des gradients, constantes sur chaque triangle, peuvent être regroupées sous la forme d'un unique nombre complexe:

$$\nabla u = \frac{\partial u}{\partial x} + i\frac{\partial u}{\partial y} = \frac{i}{|T|}\begin{pmatrix} W_1 & W_2 & W_3 \end{pmatrix}\begin{pmatrix} u_1 & u_2 & u_3 \end{pmatrix}^\top,$$

avec

$$\begin{cases} W_1 &= (X_3 - X_2) + i(Y_3 - Y_2), \\ W_2 &= (X_1 - X_3) + i(Y_1 - Y_3), \\ W_3 &= (X_2 - X_1) + i(Y_2 - Y_1). \end{cases}$$

L'énergie conforme dans un triangle s'écrit alors:

$$E_{LSCM}(T) = |T|\,|\nabla V - i\nabla U|^2 = \begin{pmatrix} W_1 & W_2 & W_3 \end{pmatrix}\begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix}$$

avec

$$\begin{cases} U_1 &= u_1 + iv_1 \\ U_2 &= u_2 + iv_2 \\ U_3 &= u_3 + iv_3 \end{cases}$$

Cette formulation plus compacte nous a été utile pour prouver deux propriétés intéressantes de notre méthode LSCM, à savoir que notre énergie quadratique est définie positive, et que le résultat est indépendent du maillage [LPRM02]. Nous montrons ici une autre formulation qui met en évidence le rôle symétrique joué par les coordonnées dans l'espace paramétrique et les coordonnées dans l'espace géométrique. Si nous notons $P_1 = X_1 + iY_1$ (resp. P_2, P_3), les nombres complexes W_1, W_2, W_3 qui correspondent aux vecteurs s'appuyant sur les côtés du triangle sont donnés par:

$$\begin{pmatrix} W_1 \\ W_2 \\ W_3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

En utilisant cette expression, il vient:

$$E_{LSCM} = \frac{1}{2} \left| \begin{pmatrix} P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} \right|^2$$

Une autre propriété intéressante des fonctions complexes dérivables est que leur dérivabilité à l'ordre 1 implique qu'elles le soient à n'importe quel ordre. Nous pouvons alors utiliser les équations de Cauchy-Riemann pour calculer les dérivées de u (resp. v) à l'ordre 2 et établir des liens intéressant avec le Laplacien:

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= \frac{\partial^2 v}{\partial x \partial y} \\ \frac{\partial^2 u}{\partial y^2} &= -\frac{\partial^2 v}{\partial x \partial y} \end{aligned}$$

soit:

$$\begin{aligned} \Delta u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= 0 \\ \Delta v &= \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} &= 0 \end{aligned}$$

Autrement dit, la partie réelle et la partie imaginaire d'une application conforme sont deux fonctions harmoniques (qui annulent le Laplacien). C'est le point de vue qui a été utilisé par Desbrun *et. al* pour développer leur méthode de paramétrisation conforme [DMA02], quasiment équivalente à notre méthode LSCM. Ainsi, Desbrun *et. al* calculent deux fonctions harmoniques en laissant le bord évoluer. Sur le bord, un ensemble de contraintes garantissent le caractère conforme de la paramétrisation en rétablissant le couplage entre les variables u_i et v_i.

Une autre manière de considérer le rapport entre ces deux méthode, mentionnée par Pinkall et Polthier[PP93a], et sans doute à la base de l'intuition de Desbrun *et. al*, est donnée

[DMA02] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of Eurographics*, pages 209–218, 2002.

[PP93a] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Math.*, 2(15), 1993.

par le problème de Plateau[Pla73,Mee81]. Etant donnée une courbe fermée, ce problème concerne l'existence d'une surface d'aire minimum dont le bord s'appuie sur cette courbe. Pour minimiser l'aire d'une surface, Douglas[Dou31] et Rado[Rad30], et plus tard Courant[Cou50] ont considéré l'énergie de Dirichlet (à savoir l'intégrale de la somme des carrés des gradients), plus facile à manipuler. Une discrétisation de cette énergie a été proposée par Pinkall et Polthier[PP93a], dans le but de résoudre le problème de Plateau dans le cas discret. L'énergie de Dirichlet diffère de l'aire de la surface d'un terme qui dépend de la paramétrisation, appellé *énergie conforme*, et qui s'annule lorsque la paramétrisation est conforme. La relation entre ces trois quantités est explicitée ci-dessous:

$$\underbrace{\int_S \det(J)ds}_{\text{aire de la surface}} = \underbrace{\frac{1}{2}\int_S \|\nabla_u X\|^2 + \|\nabla_v X\|^2 ds}_{\text{énergie de Dirichlet}} - \underbrace{\frac{1}{2}\int_S \|\nabla_v X - \mathrm{rot}_{90}(\nabla_u X)\|^2}_{\text{énergie conforme}}$$

avec:

$$\nabla_u X = \begin{pmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \end{pmatrix} \quad ; \quad \nabla_v X = \begin{pmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \end{pmatrix}$$

Cette relation se démontre facilement, en établissant l'égalité entre les différents termes intégrés:

$$\underbrace{\frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u}}_{\det(J)} = \underbrace{\frac{1}{2}\left(\left\|\begin{pmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \end{pmatrix}\right\|^2 + \left\|\begin{pmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \end{pmatrix}\right\|^2\right)}_{\|\nabla_u X\|^2 + \|\nabla_v X\|^2} - \underbrace{\frac{1}{2}\left(\left\|\begin{pmatrix} \frac{\partial x}{\partial v} + \frac{\partial y}{\partial u} \\ \frac{\partial y}{\partial v} - \frac{\partial x}{\partial u} \end{pmatrix}\right\|^2\right)}_{\|\nabla_v X - \mathrm{rot}_{90}(\nabla_u X)\|^2}$$

Ainsi, notre méthode LSCM minimise l'énergie conforme, et la méthode de Desbrun *et.al* minimise l'énergie de Dirichlet. Comme la différence entre ces deux quantités correspond à l'aire de la surface, constante, les deux méthodes sont équivalentes. Nous nous intéressons à présent à une interprétation plus géométrique de la notion d'application conforme dans le cas de fonctions linéaire par morceaux.

3.3.5 Cartes conformes analytiques et géométriques

Dans la section précédente, nous avons vu une définition *analytique* de la notion d'application conforme (à savoir en termes de relations entre les gradients). Dans cette section, nous allons voir une définition *géométrique* de ce critère, caractérisant les relations entre la forme des triangles dans l'espace 3D et dans l'espace paramétrique.

Nous supposons à présent que la forme des triangles dans l'espace 3D n'est connue qu'à travers les angles aux sommets. Nous allons tout d'abord appliquer quelques transformations à l'expression de l'énergie E_{LSCM}, dans l'objectif de faire apparaître les rapports $Z_1 = -E_2/E_3$, $Z_2 = -E_3/E_1$ et $Z_3 = -E_1/E_2$ entre les nombre complexes correspondant aux vecteurs qui s'appuient sur les côtés. Comme le montre la Figure 3.25, le nombre complexe $Z_1 = -E_2/E_3 = Re^{i\theta}$ correspond à la composée d'une rotation d'angle θ et d'une homothétie de rapport R qui transforme le côté $\overrightarrow{P_1P_2} = E_3$ en $\overrightarrow{P_1P_3} = -E_2$. Ainsi, l'énergie E_{LSCM} se réécrit:

[Pla73] J.A.F. Plateau. *Statistique Experimentale et Theorie des Liquides Soumis aux Seules Forces Moléculaires*. Gauthier-Villars, 1873.

[Mee81] William H. Meeks. A survey of the geometric results in the classical theory of minimal surfaces. In *Bol. Soc. Brasil. Mat. 12*, number 1, pages 29–86, 1981.

[Dou31] J. Douglas. Solution of the problem of plateau. In *Trans. Amer. Math. Soc.*, volume 33, 1931.

[Rad30] T. Rado. The problem of least area and the problem of plateau. In *Math. Zeit*, volume 32, 1930.

[Cou50] R. Courant. *Dirichlet's Principle, Conformal Mapping and Minimal Surfaces*. Interscience, 1950.

[PP93a] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Math.*, 2(15), 1993.

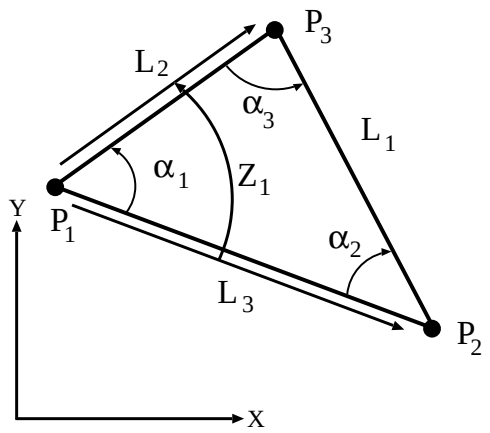


Figure 3.25: *Relations dans un triangle.*

$$
\begin{aligned}
E_{LSCM} &= \tfrac{1}{2}\left|E_1U_1 + E_2U_2 + E_3U_3\right|^2 \\
&= \tfrac{1}{2}\left|E_3(U_3 - U_1) + E_2(U_2 - U_1)\right|^2 \\
&= \tfrac{|E_3|}{2}\left|(U_3 - U_1) + \tfrac{E_2}{E_3}(U_2 - U_1)\right|^2 \\
&= \tfrac{|E_3|}{2}\left|(U_3 - U_1) - Z_1(U_2 - U_1)\right|^2
\end{aligned}
$$

Nous pouvons maintenant exprimer le terme Z_1 en fonction des angles α_1, α_2 et α_3 au sommet du triangle. Nous utilisons en particulier la relation:

$$\frac{L_1}{\sin(\alpha_1)} = \frac{L_2}{\sin(\alpha_2)} = \frac{L_3}{\sin(\alpha_3)}$$

Ainsi, il vient $L2/L3 = \sin(\alpha_2)/\sin(\alpha_3)$, et Z_1 peut alors s'écrire:

$$Z_1 = \frac{\sin(\alpha_2)}{\sin(\alpha_3)} e^{i\alpha_1}$$

Etant donnés les angles aux sommets $\alpha_1, \alpha_2, \alpha_3$, en supposant que le côté $\overrightarrow{P_1P_2}$ est de longueur 1, l'énergie E_{LSCM} s'écrit alors:

$$E_{LSCM} = \frac{1}{2}\left|(U_3 - U_1) - \frac{\sin(\alpha_2)}{\sin(\alpha_3)} e^{i\alpha_1}(U_2 - U_1)\right|^2 \tag{3.20}$$

En mesurant les angles α_1, α_2 et α_3 sur le maillage initial, nous retrouvons la méthode LSCM décrite dans la section précédente[11]. Nous allons à présent voir des méthodes permettant de mieux choisir ces angles.

La grande facilité d'implantation de LSCM a favorisé sa diffusion, à la fois dans la communauté scientifique et dans le monde industriel. Toutefois, LSCM souffre de deux sérieuses limitations:

- ◇ Comme la fonction minimisée par LSCM correspond à une énergie harmonique, des triangles ayant des angles fortement obtus peuvent générer des coefficients négatifs dans la matrice, qui vont se traduire par des retournements locaux de triangles (la fonction n'est plus bijective);

[11] à la pondération par la longueur de l'arête $\overrightarrow{P_1P_2}$ près

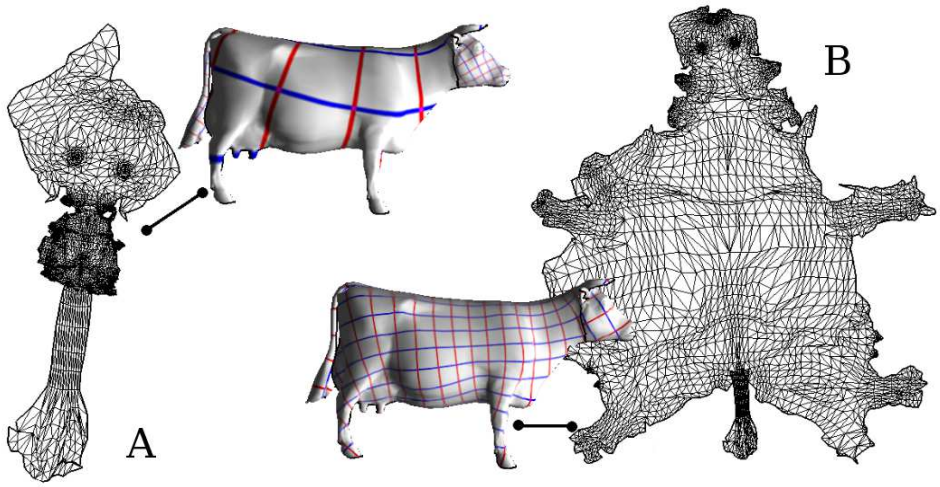


Figure 3.26: *Pour des surfaces présentant une forte courbure Gaussienne, notre méthode LSCM génère des résultats fortement déformés, loin de ce que l'utilisateur attendrait (A). La méthode ABF donne de bien meilleurs résultats (B), mais introduit des temps de calculs prohibitifs. Nous présentons ici la méthode ABF++, combinant les avantages des deux approches.*

◇ La solution est dépendante des deux points fixés, et peut parfois comporter de grandes déformations, peu naturelles. Le phénomène est particulièrement visible lorsque la surface présente une forte courbure Gaussienne. Ainsi, la Figure 3.26 montre ce phénomène: le museau et la queue de la vache sont fortements grossis, alors que les pattes dégénèrent.

Pour cette raison, nous allons étudier dans la section suivante une méthode qui génère une meilleure paramétrisation, sans retournements de triangles et avec une meilleure répartition des déformations.

3.3.6 Méthodes géométriques (ABF et ABF++)

Cette section présente la méthode ABF (Angle Based Flattening)[SdS01], développée par A. Sheffer *et. al*, ainsi que les différentes améliorations que nous avons développées en coopération avec l'auteur afin de rendre cette méthode appliquable à de grands maillages.

La méthode ABF se fonde sur l'observation suivante: l'espace paramétrique forme une triangulation 2D, définie de manière unique par les angles aux sommets des triangles en 2D (modulo une similitude appliquée dans l'espace paramétrique). Cette simple remarque a conduit A. Sheffer et E. De Sturler à re-formaliser le problème de paramétrisation de surface – à savoir trouver les (u_i, v_i) – en termes d'angles aux sommets, à savoir trouver les α_i^t, où α_i^t dénote l'angle au triangle t incident au sommet i. Nous verrons comment cette formulation permet facilement de minimiser les déformations, d'une manière mieux équilibrée que notre méthode LSCM.

Formulation ABF

En termes d'angles, l'énergie minimisée par ABF s'écrit:

$$E(\alpha) = \sum_{t \in T} \sum_{k=1}^{3} \frac{1}{w_k^t} (\alpha_k^t - \beta_k^t)^2. \tag{3.21}$$

[SdS01] Alla Sheffer and Eric de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17:326–337, 2001.

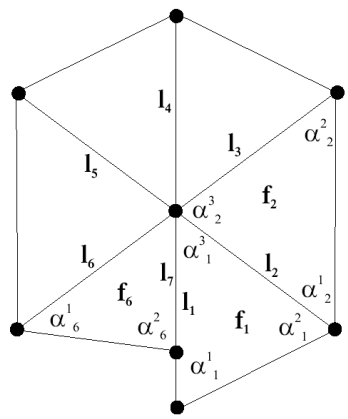


Figure 3.27: *A Violation de la contrainte de reconstruction: en faisant le tour autour du sommet k, on ne retombe pas sur le point de départ.*

où les α_k^t sont les angles 2D inconnus, et où les β_k^t sont les angles "optimaux", mesurés sur le maillage en 3D.

L'indice t parcourt l'ensemble T des triangles du maillage, et l'incide k parcourt les trois angles de chaque triangle. Les poids w_k^t sont positionnés à $\frac{1}{\beta_k^{t\,2}}$ afin de mesurer une déformation angulaire *relative* plutôt que *absolue*.

Afin d'assurer que les angles construits par l'algorithme définissent bien une paramétrisation valide, un ensemble de contraintes doit être pris en compte, et inclus dans la formulation, sous forme d'un problème d'optimisation contrainte (voir Section 2.2.6):

◇ Validité des triangles (pour chaque triangle t):

$$\forall t \in T, \quad C_{Tri}(t) = \alpha_1^t + \alpha_2^t + \alpha_3^t - \pi = 0; \tag{3.22}$$

◇ Planarité (pour chaque sommet interne v):

$$\forall v \in V_{int}, \quad C_{Plan}(v) = \sum_{(t,k)\in v^*} \alpha_k^t - 2\pi = 0, \tag{3.23}$$

où V_{int} est l'ensemble des sommets internes, et où v^* dénote l'ensemble des angles incident au sommet v.

◇ Reconstruction (pour chaque sommet interne) - cette contrainte assure que les côtés partagés par des paires de triangles ont la même longeur :

$$\forall v \in V_{int}, \quad C_{Len}(v) = \prod_{(t,k)\in v^*} \sin \alpha_{k\oplus 1}^t - \prod_{(t,k)\in v^*} \sin \alpha_{k\ominus 1}^t = 0. \tag{3.24}$$

Les indices $k \oplus 1$ et $k \ominus 1$ dénote l'angle suivant et l'angle précedent dans le triangle. La Figure 3.27) montre un exemple où cette contrainte n'est pas satisfaite: le produit $\sin \alpha_{k\oplus 1}^t \sin \alpha_{k\ominus 1}^t$ correspondant au rapport des longeurs de deux arêtes consécutives autour du sommet k, si leurs produits ne se correspondent pas, il est possible de "faire un tour" autour du sommet k sans "retomber" sur le point de départ.

Algorithme de résolution numérique

Le problème d'optimisation numérique contrainte peut se résoudre par la méthode des multiplicateurs de Lagrange (c.f. section 2.2.6). Des multiplicateurs de Lagrange sont associés à chaque type de contrainte $(\lambda_{Tri}, \lambda_{Plan}, \lambda_{Len})$. Le Lagrangien F s'écrit:

$$F(x) = F(\alpha, \lambda_{Tri}, \lambda_{Plan}, \lambda_{Len}) = E + \sum_t \lambda_{Tri}^t C_{Tri}(t) + \sum_v \lambda_{Plan}^v C_{Plan}(v) + \sum_v \lambda_{Len}^v C_{Len}(v).$$

Sheffer et de Sturler[SdS01] calculent un point fixe du Lagrangien en utilisant la méthode de Newton (c.f. Section 2.2.5), de la manière suivante:

$$\begin{array}{l} \textbf{tantque} \quad \|\nabla F(x)\| > \varepsilon \\ \quad \textbf{resoudre} \quad \nabla^2 F(x)\delta = -\nabla F(x) \\ \quad x \leftarrow x + \delta \\ \textbf{fin} \end{array} \tag{3.25}$$

Le Hessien $\nabla^2 F(x)$ (matrice des dérivées secondes, c.f. Section 2.2.3) est une matrice de dimension $4n_f + 2n_{int}$ où $n_f = |T|$ dénote le nombre de triangles du maillage, et où $n_{int} = |V_{int}|$ dénote le nombre de sommets internes.

Nous avons donc au total $3n_f$ variables, et $n_f + 2n_{int}$ multiplicateurs de Lagrange. Les systèmes linéaires mis en jeu par la méthode de Newton sont résolus par l'une des méthode évoquée en section 2.2.2 (l'article initial utilisait SuperLU).

Amélioration de l'algorithme de résolution (ABF++)

Intéressé par la qualité des résultats obtenus par la méthode ABF (Figure 3.26), j'ai étudié cette méthode. Malheureusement, le caractère non-linéaire du problème d'optimisation contrainte ainsi que la grande taille des matrices mises en jeu empêche l'utilisation de cette méthode pour des maillages de plus d'une dizaine de milliers de sommets.

Pour cette raison, j'ai étudié de plus près le problème d'optimisation numérique: une particularité de ce problème m'a frappée: le terme d'energie quadratique $\sum(\alpha_i^t - \beta_i^t)^2$ est très simple, il s'agit d'une forme quadratique dont la matrice est égale à la matrice identité. Le minimiseur est connu directement, à savoir $\forall i, \alpha_i^t = \beta_i^t$. En conséquence, la difficulté du problème ne provient pas du terme d'énergie, mais des contraintes. Nous allons voir comment ceci se traduit dans la structure du Hessien du Lagrangien, et comment la méthode du complément de Schur permet de découpler les variables et les multiplicateurs de Lagrange, de manière à simplifier grandement les calculs. D'autre part, afin de calculer les coordonnées (u_i, v_i) à partir des angles aux sommets des triangles, j'ai proposé de remplacé l'algorithme "glouton" de l'article initial par l'algorithme LSCM géométrique (Equation 3.20), qui a la propriété intéressante de ré-exprimer ce problème sous forme d'une fonctionelle d'énergie globale, et ainsi de moins soufrir des problèmes d'accumulation d'erreur rencontrés par la méthode précédente.

J'ai développé cette méthode en coopération avec le premier auteur de la méthode ABF (Alla Sheffer). Nous avons publié la méthode améliorée (ABF++) dans le journal ACM Transactions on Graphics [SLMB04].

Programmation linéaire contrainte séquentielle

La formulation initiale de la méthode ABF se fonde sur la minimisation sous contraintes d'une forme quadratique. Comme nous l'avons remarqué, cette forme quadratique est très simple, puisque son optimum est déjà connu (à savoir, les angles optimum β, mesurés sur le maillage en 3D). Les contraintes, quant à elles, sont relativement complexes, en particulier la contrainte de reconstruction (Equation 3.24).

Pour simplifier le problème, ma première idée a été d'utiliser la programmation contrainte linéaire séquentielle [NW00]. Cette technique consiste à remplacer le problème initial (avec des contraintes non linéaires) par une série de problèmes munis de contraintes linéaires. Ces contraintes linéaires sont obtenues par développement limité à l'ordre 1 des contraintes autour du point courant. En d'autre termes, ceci revient à négliger les termes

[SdS01] Alla Sheffer and Eric de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17:326–337, 2001.

[NW00] Nocedal and Wright. *Numerical Optimization*. Springer, 2000.

provenant des dérivées d'ordre 2 des contraintes, apparaissant dans la matrice Hessienne du Lagrangien $\nabla^2 F(x)$. Ceci simplifie le système résolu à chaque itération (3.25). Le prix à payer est une légère augmentation du nombre total d'itérations à affectuer avant d'atteindre la convergence.

Ainsi, le système linéaire $\nabla^2 F(x)\delta = -\nabla F(x)$ résolu à chaque itération devient:

$$\begin{bmatrix} \Lambda & J^t \\ J & 0 \end{bmatrix} \begin{bmatrix} \delta_\alpha \\ \delta_\lambda \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \text{avec:} \tag{3.26}$$

$$\Lambda = \text{diag}\left(\frac{2}{w_k^t}\right), \quad J = \left[\frac{\partial^2 F}{\partial \lambda_i \partial \alpha_k^t}\right], \quad b_1 = -\nabla_\alpha F, \quad b_2 = -\nabla_\lambda F.$$

Comme les contraintes sont à présent linéaires, le bloc supérieur gauche (Λ) devient une matrice *diagonale*. Le bloc inférieur droit est nul. En nous fondant sur cette structure très particulière de la matrice, il est possible de diminuer de manière significative la dimension des systèmes linéaires résolus à chaque itération. Pour ce faire, nous allons décomposer le Hessien en différents blocs, et utiliser deux fois la technique du complément de Schur.

Première décomposition du Hessien

Le système 3.26 peut alors s'écrire de la manière suivante:

$$\Lambda\delta_\alpha + J^t\delta_\lambda = b_1 \tag{3.27}$$

$$J\delta_\alpha = b_2. \tag{3.28}$$

Il est maintenant possible de séparer le calcul du vecteur δ_λ associé aux multiplicateurs de Lagrange, et d'exprimer le vecteur δ_α associé aux variable en fonction de δ_λ:

$$J\Lambda^{-1}J^t\delta_\lambda = b^* \quad \text{avec} \quad b^* = J\Lambda^{-1}b_1 - b_2 \tag{3.29}$$

$$\delta_\alpha = \Lambda^{-1}(b_1 - J^t\delta_\lambda) \tag{3.30}$$

La premère ligne (3.29) est obtenue en multipliant (3.27) par $J\Lambda^{-1}$ et en substituant $J\delta_\alpha$ par l'expression donnée par (3.28). La seconde ligne (3.30) est obtenue en multipliant (3.27) par Λ^{-1}. Nous attirons l'attention du lecteur sur le fait que la matrice Λ est diagonale, et que donc le calcul de Λ^{-1} est trivial. Cette technique est connue sous le nom de *complément de Schur*.

En utilisant ces expressions, l'algorithme (3.25) s'écrit:

tant que $\|\nabla F(x)\| > \varepsilon$
  calculer b, J, Λ
  resoudre Equation 3.29 $\rightarrow \delta_\lambda$
  $\delta_\alpha \leftarrow \Lambda^{-1}(b_1 - J^t\delta_\lambda)$ (Equation 3.30)
  $\lambda \leftarrow \lambda + \delta_\lambda$; $\alpha \leftarrow \alpha + \delta_\alpha$ $/*x = (\alpha, \lambda)*/$
fin

Le système initial de dimension $4n_f + 2n_{int}$ a été remplacé par un système plus petit, de dimension $n_f + 2n_{int}$, où la matrice $J\Lambda^{-1}J$ ne dépend que du Jacobien des contraintes J et de la matrice diagonale Λ (c.f. Equation 3.29). Résoudre ce système fournit le vecteur δ_λ permettant de mettre à jour les multiplicateurs de Lagrange. Il devient alors facile de calculer le vecteur δ_α mettant à jour les variables, en fonction de δ_λ (c.f. Equation 3.30). Comme la dimension du système linéaire résolu à chaque itération est bien plus petite que celle de l'algorithme initial, ce nouvel algorithme est bien plus efficace. Nous montrons à présent comment réduire encore plus la dimension de ces systèmes linéaires, en analysant la structure de la matrice $J\Lambda^{-1}J^t$ et en lui appliquant une fois de plus la technique du complément de Schur.

Seconde décomposition du Hessien

Pour analyser la structure particulière de $J\Lambda^{-1}J^t$, nous pouvons décomposer le Jacobien des contraintes J en deux sous-matrices J_1 et J_2:

$$J = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}, \tag{3.31}$$

où $J_1(n_f \times 3n_f)$ est le Jacobian des contraintes C_{Tri} et où $J_2(2n_{int} \times 3n_f)$ est le Jacobien des contraintes C_{Plan} et C_{Len}. Nous remarquons que J_1 présente une structure très simple:

$$J_1 = \begin{bmatrix} 1\,1\,1 & & & 0\,0\,0 \\ & 1\,1\,1 & & \\ & & \ddots & \\ 0\,0\,0 & & & 1\,1\,1 \end{bmatrix}.$$

De plus, ses colonnes sont orthogonales et linéairement indépendantes. Nous pouvons alors décomposer $J\Lambda^{-1}J^t$ de la manière suivante:

$$J\Lambda^{-1}J^t = \begin{bmatrix} \Lambda^* & J^{*t} \\ J^* & J^{**} \end{bmatrix} \quad \text{avec} \quad \begin{cases} \Lambda^* & (n_f \times n_f) & = & J_1\Lambda^{-1}J_1^t \\ J^* & (2n_{int} \times n_f) & = & J_2\Lambda^{-1}J_1^t \\ J^{**} & (2n_{int} \times 2n_{int}) & = & J_2\Lambda^{-1}J_2^t \end{cases} \tag{3.32}$$

En utilisant la décomposition par blocs de la matrice $J\Lambda^{-1}J^t$, l'équation 3.29 devient:

$$\begin{bmatrix} \Lambda^* & J^{*t} \\ J^* & J^{**} \end{bmatrix} \begin{bmatrix} \delta_{\lambda_1} \\ \delta_{\lambda_2} \end{bmatrix} = \begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix}. \tag{3.33}$$

En d'autres termes,

$$\Lambda^*\delta_{\lambda_1} + J^{*t}\delta_{\lambda_2} = b_1^* \tag{3.34}$$
$$J^*\delta_{\lambda_1} + J^{**}\delta_{\lambda_2} = b_2^*. \tag{3.35}$$

Puisque J_1 est orthogonale et que Λ^{-1} est diagonale, la matrice $\Lambda^* = J_1\Lambda^{-1}J_1^t$ est diagonale. Nous pouvons donc à présent calculer le vecteur δ_{λ_2} indépendament du reste, en résolvant le système suivante:

$$\left(J^*\Lambda^{*-1}J^{*t} - J^{**}\right)\delta_{\lambda_2} = J^*\Lambda^{*-1}b_1^* - b_2^*. \tag{3.36}$$

L'équation 3.36 a été obtenue en multipliant (3.34) par $J^*\Lambda^{*-1}$, puis en substituant $J^*\delta_{\lambda_1}$ par son expression (3.35). Ensuite, le vecteur δ_{λ_1} peut être calculé en fonction de δ_{λ_2}:

$$\delta_{\lambda_1} = \Lambda^{*-1}\left(b_1^* - J^{*t}\delta_{\lambda_2}\right) \tag{3.37}$$

en multipliant (3.34) par Λ^{*-1}.

Nous remarquons à présent que le calcul de δ_{λ_2} requiert la résolution d'un système linéaire de dimension $2n_{int}$, alors que la formulation du problème initial mettait en jeu un système de dimension $4n_f + 2n_{int}$. En nous fondant sur la formule d'Euler $n_f \approx 2n_{int}$, nous estimons que $4n_f + 2n_{int} \approx 10n_{int}$. Ainsi, notre nouvelle formulation a pour résultat la réduction de la dimension des matrice d'un facteur 5.

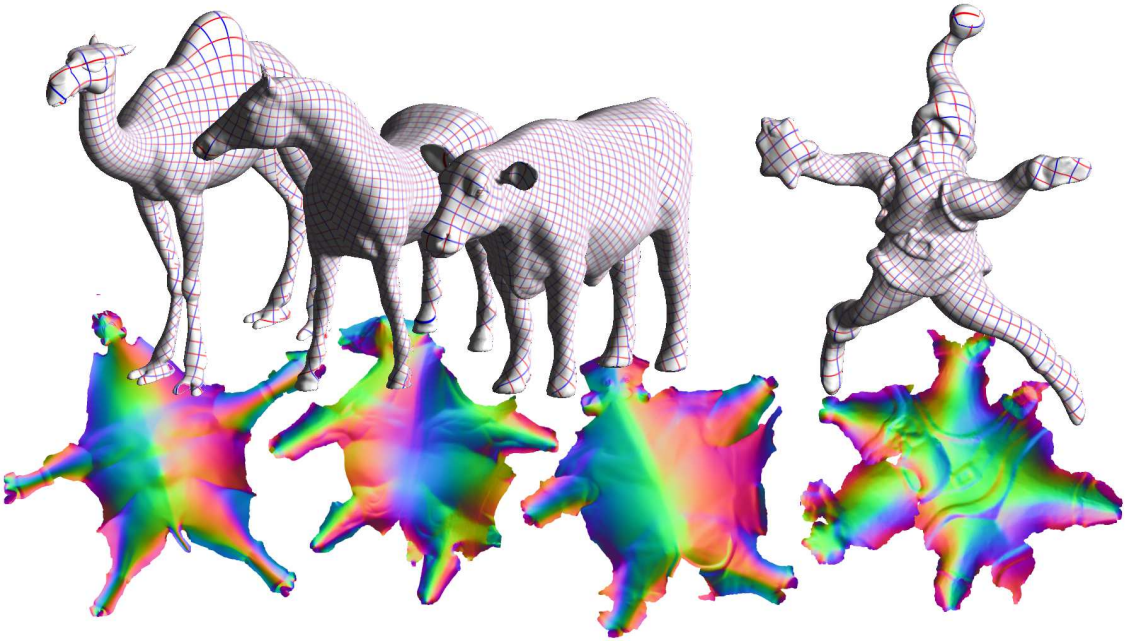


Figure 3.28: *Quelques exemples de paramétrisation obtenues grâce à notre méthode ABF++. Les résultats offrent à la fois plus de garanties (signe du Jacobien constant) et une meilleure qualité que les méthodes précédentes (y compris notre méthode LSCM).*

Algorithme ABF++

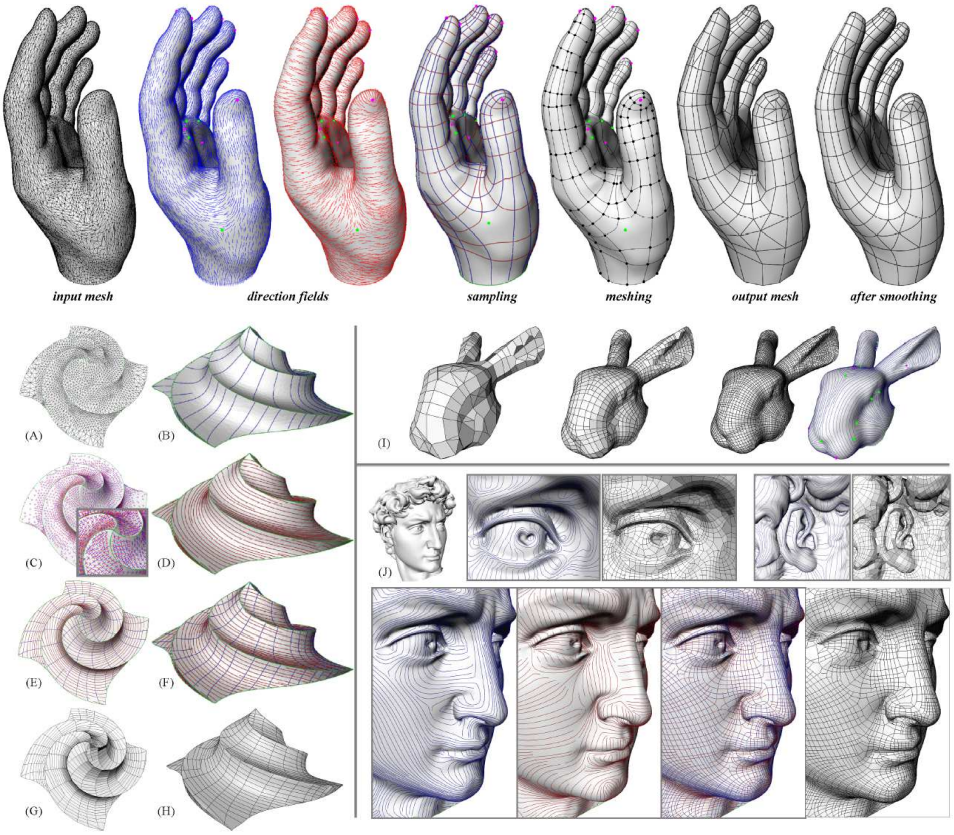
Nous pouvons à présent combiner ces différents calcul pour décrire l'algorithme de paramétrisation ABF++:

$$
\begin{array}{l}
\textbf{tant que} \quad \|\nabla F(x)\| > \varepsilon \\
\quad \text{calculer } b, J, \Lambda \\
\quad \text{resoudre Equation } 3.36 \rightarrow \delta_{\lambda_2} \\
\quad \text{calculer } \delta_{\lambda_1} (\text{Equation } 3.37) \\
\quad \text{calculer } \delta_{\alpha} (\text{Equation } 3.30) \\
\quad \lambda_1 \leftarrow \lambda_1 + \delta_{\lambda_1} \quad ; \quad \lambda_2 \leftarrow \lambda_2 + \delta_{\lambda_2} \quad ; \quad \alpha \leftarrow \alpha + \delta_{\alpha} \quad /*x = (\alpha, \lambda)*/ \\
\textbf{fin}
\end{array}
\tag{3.38}
$$

En comparaison de la formulation Newton initiale (3.25), la nouvelle méthode requiert quelques itérations de plus pour converger (typiquement de 8 à 10 au lien de 5). Toutefois, à chaque itération, les systèmes linéaires manipulés sont de dimension 5 fois plus petite. Nous utilisons SuperLU, un solveur direct creux pour résoudre le système linéaire (equation 3.36) L'algorithme final est plus de 10 fois plus rapide que l'algorithme initial. En conséquence, des surfaces de plus de 100K triangles peuvent à présent être paramétrisées en moins d'une minute. Un autre avantage est la réduction de la taille mémoire nécessaire au stockage des matrice. La Figure 3.28 montre quelques exemples de paramétrisations calculées avec notre approche. Plus récemment, nous avons travaillé avec Rhaleb Zayer à une méthode de paramétrisation dérivée également d'ABF, mais fondée sur une estimation de l'erreur à l'ordre 1 [ZLS07]. Cette dernière est bien plus simple à implanter, et également très efficace, car elle ne nécessite que la résolution de deux systèmes linéaires. Afin de garder la taille de ce mémoire raisonnable, nous ne détaillerons pas cette méthode ici.

Dans les sections suivantes, nous allons voir différentes applications des méthodes de paramétrisation. Au delà du plaquage de textures, application naturelle de ce type de méthode [Lé01], [LPRM02], nous avons expérimenté plusieurs autres méthodes, tirant parti du découplage entre la géométrie et des propriétés stockées dans l'espace paramétrique.

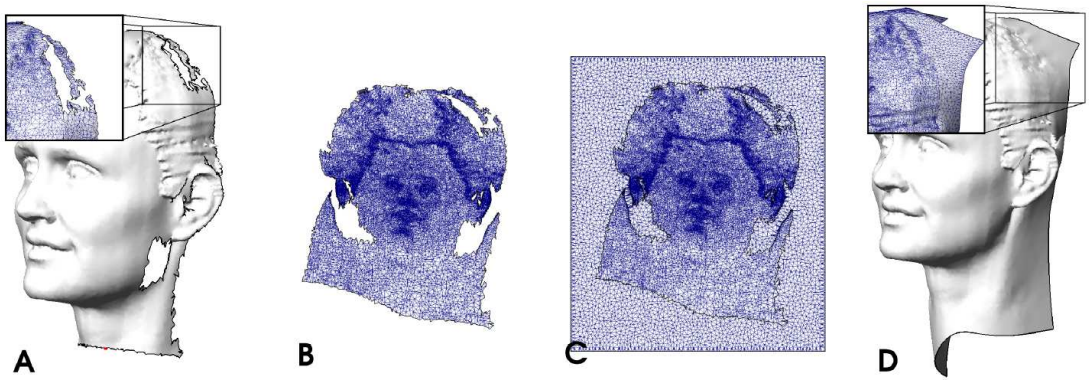
Comme une paramétrisation d'une surface mets celle-ci en correspondance avec un espace 2D plus simple, plusieurs algorithmes peuvent bénéficier de cette représentation. Dans ce chapitre, nous présentons succintement différentes applications de ces méthodes à des problèmes de re-maillage, d'extrapolation de surfaces et de simulation numérique de la lumière. Dans le cadre de ce mémoire, nous n'entrerons pas dans le détail de ces applications, pour plutôt nous concentrer sur les aspects directement liés à la géométrie numérique.



3.4 Re-maillage

Le travail de cette section a été réalisé en coopération avec Pierre Alliez (Inria Sophia Antipolis) et Mathieu Desbrun (Caltech). Nous avons publié les résultats à la conférence ACM Siggraph [ACSD+03]. Nous avons proposé une nouvelle technique de re-maillage polygonale, exploitant une spécificité importante des surfaces, à savoir l'anisotropie intrinsèque présentée par les objets (aussi bien naturels que fabriqués par l'homme). En particulier, nous utilisons les directions de courbure principales pour piloter le processus de re-maillage. Ceci revient à imiter les lignes qu'un artiste déssinerait pour créer l'objet 3D ex-nihilo. Après l'extraction et le lissage du champ de courbures, les lignes de courbures minimales et maximales sont utilisées par déterminer les arêtes du re-maillage dans les régions anisotropes, alors que les régions sphériques (isotropes) sont re-maillées par distribution de points, puisqu'aucune direction préférentielle n'existe dans ces zones. En conséquence, notre technique génère des maillages polygonaux constitués principalement de quadrilatères dans les régions anisotropes et de triangles dans les régions sphériques.

La figure ci-desous montre notre méthode appliquée à plusieurs jeux de données. A-H: le jeu de données "octa-flower" montre le comportement de notre algorithme sur des surfaces lisses par morceaux. Les directions de courbure principales sont estimées et lissées par morceaux (C). I: la tête du lapin est re-maillée en utilisant différentes densités de maillage. J: Le David de Michelange est re-maillé; les vues rapprochées sur l'oeil et sur l'oreille montrent la complexité du modèle, et montrent la manière dont les lignes de courbure s'adaptent aux structures locales.



3.5 Extrapolation de surfaces

Le lissage de maillages a été ces dernières années un sujet de recherches très actifs. Les approches existantes requièrent de fixer le bord de la surface, ou se limitent à des surfaces fermées. J'ai proposé une nouvelle approche, publiée à la conférence ACM Siggraph [Lé03]. Mon approche DDE (Dual Domain Extrapolation) offre la propriété de calculer des frontières naturelles. Ceci permet non seulement de lisser des surfaces existantes, mais également d'extrapoler leur géométrie au delà des frontières initiales. DDE se fonde sur une paramétrisation globale de la surface combinée à la minimisation des courbures principales, discrétisées sur les arêtes de la surface.

L'utilisation d'une paramétrisation globale permet de totalement découpler le critère géométrique (outer fairness) du critère de qualité de maillage (inner fairness). De plus, l'introduction de cet espace paramétrique permet d'imaginer de nouvelles manières pour contrôler la forme de la surface. Utilisée comme lisseur, notre approche calcule un maillage lisse conforme-discrèt au maillage initial. Ceci permet de lisser des maillages texturés sans introduire de déformation de textures. Notre approche est décrite sur la figure: à cause de problèmes d'ombres, les maillages scannés présentent souvent des trous complexes et des bords irréguliers (Figure A). Nous calculons une paramétrisation globale de la surface (Figure B). Extrapoler les bords devient alors un problème 2D (Figure C). La position en 3D des nouveaux sommets est alors calculée par la minimisation d'une fonction objectif (Figure D). La surface calculée par notre approche est une approximation d'une surface dite d'*énergie minimale* (MES), définie comme un point critique de la fonctionelle d'{energie suivante:

$$E_{MES} \quad = \quad \int_{\Omega} \kappa_{min}^2 + \kappa_{max}^2 \, dudv$$

Nous proposons une version discrète de cette énergie, définie par:

$$E_{MES} \simeq F(x) = \frac{1}{6} \sum_{e \in \mathscr{E}} \mathscr{A}(T) + \mathscr{A}(T') \left\| \frac{J_T.\mathbf{w}_2(e)}{\|J_T.\mathbf{w}_2(e)\|} - \frac{J_{T'}.\mathbf{w}_2(e)}{\|J_{T'}.\mathbf{w}_2(e)\|} \right\|^2$$

$$e = (i,j) \quad ; \quad \mathbf{w}_2(e) = \begin{bmatrix} v_i - v_j \\ u_j - u_i \end{bmatrix}$$

où $J(\delta)$ dénote la matrice Jacobienne de $\mathbf{x}(.,.)$ (i.e. la matrice de la différentielle $d\mathbf{x}$) au point $\mathbf{u} + \delta.\mathbf{w}$, donnée par:

$$J \quad = \quad \begin{pmatrix} \partial x/\partial u & \partial y/\partial u & \partial z/\partial u \\ \partial x/\partial v & \partial y/\partial v & \partial z/\partial v \end{pmatrix}^t \tag{3.39}$$

Pour minimiser cette fonctionelle non-linéaire, nous avons expérimentés différents algorithmes, à savoir la méthode de Newton, et les algorithmes BFGS et SQP.

3.6 Simulation numérique de la lumière

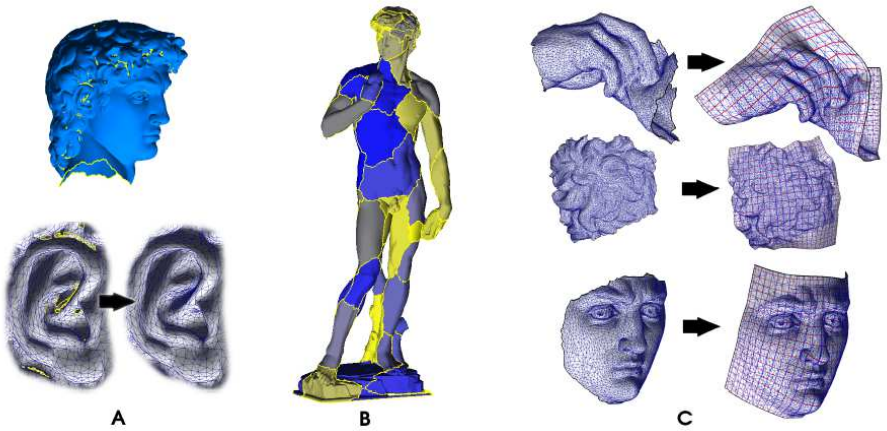
La simulation numérique de l'illumination globale est un processus impliquant de lourds calculs. La complexité de ce problème revêt deux principaux aspects:

- **complexité géométrique:** Les modèles finements maillés augmentent fortement les temps de calcul;
- **complexité de l'éclairage:** Les ombres franches et les variations d'illumination de faible échelle sont difficiles à représenter.

Pour ces deux raisons, les méthodes par éléments finis se comportent mal quand elles sont appliquées à des modèles finements maillés, à cause du fort degré de couplage que ces modèles introduisent entre la géométrie et l'illumination.

Nous avons introduit la méthode PME (Parametric Master Element), permettant d'appliquer les méthodes par éléments finis à des modèles finement maillés. Cette méthode a été développée par Gregory Lecot (doctorant co-encadré par Jean-Claude Paul et moi-ême), et a fait l'objet d'une publication à la conférence Graphite [LLAP05].

Nous utilisons une paramétrisation pour découpler la représentation de l'énergie lumineuse de celle utilisée par la géométrie de la scène. Ces deux représentations peuvent être considérées à différentes échelles, et ce de manière continue et indépendente l'une de l'autre. Ceci permet à la fois d'optimiser le calcul des échanges énergétiques à large échelle et de capturer les détails d'illumination à une échelle sub-facette. Pour construire notre représentation, nous commençons par restaurer la topologie du modèle (Figure A), puis nous décomposons le modèle en cartes homéomorphes à des disques (Figure B). Ensuite, chaque carte est paramétrisée et extrapolée (Figure C). L'équation de radiosité est alors résolue dans l'espace paramétrique.



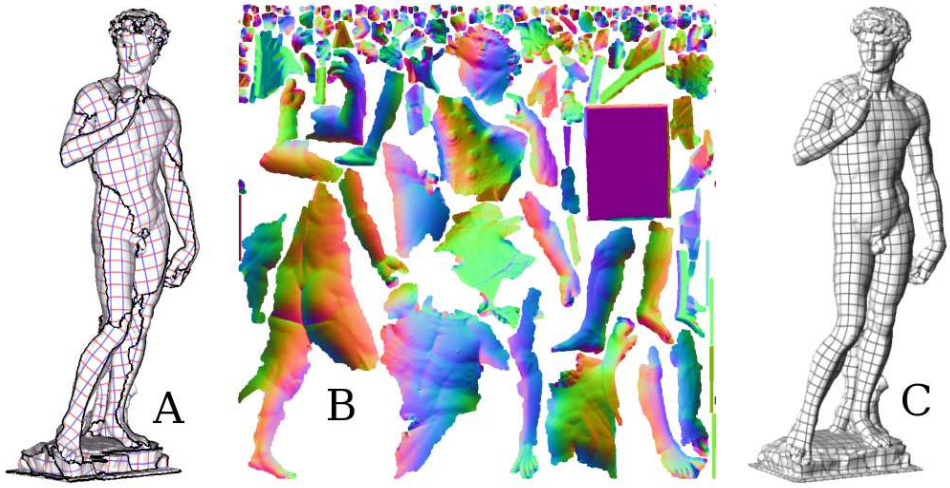


Figure 4.1: *A,B: Notre méthode ABF++ combinée avec un algorithme de segmentation permet de construire un atlas de texture. Toutefois, le grand nombre de discontinuités ainsi générées peut être problématique pour certains algorithmes. C: une paramétrisation globale ne présente pas ces problèmes. (Données: Digital Michelangelo Project, Stanford).*

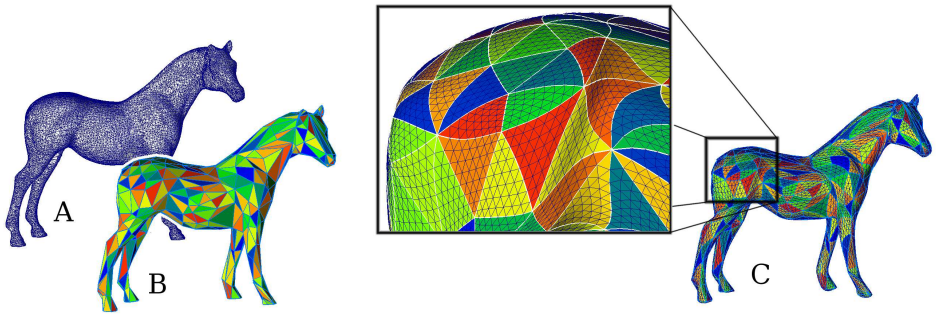


Figure 4.2: *La méthode MAPS et ses dérivées construit une paramétrisation globale en décomposant la surface initiale (A) en un ensemble de cartes triangulaires (B), puis en ré-échantillonant ces cartes de manière régulière (C).*

Chapitre 4

Paramétrisation Globale

Comme nous l'avons vu dans le Chapitre précédent, les méthodes de paramétrisation permettent relativement facilement de mettre en correspondance bijective une forme 3D avec un sous-ensemble de $\mathbb{R}^2$. Lorsque la forme a une topologie plus complexe qu'un disque, il est possible de décomposer la surface en un ensemble de cartes, homéomorphes à des disques, en utilisant un algorithme de segmentation[CSAD04]. Chacune de ces cartes est ensuite mise en correspondance avec un sous-ensemble de $\mathbb{R}^2$ (c.f. Figure 4.1-A,B). Même si cette solution fonctionne, elle n'est pas tout à fait satisfaisante pour l'esprit: pourquoi doit-on ainsi "abimer" la surface (en la découpant) alors que nous souhaitons seulement la munir d'un système de coordonnées ? Du point de vue applicatif, les frontières entre les cartes sont assez pénibles à gérer. Elles nécessitent des algorithmes plus complexes pour être prises en compte par les algorithmes de re-maillage (Section 3.4), et vont introduire des artéfacts génants dans les applications de plaquage de textures. Nous nous sommes donc intéressés à des algorithmes dits de paramétrisation *globale*, capable de calculer un système de coordonnées sur une surface de genre arbitraire, sans la découper (Figure 4.1-C).

Afin de calculer une telle paramétrisation globale pour un object de genre aritraire, la communauté du traitement numérique de la géométrie a tout d'abord étudié des méthodes opérant par segmentation, paramétrisation et ré-échantillonage de l'objet. La méthode MAPS[LSS+98] (Multiresolution Adaptive Paramétrisation of Surfaces, ou paramétrisation de surfaces multirésolution adaptative). Comme nous le montrons sur la Figure 4.2, cette méthode commence par partitionner l'objet initial (Figure 4.2-A) en un ensemble de cartes triangulaires appelé le *complexe de base* (Figure 4.2-B). Ensuite, une paramétrisation de chacun de ces triangles est calculée, puis l'objet est ré-échantilloné régulièrement dans l'espace paramétrique (Figure 4.2-C). Des raffinements de la méthodes ont ensuite amélioré la continuité d'une carte à l'autre[KLS03], formalisée par la notion de *fonction de transition*, que nous verrons dans la suite de ce chapitre. Cette représentation facilite grandement la mise en place d'une structure hiérarchique multi-résolution, ce qui a permis par exemple de définir des outils du traitement du signal pour les maillages[GSS99b].

Cette famille de méthode utilise un ensemble de triangles comme complexe de base. Pour certaines applications, telles que le plaquage de textures, ou encore l'approximation

[CSAD04] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *Computer Graphics (Siggraph)*. ACM, 2004.

[LSS+98] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. *Computer Graphics*, 32(Annual Conference Series):95–104, 1998.

[KLS03] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM TOG (SIGGRAPH)*, 2003.

[GSS99b] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. *Computer Graphics Proceedings (SIGGRAPH 99)*, pages 325–334, 1999.

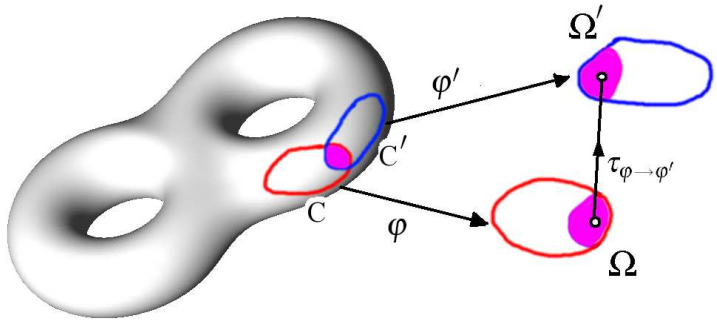


Figure 4.3: *Une paramétrisation globale (ou encore variété différentielle) est un ensemble de paramétrisations* $(\varphi, \varphi', \ldots)$ *connectées par des fonctions de transition* $(\tau_{\varphi\to\varphi'} \ldots)$.

à l'aide de Splines définies par produit tensoriel (Section 4.4), il peut être souhaitable d'utiliser un ensemble de quadrilatères comme complexe de base. Cette différence peut paraitre à première vue anecdotique, mais l'importance des applications concernées et la difficulté du problème ont fait qu'une grande activité de recherche a été récemment dévouée à ce seul problème. Une variation de la méthode MIPS[HG00], appliquée à un complexe de base formé de quadrilatères a récemment été proposée[THCM04]. Toutefois, cette méthode requiert l'interaction avec l'utilisateur pour construire le complexe de base.

La Section suivante introduit tout d'abord le formalisme nécessaire, à savoir la notion de variété différentielle. Ensuite, nous verrons quelques approches existantes, ainsi que nos contributions.

4.1 Variétés Différentielle

4.1.1 Notion de Variété Différentielle

Nous commençons par donner la définition d'une *variété différentielle* (également appelée paramétrisation globale dans notre contexte). Cette notion nous permet de définir une paramétrisation globalement lisse sur une surface de genre arbitraire, en combinant de multiples paramétrisations de cartes extraites de la surface, et connectées entre elles par des fonctions de transition. A notre connaissance, la notion de variété différentielle a été introduite pour la première fois dans le contexte de la géométrie numérique par Grimm et Hugues[GH95]. Plus récemment, une construction permettant d'obtenir des surfaces de classe C^∞ a été proposée[YZ04].

Etant donnée une surface **S**, nous considérons un ensemble de disques topologiques $\{\mathbf{C}\}$ (pouvant éventuellement se recouvrir), appelés des *cartes*, et un ensemble de fonctions $\{\varphi\}$ mettant chaque carte **C** en correspondance avec un domaine 2D Ω (c.f. Figure 4.3). Nous noterons θ, ϕ les coordonnées dans cet espace 2D[1]. L'ensemble de ces fonctions $\{\varphi\}$

[1] Nous utilisons θ et ϕ plutôt que les notations usuelles u, v ou s, t afin d'indiquer la nature périodique de ces

[HG00] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In P.-J. Laurent, P. Sablonniere, and L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.

[THCM04] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. *ACM TOG (SIGGRAPH)*, 2004.

[GH95] C. Grimm and J.F. Hugues. Modeling surfaces of arbitrary topology using manifolds. In *SIGGRAPH conference proceedings*, 1995.

[YZ04] L. Ying and D. Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM TOG (SIGGRAPH)*, 2004.

est appelée une paramétrisation globale (ou variété différentielle) s'il satisfait les conditions suivantes: *Soient deux cartes* $\mathbf{C}$ *et* $\mathbf{C}'$*, si leur intersection* $\mathbf{C} \cap \mathbf{C}'$ *est un disque topologique, alors les images de leur intersection* $\mathbf{C} \cap \mathbf{C}'$ *dans l'espace paramétrique à travers* φ *et* φ' *sont liées par une transformation géométrique différentiable* $\tau_{\varphi \to \varphi'}$*:*

$$\forall \mathbf{p} \in \mathbf{C} \cap \mathbf{C}', \quad \varphi'(\mathbf{p}) = \tau_{\varphi \to \varphi'}(\varphi(\mathbf{p}))$$

Les fonctions $\tau_{\varphi \to \varphi'}$ sont appelées *fonctions de transition*[KLS03]). Une variété est dite *affine* si toutes les fonctions de transition sont des translations. Les variétés *complexes* admettent toute application conforme bijective (c.f. Section 3.3.3) comme fonction de transition (ceci inclut les similitudes, à savoir les composées de translations,rotations et homotéties). Le lecteur pourra se réferrer à la littérature en géométrie différentielle[Wei] pour une définition complète de ces classes d'objets. Alors que l'état de l'art se limite aux variétés affines[GY03a,GGT04], nous verrons plus loin, en Section 4.2, une méthode qui permet de construire une sous-classe de variétés complexes, admettant des degrés de liberté en translation et en rotation dans l'ensemble des fonctions de transition. Ces degrés de liberté supplémentaire permettent une plus grande flexibilité lors de l'alignement de la paramétrisation avec les champs de vecteurs. Avant de présenter notre approche, nous présentons les deux principales classes de méthodes permettant de construire des paramétrisations globales.

4.1.2 Calcul extérieur

En utilisant une représentation d'une variété sous la forme d'un ensemble de cartes (ainsi que les fonctions de transition associées), il est possible de manipuler des fonctions définies sur la variété, de calculer leurs gradients et de les intégrer sur des sous-ensembles de la variété. Toutefois, ces calculs impliquent souvent la matrice Jacobienne de chaque carte de manière non triviale et sont donc très fastidieux. Malgrès la possibilité d'utiliser des outils de calcul symboliques, cette manière de procéder reste peu satisfaisante pour l'esprit.

Le calcul extérieur représente une alternative plus élégante: il s'agit d'un calcul géométrique sans coordonnées, où les fonctions sont considérées comme des objets mathématiques abstraits, combinés par des opérateurs. A la fin, pour utiliser ces fonctions, il reste nécessaire de les instancier dans des systèmes de coordonnées. Toutefois, la plupart des calculs se trouvent simplifiés à l'aide de considérations de très haut niveau, guidées par la structure algébrique des fonctions et des opérateurs agissant sur elles.

Le calcul extérieur généralise le théorème fondamental de l'analyse, qui définit l'intégration d'une fonction:

$$\int_a^b f(x)dx = F(b) - F(a)$$

où F dénote la primitive de f.

Le théorème généralisé s'écrit de la manière suivante:

$$\int \Omega d\omega = \int_{\partial\Omega} \omega$$

coordonnées dans le cadre de notre méthode PGP.

[KLS03] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM TOG (SIGGRAPH)*, 2003.

[Wei] Eric W. Weisstein. Manifold.

[GY03a] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Symposium on Geometry Processing*. ACM, 2003.

[GGT04] S.J. Gortler, C. Gotsman, and D. Thurston. One-forms on meshes and applications to 3d mesh parameterization. Technical report, Harvard University, 2004.

Le théorème de la divergence (Ostrogradsky-Gauss) en est un cas particulier:

$$\int_{\mathrm{V}ol} \mathrm{div} K dv = \int_{\partial \mathrm{vol}} K \cdot N dS$$

où N dénote la normale au bord du volume. A noter que dans la version "calcul extérieur", le produit scalaire avec la normale est déjà pris en compte par l'intégration sur le bord $\partial\Omega$. D'une manière plus générale, le calcul extérieur considère de manière symétrique les entités intégrées (les formes différentielles) et les domaines d'intégration (appelés des chaines). L'intégration d'une forme sur une chaine est alors considérée comme un "produit scalaire" $< \omega, \Omega >$ combinant la forme ω et la chaine Ω. L'opérateur bord ∂ permet de calculer le bord $\partial\Omega$ d'une chaine Ω, à savoir une autre chaine de dimension inférieure. Avec cette notation, le théorème de Stokes s'écrit:

$$< d\omega, \Omega > = < \omega, \partial\Omega >$$

autrement dit, d'une manière informelle, la dérivée extérieure d devient l'opérateur bord ∂ quand on la "fait passer" de la forme intégrée ω vers le domaine d'intégration Ω. Autrement dit, la dérivée extérieure d est duale de l'opérateur bord ∂ (et réciproquement). Pour cette raison, les formes différentielles sont également appelées des *co-chaines* et la dérivée extérieure est appelée opérateur *co-bord*.

Notons finalement que nous retrouvons également le théorème fondamental de l'analyse comme un cas particulier:

$$\int_{(a,b)} f(x)dx = \int_{a,b} dF = \int_{a^- \cup b^+} F = F(b) - F(a)$$

C'est l'orientation du bord $\partial(a,b) = a^- \cup b^+$ qui introduit le signe "-" devant $F(a)$.

Il est également bien connu que la divergence et le gradient sont des opérateurs indépendant de tout système de coordonnées (bien que leur définition soit le plus souvent écrite à travers des coordonnées). Le calcul extérieur généralise le gradient par la dérivée extérieure d, et la divergence par la co-dérivée δ, les deux pouvant être définis indépendament de tout système de coordonnées.

A notre connaissance, à part la dualité de Hodge exploitée par Pinkall et Polthier pour calculer des surfaces minimales, l'une des premières utilisations du calcul extérieur par Gu *et. al* a utilisé pour calculer des paramétrisations globales les notions fondamentales impliquées dans la preuve récente de la conjecture de Poincaré . Plus récemment, en 2005, Schröeder et Desbrun dont donné à la conférence ACM Siggraph un cours sur ces notions, les rendant ainsi accessible à une communauté plus vaste.

Les entités de base manipulées par le calcul extérieur sont les *k-formes différentielles*. Intuitivement, une k-forme peut être considérée comme une fonction que l'on souhaite intégrer sur un domaine de dimension k. Ainsi, les 0-formes correspondent aux fonctions usuelles, les 1-formes à des fonctions que l'on souhaite intégrer sur des courbes, les 2-formes à des fonctions que l'on souhaite intégrer sur des surfaces etc... Intuitivement, les 1-formes peuvent être considérées comme des champs de vecteurs, et les 2-formes comme des champs de bi-vecteurs, à savoir des éléments différentiels d'aire.

L'espace des k-formes est structuré par plusieurs opérateurs, dont la dérivée extérieure d, qui permet de définir une $k+1$-forme à partir d'une k-forme. Appliquée à une 0-forme ω (à savoir une fonction), $d\omega$ correspond au gradient de ω.

4.1.3 Homologie et co-homologie

Comme nous avons pu le voir dans la section précédente, la notion de bord d'un domaine d'intégration joue un rôle particulièrement important dans le calcul extérieur, avec en particulier le théorème de Stokes que nous avons évoqué. Ceci met en évidence des liens

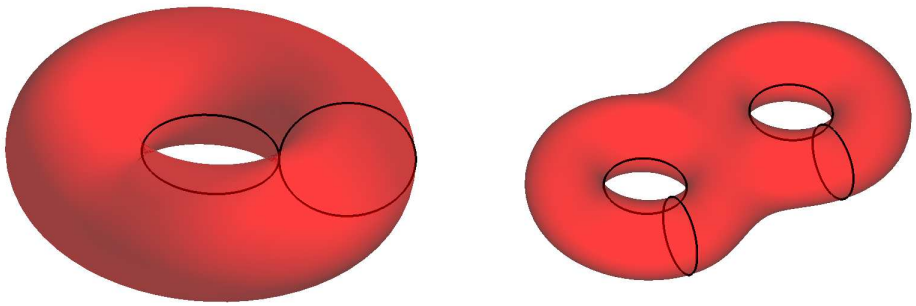


Figure 4.4: *Bases d'homologie d'un tore et d'un double tore.*

profonds avec la topologie de la surface, caractérisée par son groupe d'homologie. Cette notion permet de décomposer toute courbe de la surface comme la somme d'un ensemble de courbes fondamentales, appelée base d'homologie. La Figure 4.4 montre un exemple de base d'homologie pour un tore et pour un double tore. D'une manière plus générale, une base d'homologie d'un objet de genre g (autrement dit un objet à g anses) compte $2g$ éléments. Comme nous pouvons le voir sur la Figure, chaque anse ajoute 2 courbes fondamentales (l'une tournant autour de l'anse, et l'autre dans la direction perpendiculaire).

Nous avons mentionné dans la sous-section précédente la dualité entre les domaines d'intégration et les formes différentielle. Cette dualité s'applique également à la notion d'homologie (qui concerne les courbes, également appelées chaines), de laquelle une notion de co-homologie peut être dérivée (qui concerne les co-chaines, à savoir les formes différentielles). Pour saisir cette notion de co-homologie, nous avons besoin de deux nouvelles définitions:

- ◇ Une forme ω est dite *close* si $d\omega = 0$
- ◇ Une forme ω est dite *exacte* si elle est égale à la dérivée extérieure d'une autre forme ($\exists \sigma / \omega = d\sigma$);

Nous pouvons remarquer que les formes exactes sont également closes (puisque $dd\omega = 0 \forall \omega$). En nous fondant sur la dualité entre les formes (à savoir les objets à intégrer) et les chaines (à savoir les objets sur lesquels on intègre), nous pouvons mieux cerner la terminologie et les notations utilisées en calcul extérieur: l'opérateur bord ∂ est le dual de la dérivée extérieure d (c'est pourquoi son symbole ∂ évoque une dérivation). Une forme close ω (i.e. telle que $d\omega = 0$) est duale d'une chaine close Ω (i.e. telle que $\partial\Omega = 0$), ce qui explique le choix de l'adjectif "clos" pour qualifier cette propriété.

De plus, le bord d'une chaine est toujours clos ($\partial\partial\Omega = \oslash$), ce qui explique pourquoi la dérivée extérieure s'annule quand on l'applique deux fois ($dd\omega = 0$). En d'autres termes, le bord du bord est l'ensemble vide, donc par dualité, la dérivée de la dérivée s'annule.

Nous pouvons à présent aborder la notion de co-homologie. Sur une surface, on peut associer à chaque courbe de la base d'homologie un ensemble de 1-formes closes (par exemple des champs de vecteurs) qui sont équivalents du point de vue de la relation de co-homologie, qui se définit de la manière suivante: deux 1-formes ω_1 et ω_2 sont équivalentes si leur intégrale le long de tous les éléments de la bse d'homologie se correspondent. Nous pouvons remarquer que cela est vrai quand leur différence $\omega_2 - \omega_1$ est une forme exacte, puisque l'intégration de cette différence sur un élément Ω de la base d'homologie satisfait:

$$\exists \sigma / \omega_2 - \omega_1 = d\sigma \Rightarrow < \Omega, d\sigma > = < \partial\Omega, \sigma > = < \oslash, \sigma > = 0$$

(nous rappelons ici que les éléments de la base d'homologie sont des courbes fermées). Nous pouvons à présent donner la définition générale de la notion de co-homologie. C'est

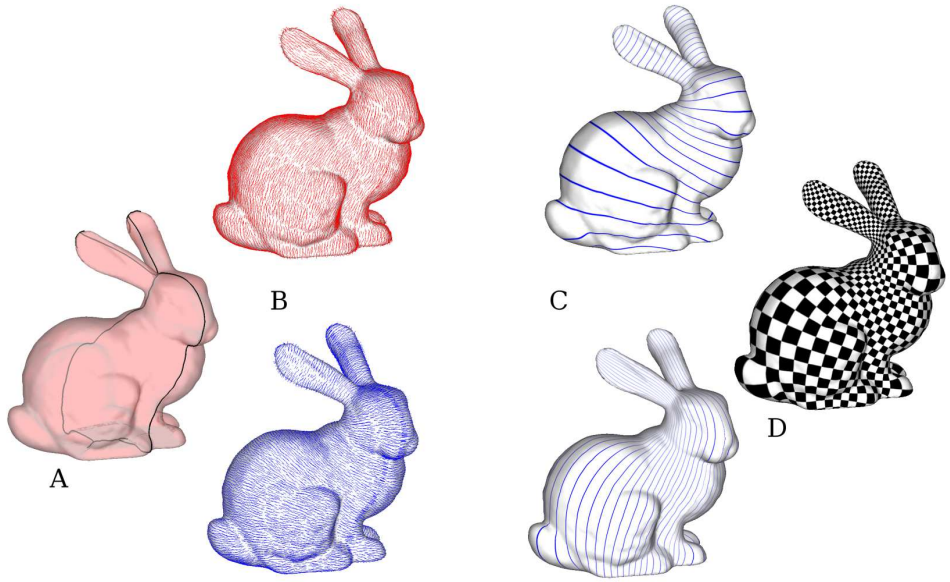


Figure 4.5: *La méthode développée par Gu et Yau pour construire une variété différentielle consiste tout d'abord à calculer une base d'homologie de la surface (A), puis à en déduire une base de co-homologie (B). Les "potentiels" u et v sont obtenus en intégrant les éléments de la base de co-homologie (C), qui définissent ensemble une fonction holomorphe (D).*

l'espace quotient des formes closes sur les formes exactes (à savoir, deux formes closes sont équivalentes si leur différence est une forme exacte). Cette relation est duale de l'homologie. Du point de vue de cette dernière, deux courbes fermées sont équivalentes si leur union définit le bord d'un sous-ensemble de la surface[2].

Comme nous le montrons sur la Figure 4.5, Gu et Yau ont utilisé ces notions pour construire des paramétrisations globales conformes sur des surfaces de genre arbitraire [GY03a, GY04]. Pour ce faire, ils calculent une fonction holomorphe, à savoir, la généralisation à des variétés de genre arbitraire des applications conformes. Nous avons déjà mentionnées ces dernières en Section 3.3.3. La méthode de Gu et Yau se fonde sur un théorème important, indiquant que chaque classe de co-gomologie contient une seule 1-forme harmonique. Nous avons déjà vu que les coordonnées réelle et imaginaire d'une application conforme définissent deux fonctions harmoniques. D'une manière similaire, une fonction holomorphe se compose de deux 1-formes harmoniques conjuguées (i.e. orthogonales). Ainsi, comme le montre la Figure 4.5, la méthode de Gu et Yau se déroule de la manière suivante: ils commencent par calculer une base d'homologie de la surface (en utilisant par exemple la méthode d'Erickson [EW05] ou encore celle de Lazarus [GTLH01]) (A), ils déduisent ensuite de cette base d'homologie une base de co-homologie et trouvent dans cette dernière une paire de 1-formes conjuguées (B). Finalement, ils intègrent ces 1-firmes afin de trouver la paramétrisation (C). Remarquons que pour un objet de genre g, la paramétrisation aura

[2]pour un ensemble de courbes fermées, définir le bord d'un sous-espace correspond au dual de la notion de forme exacte.

[GY04] X. Gu and S.-T. Yau. Optimal global conformal surface parameterization for visualization. In *Visualization conf. proc.* IEEE, 2004.

[EW05] Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[GTLH01] A. Guéziec, G. Taubin, F. Lazarus, and B. Horn. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):136–151, 2001.

nécessairement $2g$ singularités. Nous avons proposé avec Nicolas Ray une méthode pour trouver de manière optimale le placement de ces singularités [RLL$^+$06], et pour étudier leur généralisation à des indices fractionaires [LVRL06, RVL$^+$] (voir Figure 4.6).

En pratique, la méthode de Gu *et. al* opère en résolvant des systèmes linéaires avec trois types de contraintes: les 1-formes calculées doivent être harmoniques, closes et satisfaire une condition de dualité. La condition d'harmonicité est similaire à celles utilisées dans le cas de disques topologiques (c.f. Chapitre 2). Le fait qu'une 1-forme soit close se traduit par le fait que la somme des trois vecteurs s'appuyant sur les trois côtés d'un triangle soit égale à zéro (un champ de gradient est supposé annuler la divergence). La condition de dualité se substitue aux conditions aux limites utilisées dans le cas de disques topologiques, en imposant des valeurs fixes pour les intégrales de la 1-forme le long des éléments de la base d'homologie de la surface. En pratique, plusieurs sytèmes linéaires sont résolus, en contraignant ces intégrales à 1 pour l'une des courbes de la base d'homologie et 0 pour toutes les autres.

D'autres auteurs ont proposé des méthodes similaires, à la différence près qu'une paramétrisation est directement calculée (sans passer par une 1-forme ré-intégrée ensuite). Ces dernières méthodes se fondent sur une modification des conditions de Floater. Par exemple, Steiner et Fischer ont proposé de rendre l'objet équivalent à un disque topologique en utilisant un graphe de découpe (cut graph), et d'insérer des vecteurs de translations dans les conditions de Tutte correspondant aux sommets positionés sur le graphe de découpe [SF]. Tong *et.al* ont développé indépendament la même idée [TACSD06], en utilisant le formalisme du calcul extérieur que nous avons mentionné auparavent, et en ajoutant la possibilité de gérer des singularités d'indice fractionnaire.

En résumé, ces méthodes [GY02,TACSD06] créent implicitement une paramétrisation, en calculant des 1-formes différentielles, correspondant aux gradients de la paramétrisation cherchée. Au lieu d'associer des coordonnées (u,v) aux sommets, elles associent des vecteurs (du,dv) aux côtés du maillage, ce qui définit une 1-forme. Ces paramétrisations peuvent être converties en une forme spéciale de paramétrisation planaire, en fixant un sommet, et en parcourant récursivement les arêtes, en ajoutant à chaque fois les vecteurs (du,dv) aux coordonnées des sommets précédemment parcourus. Comme la 1-forme est close, ceci garantit que le résultat obtenu est indépendant de l'ordre du parcours effectué sur le maillage. Pour les surfaces de genre arbitraire, appliquer cette méthode va recouvrir le plan plusieurs fois. Ces paramétrisations sont continues presque partout, à l'exception d'un petit nombre de points singuliers. Nous en montrons quelques exemples sur la Figure 4.6. Ces singularités sont caractérisées par la notion d'indice, expliqué sur la même figure. Un théorème important (Poincaré-Hopf) indique que la somme des indices des singularités d'un champ de vecteur sur une surface correspond à la caractéristique d'Euler-Poincaré $\mathscr{X} = 2g-2$ de cette surface. Nous avons étudié avec Nicolas Ray, Bruno Vallet et Wan-Chiu Li une représentation discrète de ces champs de vecteurs, permettant de représenter des singularités d'indice arbitraire [RVL$^+$, LVRL06] (voir Figure 4.7). Afin de garder la longueur de ce mémoire dans des limites raisonnables, nous n'entrerons pas dans le détail de notre approche, et avons choisi de nous concentrer sur les aspects liés à la paramétrisation des surfaces.

[SF] D. Steiner and A. Fischer. Planar parameterization for closed manifold genus-g meshes using any type of positive weights.

[TACSD06] Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. Designing quadrangulations with discrete harmonic forms. In *ACM/EG Symposium on Geometry Processing*, 2006.

[GY02] X. Gu and S.-T. Yau. Computing conformal structures of surfaces. *Communications in Information and Systems*, 2(2):121–146, 2002.

[TACSD06] Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. Designing quadrangulations with discrete harmonic forms. In *ACM/EG Symposium on Geometry Processing*, 2006.

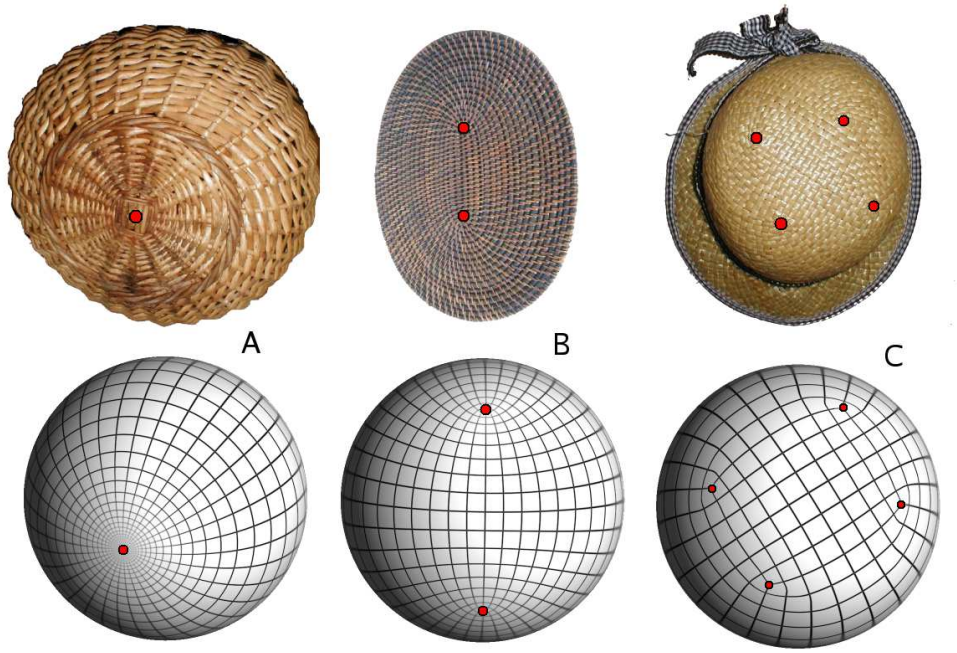


Figure 4.6: *Il est bien connu des mathématiciens et des artisans en vannerie que le maillage d'un objet de genre g génère* $2g$ *singularités, ou pôles. Ainsi, une sphère* ($g = 1$) *a deux pôles (A). Il est possible de subdiviser un pôle en deux demi-pôles (B) ou en quatre quart de pôles (C). L'odre de multiplicité d'un pôle, ou encore l'angle de rotation que fait le maillage autour du pôle est appelé l'*index *du pôle.*

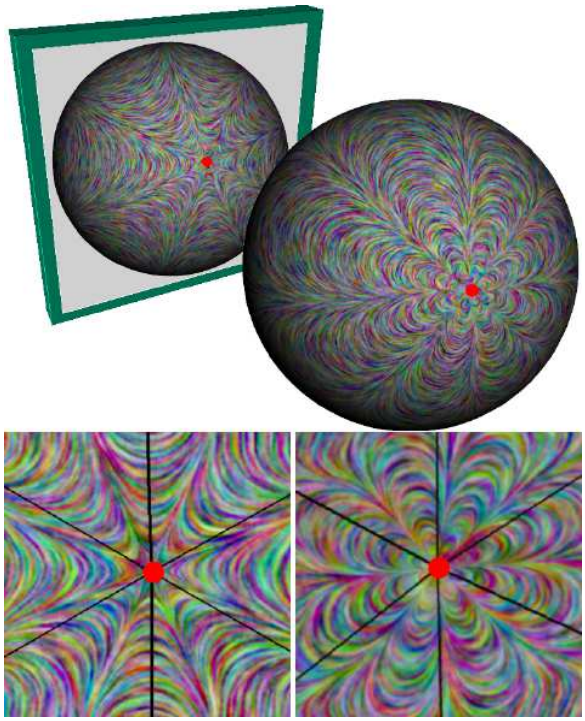


Figure 4.7: *En haut: une sphère avec seulement deux singularités d'ordre supérieur, d'indice +5 (devant) et -3 (derrière), la somme des deux correspondant à la caractéristique d'Euler-Poincaré (2 pour une sphère). En bas: un zoom sur le voisinage des singularités, qui montre que la nature non-linéaire de ces dernières est bien capturée par notre approche.*

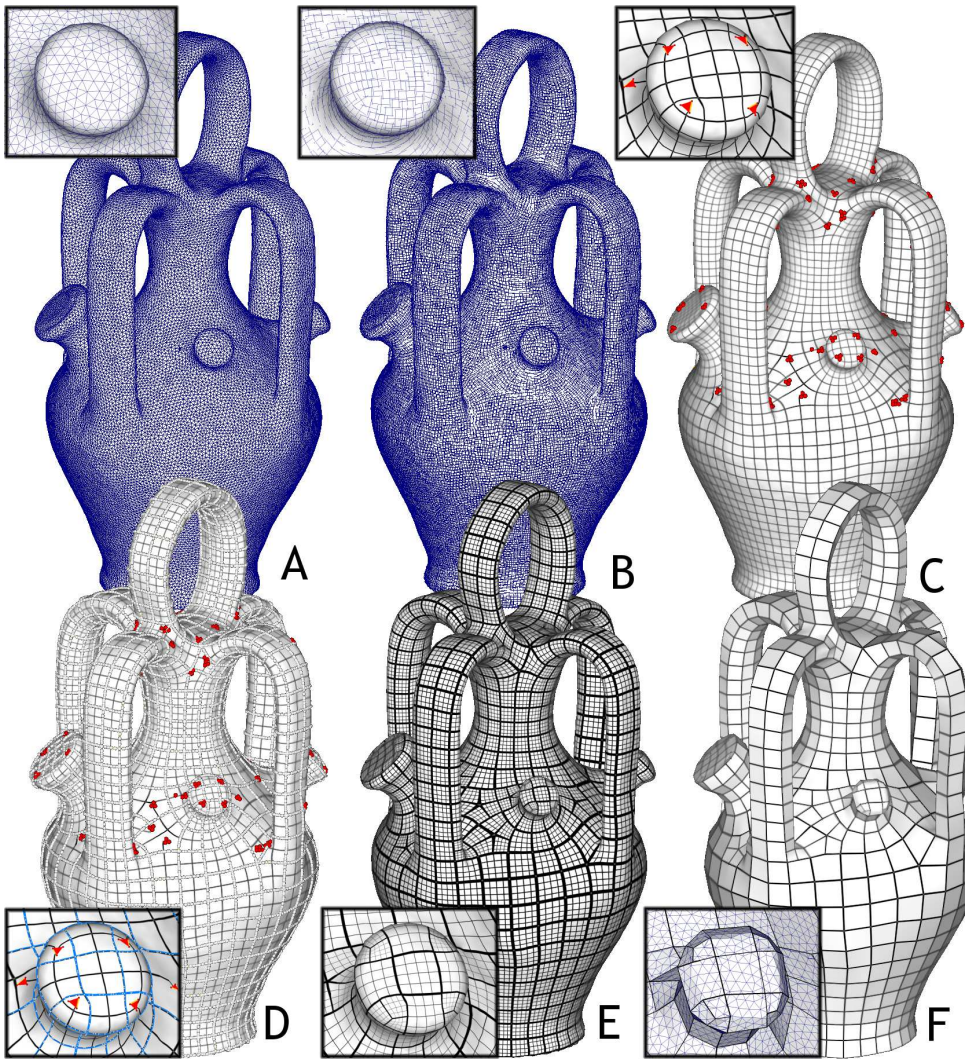


Figure 4.8: *Vue d'ensemble de notre algorithme. A: maillage initial; B: directions de courbure principales lissées ; C: courbes iso-$k\pi$ en θ et en ϕ. Les sommets, arêtes et triangles singuliers sont marqués en rouge; D: structure des cartes (extraite despuis les courbes iso-$2k\pi$); E: résultat final, obtenu en re-paramétrant les cartes contenant des singularités; F: visualisation du domaine paramétrique.*

4.2 Paramétrisation Globale Périodique

Le travail décrit dans cette section et dans la section suivante a été réalisé en coopération avec Nicolas Ray (chercheur de l'équipe), et Wan-Chiu Li (doctorant co-encadré par Jean-Claude Paul, Nicolas Ray et moi-même).

4.2.1 Variétés différentielles triangulées

Notre objectif est à présent de construire une paramétrisation globale, telle que les gradients $\nabla\theta$ et $\nabla\phi$ des paramètres θ, ϕ soient alignés avec deux champs de vecteurs donnés (par exemple, les directions principales de courbure). Nous commençons par étudier les cartes les plus élémentaires possibles, à savoir les triangles de la surface. Dans notre contexte de surfaces triangulées linéaires par morceaux, la paramétrisation globale est définie par les coordonnées θ_i^T, ϕ_i^T aux coins des triangles, où l'indice global i dénote un sommet, et T dénote un triangle. En exprimant la définition d'une variété différentielle associée à ces

cartes élémentaires, il sera possible de considérer des cartes plus générales, en assemblant des triangles dans l'espace des paramètres, comme nous le verrons en Section 4.2.7.

Nous considérons tout d'abord le cas d'une variété affine (à savoir, les fonctions de transition $\tau_{\varphi\to\varphi'}$ sont des translations). Nous verrons ensuite comment prendre en compte des degrés de liberté en rotation. Etant donnés deux triangles $T=(i,j,k)$ et $T'=(k,j,l)$ partageant k'arête (j,k), leurs paramètres (θ,ϕ) définissent une variété affine si:

$$\begin{pmatrix}\theta_j^T\\ \phi_j^T\end{pmatrix}-\begin{pmatrix}\theta_j^{T'}\\ \phi_j^{T'}\end{pmatrix} \quad = \quad \begin{pmatrix}\theta_k^T\\ \phi_k^T\end{pmatrix}-\begin{pmatrix}\theta_k^{T'}\\ \phi_k^{T'}\end{pmatrix} \tag{4.1}$$

Etant donnée cette contrainte, nous allons à présent définir une fonctionelle d'énergie F, dépendant de tous les paramètres (θ_i^T,ϕ_i^T) aux coins des triangles, et caractérisant l'alignement des gradients $(\nabla\theta,\nabla\phi)$ avec les directions principales de courbure. Dans notre formulation de la fonctionelle d'énergie F, au lieu d'exprimer l'équation 4.1 comme une contrainte, nous allons remplacer les variables (θ_i^T,ϕ_i^T) par des variables alternatives, associées aux sommets (plutôt qu'aux coins des triangles). Ces variables seront choisies de manière à satisfaire naturellement les contraintes. Nous verrons ensuite comment retrouver les paramètres (θ_i^T,ϕ_i^T) associés aux coins des triangles depuis les variables alternatives associées aux sommets.

Nous introduisons une restriction supplémentaire sur les fonctions de transition $\tau_{\varphi\to\varphi'}$: les coordonnées des vecteurs de translation connectant deux cartes doivent avoir des coordonnées multiples de 2π. Avec cette contrainte supplémentaire, nous avons:

$$\begin{pmatrix}\cos\theta_j^T\\ \sin\theta_j^T\end{pmatrix}=\begin{pmatrix}\cos(\theta_j^{T'}+2s\pi)\\ \sin(\theta_j^{T'}+2s\pi)\end{pmatrix}=\begin{pmatrix}\cos\theta_j^{T'}\\ \sin\theta_j^{T'}\end{pmatrix} \quad ; \quad \begin{pmatrix}\cos\phi_j^T\\ \sin\phi_j^T\end{pmatrix}=\begin{pmatrix}\cos\phi_j^{T'}\\ \sin\phi_j^{T'}\end{pmatrix} \tag{4.2}$$

(cette condition est également satisfaire au sommet k). En conséquence, et étant donné un sommet i, pour tous les triangles T incidents à i, les valeurs de $\cos\theta_i^T$ et $\sin\theta_i^T$ (resp. ϕ) se correspondent. Nous introduisons alors les variables $U_i=(\cos\theta_i^T,\sin\theta_i^T)$ et $V_i=(\cos\phi_i^T,\sin\phi_i^T)$, qui ne dépendent plus du triangle T et sont donc attachées aux sommets (plutôt qu'aux coins des triangles).

Nous considérons à présent le cas plus générale d'une sous-classe de variétés complexes, admettant pour fonctions de transition $\tau_{\varphi\to\varphi'}$ toute combinations de translations et de rotations. Les coordonnées des vecteurs de translation doivent être comme précédemment des multiples de 2π, et les angles de rotation doivent être des multiples de $\pi/2$. Nous appellerons *Paramétrisation Globale Périodique* une variété complexe satisfaisant ces contraintes. Sous ces condition, la condition de compatibilité entre deux triangles (Equation 4.2) devient:

$$\begin{aligned}&\exists r\in\{0,1,2,3\}\\ &\begin{pmatrix}\cos\theta_j^T\\ \sin\theta_j^T\end{pmatrix}=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}^r\begin{pmatrix}\cos\theta_j^{T'}\\ \sin\theta_j^{T'}\end{pmatrix} \quad ; \quad \begin{pmatrix}\cos\phi_j^T\\ \sin\phi_j^T\end{pmatrix}=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}^r\begin{pmatrix}\cos\phi_j^{T'}\\ \sin\phi_j^{T'}\end{pmatrix}\end{aligned} \tag{4.3}$$

En conséquence, étant donné un sommet i, pour tous les triangles T incidents a i, les valeurs de $\cos\theta_i^T$ et $\sin\theta_i^T$ (resp. ϕ) se correspondent, à un changement de signe et à un échange des coordonnées près.

Nous étudions à présent la manière d'exprimer l'alignement des gradients avec un champ de vecteur donné en fonction des variables (U_i,V_i) (Sections 4.2.2,4.2.3,4.2.4) et expliquons comment retrouver les paramètres (θ_i^T,ϕ_i^T) à partir de ces variables (Section 4.2.6). Nous verrons ensuite comment extraire une structure de cartes (Section 4.2.7), et comment retrouver une paramétrisation de chacune de ces cartes à partir des coordonnées (θ_i^T,ϕ_i^T) des triangles (Section 4.2.8).

4.2.2 Alignement avec les vecteurs de contrôle

Les données en entrée de notre algorithme sont deux champs de vecteurs *de contrôle* $\vec{K}$ et $\vec{K}^{\perp}$ définis sur la surface S, et un paramètre ω définissant la taille des cartes. Les champs de vecteurs de contrôle sont attachés aux sommets de la surface, et sont linéaires sur les triangles. La signification du paramètre ω et la manière de le choisir seront explicitées plus loin. Notre méthode a pour objectif de construire une variété complexe $\{\varphi^T\} = \{(\theta^T, \phi^T)\}$ telle que toute fonction φ^T associée au triangle T satisfasse:

$$\nabla\theta^T = \omega\vec{K} \quad ; \quad \nabla\phi^T = \omega\vec{K}^{\perp} \tag{4.4}$$

De plus, cette variété complexe doit être une paramétrisation globale périodique, à savoir les fonctions de transitions $\tau_{T\to T'}$ doivent être uniquement composées de translations multiples de 2π et de rotations multiples de $\pi/2$.

Les gradients de la paramétrisation ne sont pas supposés dépendre de la norme des vecteurs de contrôle. Comme notre objectif est de construire une paramétrisation *la plus isométrique possible*, nous normalisons les vecteurs de contrôle $\|\vec{K}\| = \|\vec{K}^{\perp}\| = 1$. Notre publication [RLL+06] explique également comment minimiser le rotationel du champ de vecteurs, ce qui permet de minimiser également le nombre de singularités. Ceci conduit a une paramétrisation qui n'est plus isométrique (mais qui reste conforme).

En prenant en compte cette normalisation, nous voyons que le paramètre ω contrôle la période des fonctions θ et ϕ. Comme nous le verrons en section 4.2.7, nous utiliserons les périodes d'amplitude 2π de la paramétrisation pour définir la structure des cartes. Ainsi, le paramètre ω détermine la taille des cartes. Dans tous nos exemples, nous avons utilisé pour ω dix fois la longueur moyenne des arêtes du maillage. A noter que si ω est trop grand, ceci pourrait générer des cartes qui ne sont plus homéomorphes à des disques. Nous pouvons facilement détecter de telles configuration en calculant la caractéristique d'Euler-Poincaré des cartes, et diminuer ω en conséquence si de telles configuration sont détectées.

Comme nous avons $\mathrm{rot}(\nabla\rho) = 0$ pour tout champ scalaire ρ, une solution de l'équation 4.4 n'existe que si $\mathrm{rot}(\vec{K}) = \mathrm{rot}(\vec{K}^{\perp}) = 0$[Nee94]. En général, les champ de vecteurs de contrôle ont un rotationel non-nul. Ainsi, nous devons nous tourner vers une formulation moins forte du problème, en termes de minimisation d'énergie:

$$F \quad = \quad \int_S \left(\|\nabla\theta^T - \omega\vec{K}\|^2 + \|\nabla\phi^T - \omega\vec{K}^{\perp}\|^2 \right) dS \tag{4.5}$$

Etant donné cette définition du problème, la difficulté principale est d'exprimer l'alignement des gradients de la paramétrisation avec les vecteurs de contrôle indépendament des degrés de liberté en translation et en rotation définissant la variété complexe. Nous commençons par introduire l'invariance en translation dans la formulation initiale (Section 4.2.3), puis raffinons cette formulation en introduisant les degrés de liberté en rotation (Section 4.2.4).

4.2.3 Invariance en translation

La principale difficulté dans la formulation du problème donnée par l'équation 4.5 est la nécessité de trouver une fonction *périodique* qui satisfasse le critère. Comme nous l'avons expliqué en Section 4.2, pour prendre en compte l'invariance en translation, nous proposons d'utiliser la périodicité modulo 2π des fonctions sinus et cosinus. Comme nous le montrons plus loin, il est alors possible de ré-exprimer l'alignement des gradients avec les vecteurs de contrôle en fonction de variables intermédiaires $U = (\cos\theta, \sin\theta)$ et $V = (\cos\phi, \sin\phi)$, correspondant aux sinus et cosinus des paramètres θ et ϕ. Les vecteurs U et V deviennent alors les inconnues de notre problème. Ainsi, nous obtiendrons une définition du minimiseur de notre fonctionelle d'énergie F sous la forme d'une fonction périodique.

[Nee94] Tristan Needham. *Visual Complex Analysis*. Oxford Press, 1994.

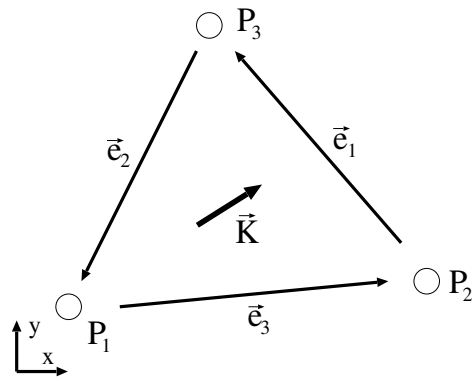


Figure 4.9: *Notations dans un triangle.*

Comme nous l'avons démontré dans notre article [RLL+06], nous pouvons minimiser la fonction F^* définie ci-dessous, qui admet le même minimiseur que la fonction F:

$$F^* = \sum_T \int_T \left(\|\nabla\theta^T - \omega\vec{K}_T\|^2 + \|\nabla\phi^T - \omega\vec{K}_T^\perp\|^2 \right) ds \tag{4.6}$$

où $\vec{K}_T$ et $\vec{K}_T^\perp$ dénotent la moyenne de $\vec{K}$ (resp. $\vec{K}^\perp$) sur le triangle T. Comme $\nabla\theta^T$ et $\nabla\phi^T$ sont constant sur chaque triangle, nous avons:

$$F^* = \sum_T \left(\|\nabla\theta^T - \omega\vec{K}_T\|^2 + \|\nabla\phi^T - \omega\vec{K}_T^\perp\|^2 \right) A_T \tag{4.7}$$

où A_T dénote l'aire du triangle T. Nous considérons à présent l'un des termes de cette somme:

$$F_T = \left(\|\nabla\theta^T - \omega\vec{K}_T\|^2 + \|\nabla\phi^T - \omega\vec{K}_T^\perp\|^2 \right) A_T \tag{4.8}$$

Comme il est difficile de directement introduire l'invariance en translation dans F_T, nous alons tout d'abord étudier $F_{T,i}^\theta$, l'énergie le long des côtés $\vec{e}_i$ de T (Figure 4.9) par rapport à θ. L'énergie $F_{T,i}^\phi$ par rapport à ϕ se calcule de manière similaire. Nous allons alors exprimer F_T comme une combinaison linéaire des $F_{T,i}^\theta$'s et des $F_{T,i}^\phi$.

Intuitivement, si nous considérons la différence le long du côté $\vec{e}_i$, nous devons considérer la différence entre la projection de $\vec{K}$ sur le côté et le gradient $\nabla\theta$ le long du côté. La projection est donnée par $\vec{K}\cdot\vec{e}_i/\|\vec{e}_i\|$, et le gradient le long du côté par $(\theta_{i\oplus 2} - \theta_{i\oplus 1})/\|\vec{e}_i\|$ où $i \in \{1,2,3\}$ dénote un indice local dans le triangle T (voir Figure 4.9) et $\oplus$ dénote l'addition modulo 3.

Nous définissons l'énergie le long des côtés:

$$F_{e_i}^\theta = \int_{\vec{e}_i} \left(\theta_{i\oplus 2} - \theta_{i\oplus 1} - \vec{K}\cdot\vec{e}_i \right)^2 / \|\vec{e}_i\|^2 ds \tag{4.9}$$

comme $\vec{K}$ et $\vec{K^\perp}$ sont linéaires le long des arêtes, nous pouvons remplacer $\vec{K}$ par $\vec{K}_i = (\vec{K}_{i\oplus 2} + \vec{K}_{i\oplus 1})/2$ et multiplier la fonction par $\|\vec{e}_i\|$ sans changer le minimiseur. La nouvelle fonctionelle d'énergie par côté est alors donnée par:

$$F_{T,i}^\theta = \left((\theta_{i\oplus 2} - \theta_{i\oplus 1}) - \omega\vec{K}_i\cdot\vec{e}_i \right)^2 \tag{4.10}$$

En utilisant cette nouvelle formulation de l'énergie, il est alors facile d'introduire l'invarance en translation, en remplaçant la différence par une différence indépendante d'une translation modulo 2π:

$$F_{T,i}^\theta = \min_s \left\{ ((2s\pi + \theta_{i\oplus 2} - \theta_{i\oplus 1}) - \omega\vec{K}_i\cdot\vec{e}_i)^2 \right\} \tag{4.11}$$

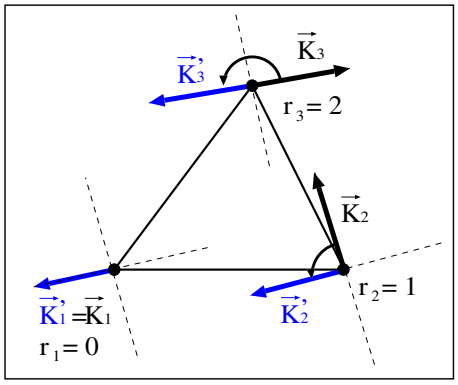


Figure 4.10: *Ré-orientation locale des vecteurs de contrôle.*

En réalisant un développement limité à l'ordre 1, nous obtenons une approximation de cette différence par la norme de la différence des vecteurs groupant les sinus et cosinus des paramètres. Nous obtenons ainsi:

$$F^{\theta}_{T,i} \simeq \left\| U_{i\oplus 2} - \begin{pmatrix} \cos(\omega \vec{K}_i \cdot \vec{e}_i) & -\sin(\omega \vec{K}_i \cdot \vec{e}_i) \\ \sin(\omega \vec{K}_i \cdot \vec{e}_i) & \cos(\omega \vec{K}_i \cdot \vec{e}_i) \end{pmatrix} U_{i\oplus 1} \right\|^2 \tag{4.12}$$

avec:

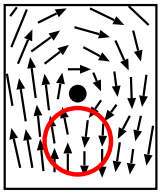
$$U_i = (\cos\theta_i, \sin\theta_i)$$

Comme nous pouvons le voir, en utilisant cette formulation, nous ne dépendons plus du coefficient en translation s (Equation 4.11).

A cette étape, nous pouvons nous contenter de minimiser la somme $\sum_{T,i\in 1,2,3}(F^{\theta}_{T,i} + F^{\phi}_{T,i})$, qui correspond à une version discrète de l'énergie, estimée sur les arêtes. Notre article [RLL+06] donne une meilleure estimation de cette énergie, fondée sur le calcul d'intégrales sur des triangles. Le résultat final donne une expression similaire, avec des coefficients supplémentaires pondérant chaque terme associé à une arête.

Cette formulation de F^* prend en compte l'invariance par translation. Nous montrons à présent comment introduire l'invariance en rotation dans la formulation.

4.2.4 Invariance en rotation



En général, il n'est pas possible de trouver une orientation *globale* d'un champ de vecteur consistante sur l'ensemble du domaine de définition (c.f. la région cerclée). Pour cette raison, nous modifions la définition de notre énergie (Equation 4.8) en permettant une ré-orientation *locale* des vecteurs de contrôle (Figure 4.10). Ces vecteur réorientés $\vec{K}_1$, $\vec{K}_2$ et $\vec{K}_3$ aux sommets du triangles (Figure 4.10) peuvent à présent être transformés par des rotations dont les angles sont multiples de $\pi/2$. Ainsi, $\vec{K}_2$ (resp. $\vec{K}_3$) se trouve aligné avec $\vec{K}_1$ en appliquant r_2 rotations d'angle $\pi/2$ (resp. r_3).

La rotation est appliquée de manière simultanée aux deux champs de vecteurs de contrôle $(\vec{K}_i, \vec{K}_i^{\perp})$ et aux inconnues $(\theta_i \quad \phi_i)$. Nous pouvons remarquer en particulier qu'une différence impaire des paramètres de rotation r_i le long d'un côté induit un échange des inconnues (i.e., un couplage entre des variables θ et des variables ϕ).

Pour définir la fonction objectif F_T sur les triangles, nous utilisons la même approche que dans la section précédente. Nous exprimons tout d'abord la déviation $F_{T,i}$ le long d'un côté, puis exprimons l'énergie sur un triangle F_T comme une combinaison linéaire des énergies des arêtes $F_{T,i}$. Comme les variables θ et les variables ϕ peuvent à présent être couplées par les équations, nous ne pouvons plus à présent les traiter de manière séparée.

En lui ajoutant l'invariance par rotation, l'Equation 4.11 devient:

$$F_{T,i} = \min_{s,t} \left\| \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_{i\oplus 2}} \begin{pmatrix} \theta_{i\oplus 2} \\ \phi_{i\oplus 2} \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_{i\oplus 1}} \begin{pmatrix} \theta_{i\oplus 1} + 2s\pi \\ \phi_{i\oplus 1} + 2t\pi \end{pmatrix} - \begin{pmatrix} \delta_i \\ \delta_i^{\perp} \end{pmatrix} \right\|^2$$

avec:

$$\begin{aligned} r_i &= \underset{r \in \{0,1,2,3\}}{\operatorname{argmax}} \left(\vec{K}_1 . \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^r \vec{K}_i \right) \; ; \; \vec{K}_i{}' = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_i} \vec{K}_i \\ \delta_i &= \omega/2 \left(\vec{K}'_{i\oplus 1} + \vec{K}'_{i\oplus 2} \right) \cdot \vec{e}_i \quad ; \quad \delta_i^{\perp} = \omega/2 \left(\vec{K}'^{\perp}_{i\oplus 1} + \vec{K}'^{\perp}_{i\oplus 2} \right) \cdot \vec{e}_i \end{aligned} \tag{4.13}$$

Comme dans la section précédente, afin de prendre en compte la périodicité des paramètres (θ, ϕ), nous résolvons l'équation en fonction des sinus et des cosinus de ces paramètres. L'énergie $F_{T,i}$ fonction de ces variables intermédiares devient alors (en utilisant la même approximation à l'ordre 1):

$$\begin{aligned} & F_{T,i} \simeq \\ & \left\| M^{r_{i\oplus 2}} X_{i\oplus 2} - \begin{pmatrix} \cos\delta_i & -\sin\delta_i & 0 & 0 \\ \sin\delta_i & \cos\delta_i & 0 & 0 \\ 0 & 0 & \cos\delta_i^{\perp} & -\sin\delta_i^{\perp} \\ 0 & 0 & \sin\delta_i^{\perp} & \cos\delta_i^{\perp} \end{pmatrix} M^{r_{i\oplus 1}} X_{i\oplus 1} \right\|^2 \\ & \text{avec:} \quad M = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad ; \quad X_i = \begin{pmatrix} U_i \\ \hline V_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \\ \cos\phi_i \\ \sin\phi_i \end{pmatrix} \end{aligned} \tag{4.14}$$

Comme dans la section précédente, nous pouvons simplement calculer le minimiseur de $F^* = \sum_{T,i\in 1,2,3}(F^{\theta}_{T,i} + F^{\phi}_{T,i})$, ce qui correspond à une version discrète de l'énergie, ou utiliser les poids qui permettent de calculer des intégrales sur les triangles.

Nous avons à présent:

$$F^* = \sum_T F_T = \sum_T \sum_{i=1}^{3} \lambda_i^T (F^{\theta}_{T,i} + F^{\phi}_{T,i}) \tag{4.15}$$

où $F^{\theta}_{T,i}, F^{\phi}_{T,i}$ sont donnés par l'Equation 4.14. Les termes λ_i^T peuvent être fixés à 1. Notre article [RLL+06] donne également une meilleure manière de les calculer, en intégrant l'énergie sur les triangles. La section suivante présente l'algorithme permettant de calculer le minimiseur de F^*.

4.2.5 Résolution numérique

Pour calculer le minimiseur de F^*, nous fixons l'un des sommets $U_1 = (1,0), V_1 = (1,0)$ et minimisons F^* par rapport à toutes les autres variables. Comme F^* est une forme quadratique, ceci revient à résoudre un système symétrique creux (c.f. Section 2.2.4). Pour de grands modèles, comportant plus 50K sommets, les normes des U_i, V_i décroissent rapidement quand nous nous éloignons du sommet fixé, ce qui cause à la fois des biais de pondération et des instabilité numériques. Pour résoudre ce problème, nous ajoutons un terme non-linéaire, contraignant la norme des vecteurs U, V:

$$F^{**} = F^* + \varepsilon \sum_i \left((\|U_i\|^2 - 1)^2 + (\|V_i\|^2 - 1)^2 \right)$$

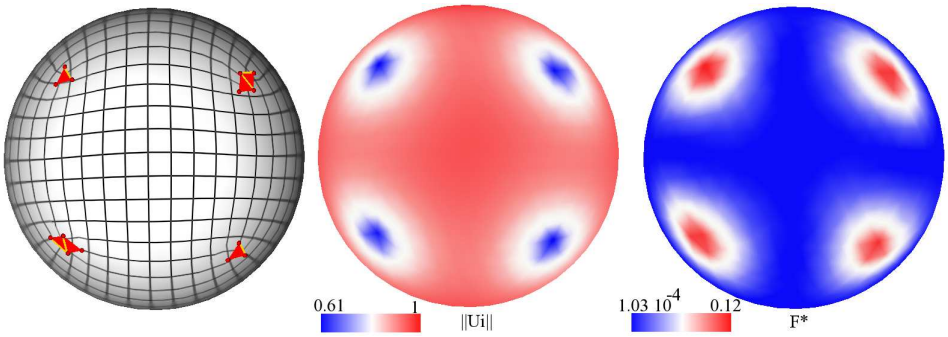


Figure 4.11: *Norme des vecteurs $\|U_i\|$ et fonctionelle d'énergie F^*. Comme nous pouvons le voir, les singularités sont caractérisées à la fois par une norme des U_i nulle (resp. V_i) et par une grande valeur d'énergie F^*.*

Nous utilisons l'algorithme de Newton (Section 2.2.5) pour minimiser cette énergie non-linéaire. Nous initialisons la méthode de Newton avec le résultat de la formulation linéaire. La convergence, à savoir $\nabla F^{**} < 10^{-6}$, a été atteinte pour moins de 6 itérations de Newton pour tous nos tests.

Comme nous le montrons sur la Figure 4.11, le terme de contrôle de la norme a une propriété intéressante. Les points singuliers sont caractérisés par des vecteurs U, V de norme nulle. Comme deux points singuliers proches l'un de l'autre augmentent considérablement l'énergie, cette énergie aura tendance à éviter de telles configuration et à les répartir uniformément sur toute la surface.

4.2.6 Reconstruction de la paramétrisation θ, ϕ

Le résultat de l'algorithme numérique décrit dans la section précédente est un ensemble de vecteurs U_i, V_i, correspondant aux sinus et aux cosinus des variables θ_i, ϕ_i qui définissent la paramétrisation globale. Pour retrouver les θ_i, ϕ_i à partir de ces variables, nous procédons de la manière suivante:

1. reconstruction de la paramétrisation (θ, ϕ) sur les cartes les plus simples possible, à savoir sur les triangles individuels (Section 4.2.6),
2. détection des sommets, arêtes et triangles singuliers (Section 4.2.6),
3. construction de la structure globale des cartes, fondée sur les périodes d'amplitude 2π des paramétrisations des triangles, et calcul des paramétrisation des cartes pour celles qui ne contiennent pas de singularités (Section 4.2.7),
4. re-paramétrisation des cartes qui contiennent des singularités (Section 4.2.8).

Paramétrisation des triangles individuels

Etant donnés les vecteurs U, V aux sommets d'un triangle $T = (i, j, k)$, trouver les coordonnées θ_i, ϕ_i (resp. j, k) revient à déterminer les entiers correspondant aux degrés de liberté en translation (s_i, t_i) et en rotation (r_i). Nous calculons explicitement les entiers r, s, t qui minimisent le terme d'énergie sur les arêtes (Equation 4.13) dans chaque triangle, de la manière suivante.

Pour déterminer la position globale et l'orientation d'un triangle dans l'espace paramétrique, nous positionnons tout d'abord les degrés de liberté entiers r_i, s_i, t_i du premier sommet i à $(0,0,0)$. Ainsi, les coordonnées θ_i^T, ϕ_i^T du premier sommet i sont données par $\theta_i^T = \text{angle}(U_i)$ et $\phi_i^T = \text{angle}(V_i)$ où $\text{angle}(U_i) = \text{sign}(U_{i,y})\arccos(U_{i,x}/\|U_i\|)$.

Nous pouvons alors trouver les coordonnées des deux autres sommets j et k en déterminant les différences s_e^T, t_e^T, r_e^T des degrés de liberté en translation et en rotation le long

de l'arête $e = (i, j)$ (resp. $(i,k),(j,k)$), donnés par $s_e^T = (s_j^T - s_i^T), t_e^T = (t_j^T - t_i^T)$ and $r_e^T = (r_j^T - r_i^T)$.

Nous considérons tout d'abord l'arête (i, j). Etant données les coordonnées (θ_i^T, ϕ_i^T) au sommet i et les vecteurs de contrôle $(K_i, K_i^\perp)$ et $(K_j, K_j^\perp)$ aux sommets i, j, l'Algorithm 9 calcule explicitement les différences s_e^T, t_e^T, r_e^T, puis les valeurs de θ_j^T et ϕ_j^T. Le calcul pour les deux autres côtés s'obtient par permutation circulaire des indices (i, j, k).

Algorithme 9 reconstruction de θ, ϕ le long d'une arête

propagation depuis i **vers** j **le long de** $e = (i, j)$:
// determiner et appliquer la rotation r_e

$$r_e^T \leftarrow \underset{r \in \{0,1,2,3\}}{\operatorname{argmax}_r} \left(\vec{K}_i \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^r \vec{K}_j \right)$$

$$\vec{K}_j \leftarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_e^T} \vec{K}_j \quad ; \quad \vec{K}_j^\perp \leftarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_e^T} \vec{K}_j^\perp$$

$$\begin{matrix} \theta_j^T \leftarrow \operatorname{angle}(U_j^T) \\ \phi_j^T \leftarrow \operatorname{angle}(V_j^T) \end{matrix} \quad ; \quad \begin{pmatrix} \theta_j^T \\ \hline \phi_j^T \end{pmatrix} \leftarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_e^T} \begin{pmatrix} \theta_j^T \\ \hline \phi_j^T \end{pmatrix}$$

// determiner et appliquer les translations s, t
$\vec{n} \leftarrow \vec{e}/\|\vec{e}\|$
$s_e^T \leftarrow \operatorname{argmin}_s \left| \theta_i^T - (\pi/\omega)\vec{n} \cdot (\vec{K}_i + \vec{K}_j) - \theta_j^T + 2s\pi \right|$
$t_e^T \leftarrow \operatorname{argmin}_t \left| \phi_i^T - (\pi/\omega)\vec{n} \cdot (\vec{K}_i^\perp + \vec{K}_j^\perp) - \phi_j^T + 2t\pi \right|$
$\theta_j^T \leftarrow \theta_j^T + 2s_e^T\pi \quad ; \quad \phi_j^T \leftarrow \phi_j^T + 2t_e^T\pi$

Cet algorithme calcule tout d'abord si les vecteurs de contrôle subissent une rotation le long de l'arête. Dans ce cas, nous devons calculer la manière dont cette rotation affecte la correspondance entre θ, ϕ and U, V. Par exemple, une rotation d'angle $\pi/2$ échange U et V. Dans ce cas, θ devient une fonction de V et ϕ une fonction de $-U$. Nous déterminons ensuite les différence s_e^T, t_e^T de degrés de liberté en translation, en minimisant explicitement l'énergie des arêtes.

Caractérisation des sommets, arêtes et triangles singuliers

Une fois que nous avons reconstruit la paramétrisation séparément dans chaque triangle, nous devons vérifier si ces triangles peuvent être assemblés dans l'espace paramétrique de manière à fournir une paramétrisation valide. Nous savons déjà que la solution de la version continue de cette équation présente des singularités, où les dérivées de la solution s'annulent. Dans notre contexte discret, ces singularités aparaissent sur des sommets, arêtes et triangles, qui violent alors les conditions assurant une paramétrisation valide. Ces conditions sont décrites dans la section 3.3.6, traitant de la méthode ABF[SdS01]):

- ◇ **Sommets singuliers:** un sommet v est singulier si la somme des angles incidents à v n'est pas égale à 2π. En particulier, un sommet singulier v peut également être caractérisé par le fait que l'algorithme 9 appliqué au voisinage de v construit un chemin ouvert.
- ◇ **Arêtes singulières:** une arête $e = (i, j)$ est singulière si sa longueur dans l'espace paramétrique ne correspond pas à celle de l'arête opposée $e' = (j, i)$. Ceci peut arriver si

[SdS01] Alla Sheffer and Eric de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17:326–337, 2001.

l'un des triangles adjacents à e est singulier.

◇ **Triangles singuliers:** un triangle T est singulier si l'application de l'algorithm 9 à ses trois côtés crée un chemin ouvert, ou un triangle d'aire signée négative.

Nous testons ces conditions explicitement. La Figures 4.8 montre des exemples de singularités.

4.2.7 Structure globale des cartes

Une fois que nous avons calculé la paramétrisation locale de chaque triangle T, nous construisons la structure globale de cartes. Dans notre approche, les frontières des cartes sont définies comme les courbes iso-$2k\pi$ des paramètres θ et ϕ. Ceci définit un ensemble de segments dans chaque triangle. Nous montrons ci-dessous que l'ensemble de toutes les courbes iso-$2k\pi$ de θ et ϕ est invariant par application des fonctions de transition admissibles (translations multiples de 2π et rotations d'angle multiple de $\pi/2$). Par conséquent, les extrémités des segments calculées indépendament dans chaque triangle se correspondent le long des arêtes non singulières de la triangulation, et ces segment forment un ensemble de lignes polygonale continues.

◇ **invariance de l'ensemble des iso-courbes par translation :** si un triangle T est traversé par une courbe iso-$2k\pi$ de θ (resp. ϕ), ce triangle translaté de $2s\pi$ est également traversé au même endroit par une courbe iso-$2(k+s)\pi$ de θ (resp. ϕ).

◇ **invariance of the set of iso-lines under valid rotations :** si un triangle T est traversé par une courbe iso-$2k\pi$ de θ (resp. ϕ), ce triangle tourné de $\pi/2$ est traversé au même endroit par une courbe iso-$2(-k)\pi$ de ϕ (resp. courbe iso-$2k\pi$ de θ). La même argumentation s'applique à des rotation dont l'angle est un multiple de $\pi/2$.

Nous remarquons qu'étant donnée une paire de triangles adjacents T_1 et T_2 se partageant l'arête non-singulière $e = (i, j)$, la fonction de transition $\tau_{T_1 \to T_2}$ connectant les paramétrisations locales de ces deux triangles se calcule de la manière suivante:

$$\begin{aligned} \tau_s &= s_i^{T_1} - s_i^{T_2} \\ \tau_t &= t_i^{T_1} - t_i^{T_2} \\ \tau_r &= r_e^{T_1} - r_e^{T_2} \\ \tau(p) &= R^{\tau_r} p + (\tau_s, \tau_t) \end{aligned} \tag{4.16}$$

où R dénote la rotation d'angle $\pi/2$. Ainsi, les fonctions de transition préservent globalement l'ensemble des courbes iso-θ et iso-ϕ, formées d'un ensemble de segments sont les extrémités se correspondent.

L'Algorithme 10 calcule les frontières de cartes. Cette algorithme se fonde sur les structures de données d'ensembles simpliciaux à plongement hiérarchique que j'ai développées [DLM00], [LLP05].

Chaque triangle contient une liste de segment, et chaque arête une liste d'extrémités de segments. L'algorithme calcule les segments individuels définis par l'intersection de chaque triangle avec les courbes ($\theta = 2k\pi$) et ($\phi = 2k\pi$) lines. Pour chaque extrémité de segment, nous calculons les coordonnées 3D et les paramètres 2D. L'algorithme connecte ensuite dans chaque arête les extrémités de segments qui se correspondent, et calcule dans chaque triangle les intersections entre segments. Ensuite, les segments "pendants" sont supprimés (à savoir, tous les sommets de valence 1 sont "grignotés" jusqu'à trouver un sommet de valence supérieure à 2).

La paramétrisation dans les cartes qui ne contiennent pas de singularité peut alors facilement être retrouvée, par un algorithme "glouton" qui assemble les triangles dans l'espace paramétrique [SdS01]. Les cartes restantes contiennent des singularités, et requièrent un traitement spécifique afin de générer une paramétrisation valide. Comme nous le montrons dans la Section suivante, ces cartes sont subdivisées et re-paramétrées. A cette étape,

Algorithme 10 Calcul des frontières de cartes

calcul des frontieres de cartes:
pour chaque triangle T
  si T est non-singulier
    pour $k \in \mathbb{N}$ tel que $2k\pi \in [\min_T(\theta), \mathrm{Max}_T(\theta)]$
      Ligne $l \leftarrow$ ligne d'equation $(\theta = 2k\pi)$
      Segment $S \leftarrow l \cap T$ $//dans\ l'espace\ parametrique$
      stocker S dans T
      stocker les extremites de S dans les aretes de T
    fin$//pour$
    pour $k \in \mathbb{N}$ tel que $2k\pi \in [\min_T(\phi), \mathrm{Max}_T(\phi)]$
      Ligne $l \leftarrow$ ligne d'equation $(\phi = 2k\pi)$
      Segment $S \leftarrow l \cap T//dans\ l'espace\ parametrique$
      stocker S in T
      stocker les extremites de S dans les aretes de T
    fin$//pour$
  fin$//si$
fin$//pour$
pour chaque arete e
  connecter les extremitees stockees dans e et ayant la meme position en 3D
fin$//pour$
pour chaque triangle T
  calculer les intersections entre les cotes stockes dans T
fin$//pour$
enlever recursivement les aretes pendantes

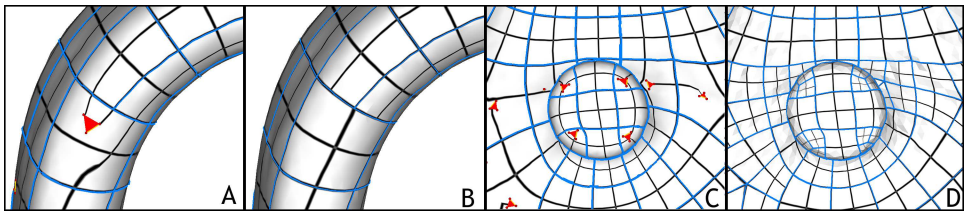


Figure 4.12: *Re-paramétrisation des cartes contenant des singularités. A,B: les cartes comportant quatre coins sont re-paramétrées en utilisant la méthode "mean value coordinates" de Floater. C,D: les cartes comportant un nombre arbitraire de coins sont subdivisées en cartes quadrilatères, et celles-ci sont re-paramétrées.*

l'utilisateur peut éventuellement modifier de manière interactive les frontières de cartes, à l'aide de notre structure de données [LLP05].

4.2.8 Re-paramétrisation des cartes comportant des singularités

Dans les cartes comportant des singularités, il est impossible d'obtenir une paramétrisation valide en assemblant simplement les paramétrisations individuelles des triangles dans l'espace paramétrique. Dans la plupart des cas, il serait possible de subdiviser les cartes le long des séparatrices de la singularité. Toutefois, l'algorithme correspondant est assez délicat à implanter, sensible aux instabilités numérique, et susceptible d'échouer pour des configurations particulières des singularités. Nous avons donc choisi une approche moins élégante, mais plus pragmatique:

◇ Les cartes comportant quatre coins sont re-paramétrées, en utilisant la méthode "mean value coordinates" de Floater [Flo03] (Figure 4.12 A et B). La paramétrisation des som-

[Flo03] M. S. Floater. Mean value coordinates. *CAGD*, 20:19–27, 2003.

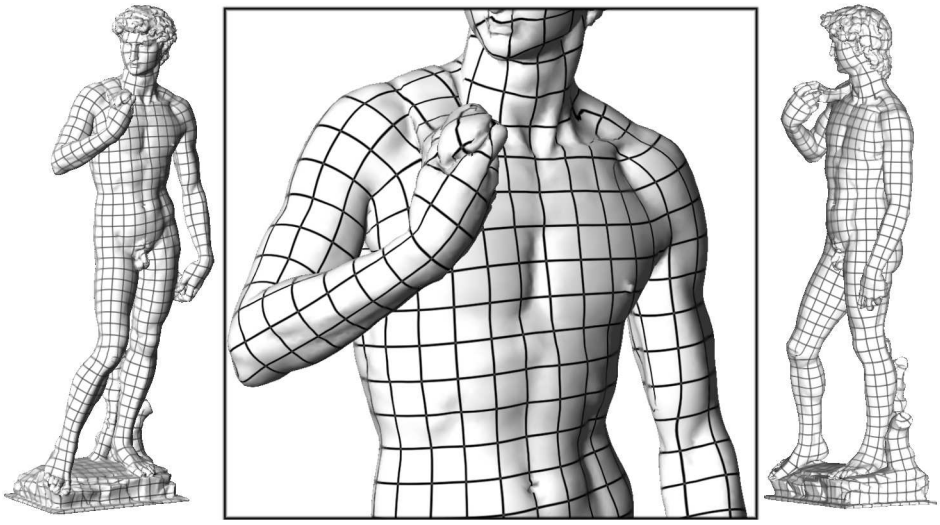


Figure 4.13: *Paramétrisation globale du David de Michelange (digitalisation fournie par le "Digital Michelangelo Project" de l'université de Stanford). La figure montre les courbes iso $k\pi$.*

mets du bord est ajustée afin d'assurer la continuité C^0 le long des frontières de la carte.

- ◇ Les cartes comportant un nombre arbitraire de coins sont subdivisées en cartes quadrilatères, en insérant un sommet au centre de la carte, et un sommet au milieu de chaque bord de la carte. (Figure 4.12 C et D). Les nouveaux sommets sont connectés par des géodésiques, et le maillage est découpé le long de ces géodésiques, en utilisant notre structure de données [LLP05]. Les cartes quadrilatères résultantes sont re-paramétrées (également par la méthode MVC de Floater).
- ◇ Finalement, pour améliorer la continuité au travers des frontières de cartes, nous utilisons une méthode de relaxation globale [KLS03,SPPH04].

Dans nos expériences, seule une petite fraction des cartes (entre 2 et 5 %) contenaient des singularités et ont nécessité ce traitement supplémentaire.

L'ensemble d'algorithme présentés dans cette section permet ainsi de calculer une paramétrisation globale d'un objet de genre arbitraire. Nous verrons dans les deux sections suivantes comment utiliser cette paramétrisation pour re-mailler une surface, ou encore pour convertir une surface triangulée en surface paramétrique.

4.3 Re-maillage implicite

Nous avons vu en section 3.4 un algorithme permettant de re-mailler une surface, en exprimant le problème du re-maillage dans l'espace paramétrique, comme l'intégration explicite des lignes de courbure principale. Nous allons étudier dans cette section une version implicite de cet algorithme, fondée sur notre approche de paramétrisation globale présentée dans la section précédente. L'avantage principal est une meilleure robustesse, et une plus grande régularité des maillages générés.

Un résultat de la théorie de l'approximation[d'A00] prédit que des performances max-

[KLS03] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM TOG (SIGGRAPH)*, 2003.

[SPPH04] J. Schreiner, A. Prakash, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM TOG (Proc. SIGGRAPH 2004)*, 2004.

[d'A00] E.-F. d'Azevedo. Are bilinear quadrilaterals better than linear triangles? *SIAM J. of Scientific Computing*, 22(1):198–217, 2000.

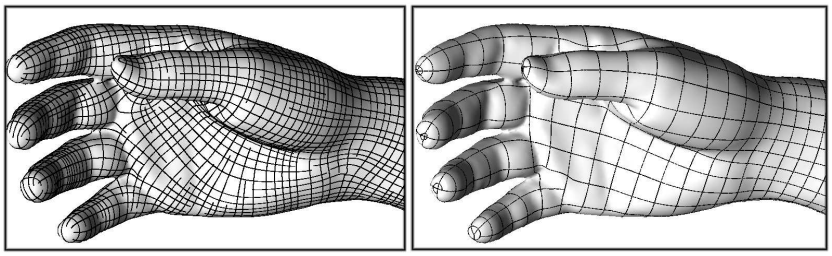


Figure 4.14: *A gauche: nos techniques de re-maillage explicite (c.f. Section 3.4) génèrent parfois un placement irrégulier des arêtes du maillage, et ne parvient pas toujours à les refermer. A droite: notre formulation implicite génère un re-maillage plus régulier.*

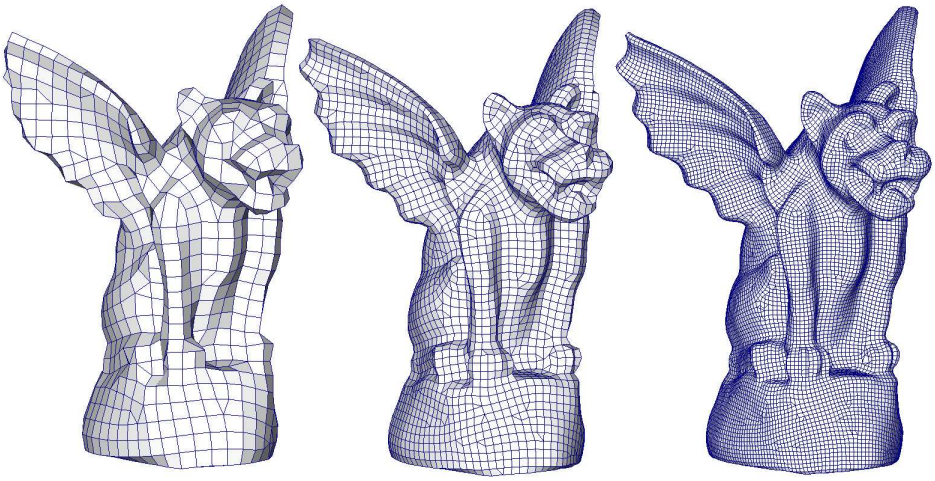


Figure 4.15: *Re-maillages quadrilatères d'une surface à trois résolutions différentes.*

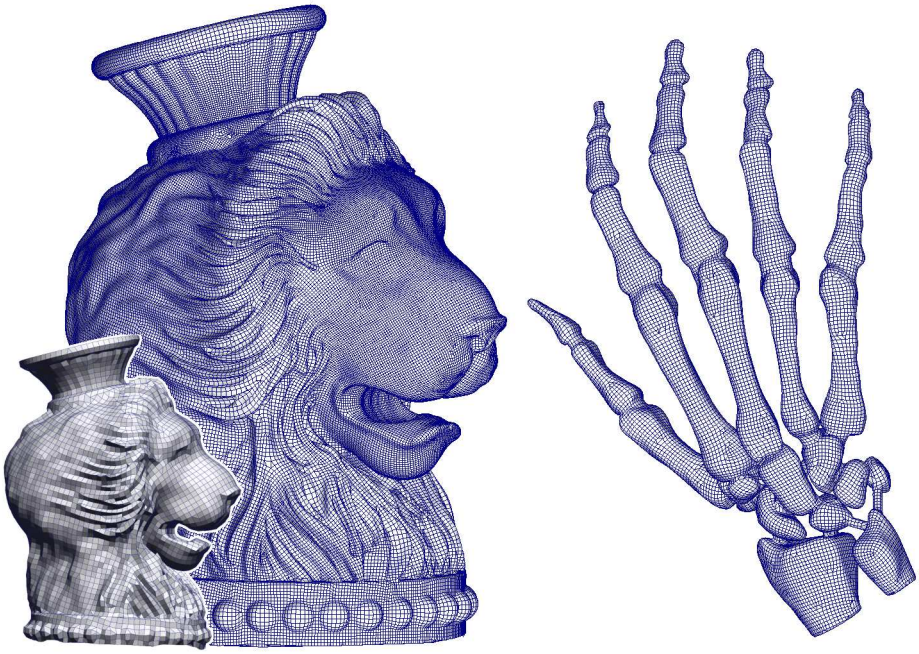


Figure 4.16: *Re-maillages quadrilatères. A gauche: un maillage ayant une forte densité de détails; A droite: un maillage présentant de petits détails, susceptibles dêtre ignorés par les approches explicites (Runge-Kutta).*

imales en termes d'approximation seront obtenues lorsque les éléments du maillage sont alignés avec les axes de courbure principale. Nous avons exploité cette idée en coopération avec Pierre Alliez, comme nous l'avons expliqué en section 3.4. Notre méthode consistait à placer un ensemble de lignes de courant tangentes aux directions principales de courbures. Ces lignes de courant sont caractérisées par une équation différentielle ordinaite, dont la solution peut être calulée numériquement en utilisant un schéma d'intégration explicite, tel que la méthode de Runge-Kutta. Cette intégration peut être réalisée dans l'espace paramétrique, comme nous l'avons initialement proposé. Des travaux plus récents[MK04] réalisent ce calcul directement sur la surface initiale, sans passer par une paramétrisation. Toutefois, la principale difficulté avec ce type d'approche reste le placement des lignes de courant sur la surface, de manière à définir un maillage le plus régulier possible. Les approches existantes, fondées sur une variante d'algorithmes classiques en visualisation de dynamique des fluides[JL97] ne parviennent pas toujours à construire un maillage régulier. Ainsi, la Figure 4.14-Gauche montre de fortes variations dans la tailles des quadrilatères.

La méthode de Dong *et. al*[DKG04] est une tentative de schéma *implicite*, et partage plusieurs points communs avec notre approche. Cette methode construit une fonction harmonique scalaire, et extrait un re-maillage en utilisant un sous-ensemble des iso-courbes de cette fonction. Toutefois, cette méthode requiert de l'utilisateur qu'il définisse à la main les points singulier. De plus, les arêtes ne sont pas alignées avec les directions de courbure principale. Enfin, la génération des arêtes dans la direction orthogonale aux courbes d'iso-valeurs requiert toujours un schéma explicite.

En utilisant la paramétrisation calculée par notre méthode, le re-maillage devient triv-

[MK04] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proc. Pacific Graphics*, pages 207–216, 2004.

[JL97] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proc. of the Workshop on Vis. in Sci. Comp.* Eurographics, 1997.

[DKG04] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. Technical report, August 2004.

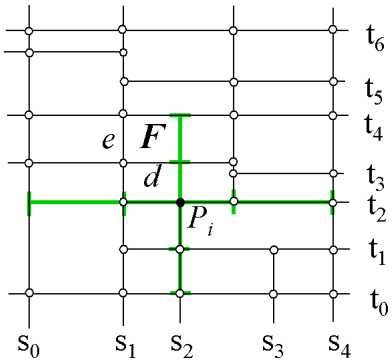


Figure 4.17: *La pré-image d'un T-maillage*

ial. Nous trouvons les sinus et les consinus des paramètres θ et ϕ, puis calculons les paramétrisations par triangles (Sections 4.2). Pour les problèmes de re-maillage, nous pouvons nous passer des étapes supplémentaires (arrangement des cartes et re-paramétrage des cartes singulières). Ainsi, nous pouvons directement extraite les courbes iso-θ et iso-ϕ de la paramétrisation et les utiliser pour former un maillage initial. Ce processus est similaire à l'extraction des frontières de cartes utilisée en Section 4.2.7. La seule différence est qu'au lien d'extraire des courbes iso-$2k\pi$, nous extrayons des courbes avec une densité plus grande, spécifiée par l'utilisateur. Comme notre paramétrisation est quasi-conforme, nous obtenons des quadrilatères bien formés partout (à l'exception du voisinnage immediat des singularités). Comme nous l'avons fait dans notre approche initiale (Section 3.4), nous décomposons les polygônes qui ne sont pas des quadrilatères en un ensemble de quadrilatères et de triangles. Les joinctions en T sont corrigées en introduisant des triangles supplémentaires dans le maillage.

La Figure 4.16 montre des exemples de modèles re-maillés en utilisant notre méthode. La qualité des maillages est très bonne, la plupart des éléments sont quasiment carrés, alignés avec les directions de coubure principale. La surface "lion" est re-maillée avec deux résolutions différentes. La même paramétrisation a été utilisée pour ces deux re-maillage, la seule différente étant la densité de paramètres utilisée pour extraire les courbes d'iso-valeur. La surface "squelette de la main" présente des détails géométriques très fin, tels que les zones tubulaires de la main. Notre méthode génère des éléments quasiment carrés dans ces régions. Les méthodes d'intégration implicite précédentes, dont la notre présentée en Section 3.4, auraient de grandes difficulté à capturer ces régions, puisque la probabilité d'y placer une ligne de courant serait assez faible.

4.4 Conversion de maillages en surfaces paramétriques

4.4.1 Les surfaces T-Splines

La principale limitation des représentations usuelles en CAO/CFAO[3] concerne leur manque de flexibilité, et la nature fortement contrainte des maillages de contrôle admissible pour une classe donnée de surfaces. Par exemple, le maillage de contrôle d'une B-Spline standard doit être localement équivalent à une grille réguilère (un cylindre et un tore sont les seules variations possibles). Une nouvelle classe de surfaces, les T-Splines, a été récemment introduite par Sederberg pour obtenir plus de flexibilité [SZBN03].

[3]Conception Assistée par Ordinateur / Conception et Fabrication Assistée par Ordinateur

[SZBN03] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad H. Nasri. T-splines and T-NURCCs. *ACM TOG (SIGGRAPH)*, 2003.

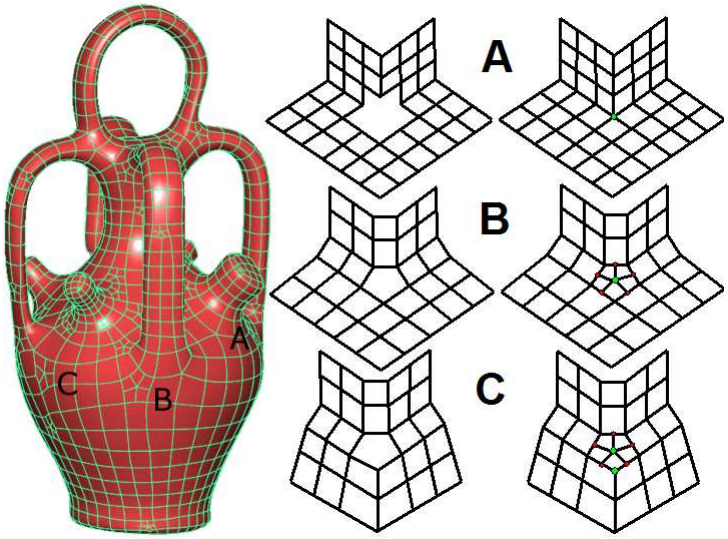


Figure 4.18: *Polygônes de valence N et sommets extraordinaires.*

Les points de contrôle d'une surface T-Spline sont arrangés dans une grille appelée un *T-maillage*[4], dans lequel des raffinements locaux peuvent être réalisés. Si le T-maillage forme une grille régulière, alors la T-Spline correspond exactement à une B-Spline usuelle. La Figure 4.17 montre la pré-image d'un sous-ensemble d'un T-maillage. Toutefois, malgrès la souplesse supplémentaire offerte par les T-jonctions, un T-maillage ne peut pas contenir de sommet de valence arbitraire. Pour cette raison, Sederberg *et. al* ont également proposé la représentation T-NURCC (Non-Uniform Rational Catmull Clark)[SZBN03], qui permet de remplir les voisinages des sommets de valence arbitraire (également appelés *sommets extraordinaires*) tout en se raccordant de manière continue avec les T-Splines.

Dans une T-Spline, chaque arête du T-Maillage a un *intervalle de noeud*[5] associé, qui doit satisfaire deux règles de validité:

Règle 1: Sur une face donnée, la somme des intervalles de noeuds sur des arêtes opposées doit être égale.

Rèfle 2: Si deux T-jonctions sur deux arêtes opposées peuvent être connectées sans violer la règle 1, alors l'arête correspondante doit être présente dans le T-maillage.

Etant donné un T-maillage valide, l'équation d'une T-Spline est donnée par:

$$\mathbf{S}(s,t) = \frac{\sum_{i=1}^{n} w_i P_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}, \quad s,t \in D \tag{4.17}$$

où: P_i set le $i$ème point de contrôle (x_i, y_i, z_i), $i = 1 \cdots n$. La fonction de mélange $B_i(s,t) = N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s) N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t)$, where, $N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s)$ est la fonction de base des B-Splines cubiques associée au vecteur de noeuds[6] $s_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$ et $N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t)$ est associé au vecteur de noeuds $t_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$.

Les deux vecteurs de noeuds s_i et t_i des deux fonctions de base associées au point de contrôle P_i sont calculés à partir du T-maillage de la manière suivante (c.f. Figure 4.17): Soit (s_{i2}, t_{i2}) les coordonnées de noeud de P_i. Considérons un rayon dans l'espace paramétrique $R(\alpha) = (s_{i2} + \alpha, t_{i2})$. Alors s_{i3} et s_{i4} sont les s coordonnées des deux premiers s-côtés intersectés par le rayon dans la direction des α positif. Un s-côté est défini comme un segment vertical sur lequel s est constant. Les s_{i0}, s_{i1} et t_i sont calculés de manière similaire.

[4] T-Mesh
[5] knot inverval
[6] knot vector

Tout d'abord, nous assurons la validité du T-maillage partout, et "poussons" les problèmes vers les sommets extraordinaires quand nous ne pouvons les éviter (les voisinages de ces singularités seront remplacés par des T-NURCCes, comme nous l'expliquons dans la prochaine Section):

1. A chaque fois que cela est possible, satisfaire la Règle 2 en insérant les arêtes manquantes dans toutes les facettes du T-maillage;
2. Pour toutes les facettes invalides restantes (ce qui inclut les facettes de valence N), les convertir en sommets extraordinaires de la manière explicitée ci-dessous (voir également la Figure 4.18).

◇ A: dans chaque facette de valence N paire, un nouveau sommet est ajouté et connecté à un sommet sur deux de la facette. Ceci crée N quadrilatères supplémentaire et un sommet extraordinaire.

◇ B: dans chaque facette de valence N impaire, un nouveau sommet est inséré et connecté à une nouvelle T-jonction sur chaque arête de la facette. Ceci crée N quadrilatères supplémentaires, N T-jonctions et un sommet extraordinaire ;

◇ C: de manière similaire, les facettes de valence N impaire avec un sommet de valence 3 sur le bord sont re-maillés comme nous le montrons sur la figure.

A chaque fois qu'une arête est subdivisée, les intervalles de noeuds associés sont divisés par 2. Ces trois opérations garantissent que les intervalles de noeuds restent cohérents.

Après cette étape, le maillage de contrôle est un T-maillage valide, excepté dans le voisinage des sommets extraordinaires. Les facettes concernées sont remplacées par des T-NURCCes, comme nous l'expliquons dans la section suivante.

4.4.2 Sommets extraordinaires et T-NURCC

Une surface de type TNURCC[7] est une surface NURCC, à savoir une modification des NURSS[8] cubiques[SZBN03], admettant des T-jonctions dans le même esprit que les T-Splines. Les surfaces NURCCs sont une générationsation des surfaces B-Spline non-uniformes définies par produit tensoriel et des surfaces de subdivision de Catmull-Clark. Ces surfaces satisfont la contrainte assurant que deux arêtes opposées d'un quadrilatère du maillage de contrôle ont le même intervalle de noeuds. Ceci rends possible la subdivision locale des surfaces NURCC. Dans notre implantation, nous appliquons conceptuellement deux étapes de subdivision locale à chaque sommet extraordinaire (c.f. Figure 4.19). Ceci génère un ensemble de points de contrôle additionels, symbolisés en rouge sur la Figure, et exprimés par des combinaisons linéaires des points de contrôle initiaux. Les coefficients de ces combinaisons linéaires (qui dépendent de la valence du sommet) sont données dans l'article initial de Sederberg *et. al*[SZBN03]. En pratique, nous gardons une copie du maillage de contrôle original, et appliquons les subdivisions locales à une copie. En appliquant ces subdivisions, nous gardons associé à chaque nouveau point de contrôle la liste des points de contrôle initial dont il dépend ainsi que les coefficients. Cette représentation peut directement être utilisée dans l'étape d'ajustement aux données, présentée dans la section suivante.

4.4.3 Ajustement aux données

Pour réaliser l'ajustement aux données, il est nécessaire de parvenir à définir une métrique d'erreur, que nous chercherons à minimiser. En d'autres termes, il va être nécessaire de mettre la surface T-Spline reconstruite en correspondance avec les données initiale de manière à mesurer la déviation entre les deux. Les méthodes sans paramétrisation[MK04],

[7] Non Uniform Rational Cubic
[8] Non Uniform Rational Subdivision Surfaces

[MK04] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proc. Pacific Graphics*, pages 207–216, 2004.

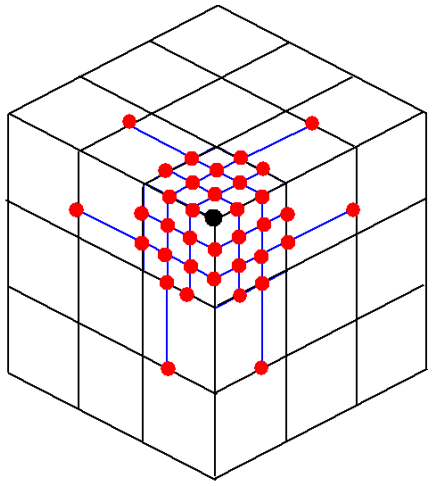


Figure 4.19: *Nous calculons une approximation du voisinage d'un sommet extraordinaire de valence 3, représenté par une surface T-NURCC, en appliquant deux niveaux de subdivision. L'approximation ainsi construire est une surface T-Spline valide.*

réalisent cette mise en correspondance en se fondant exclusivement sur des relations géométriques de proximité, à savoir en projetant les sommets initiaux sur la surface reconstruite. Les méthodes à base de paramétrisation établissent cette correspondance en munissant chaque sommet initial de coordonnées paramétriques, compatible avec la représentation de la surface reconstruite. Notre approche s'apparente à cette classe de méthodes.

Etant donnée une paramétrisation $(s,t) \mapsto \mathbf{S}(s,t) \in \mathbb{R}^3$ de la surface, nous allons minimiser la fonctionelle d'énergie suivante, comme dans les méthodes d'ajustement aux données régularisées classiques[Gre94]:

$$E = E_{fit} + \sigma E_{fair}$$

$$\text{avec:} \begin{cases} E_{fit} & = & \int \|\mathbf{S}(s,t) - \mathbf{M}(s,t)\|^2 dsdt \\ E_{fair} & = & \int \left((\frac{\partial^2 \mathbf{S}}{\partial s^2})^2 + 2(\frac{\partial^2 \mathbf{S}}{\partial s \partial t})^2 + (\frac{\partial^2 \mathbf{S}}{\partial t^2})^2 \right) dsdt \end{cases} \tag{4.18}$$

Dans cette équation, $\mathbf{M}(s,t)$ dénote une paramétrisation de la surface initiale triangulée, et n dénote le nombre de points de contrôle. Comme ce qui est courament fait en ajustement de Splines, nous approximons le terme d'ajustement aux données E_{fit} en utilisant un ensemble discret de m échantillons:

$$E_{fit} \simeq \sum_{k=1}^{m} \|\mathbf{S}(s_k,t_k) - (x_k,y_k,z_k)\|^2$$

Pour chaque échantillon (x_k,y_k,z_k) de la surface initiale, (s_k,t_k) dénote ses coordonnées dans l'espace paramétrique. Une idée naturelle serait d'utiliser les sommet de la surface initiale, mais il est plus judicieux de la ré-échantilloner, en utilisant dans chaque carte une grille régulière d'échantillons (nous utilisons 10×10 échantillons par carte dans notre implantation). Ceci permet de se rendre plus indépendant de la qualité du maillage initial.

L'approximation de l'énergie plaques-minces E_{fair} évite les oscillations parasite de la surface Spline. Toutefois, si pour le coefficient σ nous utilisons une valeur trop grande, la surface Spline finale risque de passer loin des sommets initiaux. Dans nos exemples, nous utilisons $\sigma = 0.05$.

Nous pouvons remarquer que par construction, notre maillage de contrôle et les vecteurs de noeuds associés définissent une T-Spline standard (ou semi-standard). Ainsi, les dénom-

[Gre94] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13:143–154, 1994.

inateurs de la T-Spline sont tous égaux à 1, et nous pouvons nous concentrer sur les numérateurs.

$$\mathbf{S}(s,t) = \sum_{i=1}^{n} \mathbf{P}_i \mathbf{B}_i(s,t)$$

Chaque coordonnée x, y, z peut être traitée indépendament. Pour la coordonnée x, le terme d'ajustement aux données est:

$$E_{fit}^{x} = \sum_{k=1}^{m} \left(\sum_{i=1}^{n} X_i \mathbf{B}_i(s_k,t_k) - x_k \right)^2 \tag{4.19}$$

où X_i (resp Y_i, Z_i) dénote les coordonnées du point de contrôle $\mathbf{P}_i$. Les X_i qui minimisent l'"Equation 4.19 sont également solution d'un système linéaire $A^t AX = A^t b$, où les coefficients de la matrice A $(m \times n)$ sont donnés par $a_{k,i} = B_i(s_k,t_k)$ et le second membre par $b_k = x_k$. Le vecteur d'inconnues X correspond à l'enemble des coordonnées x de tous les points de contrôle. En ajoutant le terme de lissage E_{fair}, le système linéaire devient:

$$\left(A^t A + \sigma(A_{ss}^t A_{ss} + 2A_{st}^t A_{st} + A_{tt}^t A_{tt})\right) X = A^t b \tag{4.20}$$

où les coefficients $(\cdot_{k,i})$ des matrices A_{ss}, A_{st}, A_{tt} de taille $m \times n$ correspondent aux dérivées secondes $B_{iss}(s_k,t_k), B_{ist}(s_k,t_k)$ et $B_{itt}(s_k,t_k)$ des fonctions de base B_i.

Pour résoudre ce problème d'ajustement aux données régularisé, les deux difficultés restante concernent le calcul des paramètres (s_k,t_k) associés aux sommets de la surface originale, et le calcul des matrices $A, A_{ss}, A_{st}, A_{tt}$ et du second membre b par accumulation des contributions de toutes les fonctions de base. Ces deux aspects sont abordés dans les deux sections suivantes.

Calcul des paramètres (s_k,t_k)

La paramétrisation de la surface triangulée de départ est obtenue de la manière suivante, en se fondant sur notre methode exposée en Section 4.2. Comme sans la section précédente, le maillage de départ est décomposé en un ensemble de cartes quadrilatères. En gardant la relation entre le maillage de contrôle et la surface, il est possible de retrouver pour chaque facette du maillage de contrôle l'ensemble de triangles contenus par cette facette (en utilisant un algorithme glouton). Deux cas différents peuvent se présenter, suivant la présence de singularités. Comme nous l'avons expliqué en Section 4.2, nous procédons de la manière suivante:

- ◇ *aucun triangle, arête, ou sommet n'est singulier:* une paramétrisation peut alors facilement être obtenue depuis les valeurs des paramètres (θ, ϕ), en assemblant récursivement les triangles dans l'espace paramétrique;
- ◇ *la facette contient des singularités:* nous re-paramétrons l'intérieur de cette facette en utilisant la méthode de Floater[Flo03].

Les facettes du maillage de contrôle modifiées par l'utilisateur sont considérées comme appartenant à la deuxième catégorie (en effet, le lien entre le bord des cartes et la paramétrisation iterne pouvant avoir été brisé, les cartes concernées nécessitent un re-paramétrage). Une fois que toutes les cartes sont munies d'une paramétrisation valide, comme précédement, nous optimisons la continuité inter-cartes par une méthode de relaxation globale[SPPH04], appliquée aux cartes ayant subies une re-paramétrisation (la paramétrisation des autres cartes présente déjà une continuité globale).

[Flo03] M. S. Floater. Mean value coordinates. *CAGD*, 20:19–27, 2003.

[SPPH04] J. Schreiner, A. Prakash, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM TOG (Proc. SIGGRAPH 2004)*, 2004.

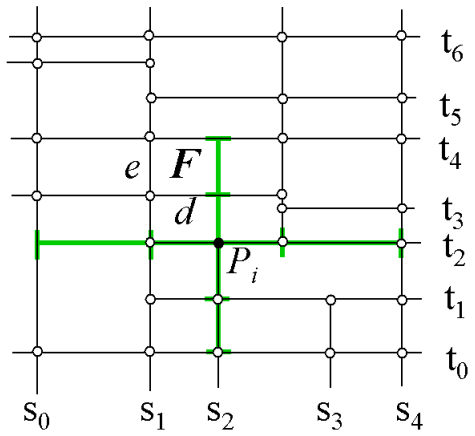


Figure 4.20: *La pré-image d'un sommet du maillage de contrôle*

Construction et résolution du système linéaire

Pour résoudre notre problème d'ajustement aux données régularisé (Equation 4.18), la manière la plus naturelle consiterait à parcourir la surface carte par carte. Ceci reviendrait à parcourir les matrices $A, A_{ss}, A_{st}, A_{tt}$ ligne par ligne, et permettrait de construire directement la matrice globale de la forme quadratique sans avoir à stocker de matrices intermédiaires.

Toutefois, ceci requiert d'être capable de construire les pré-images de chaque carte, ce qui se révèle malaisé du point de vue de l'implantation. Pour cette raison, nous préférons parcourir les noeuds de contrôle (plutôt que les cartes), qui ont des pré-images bien plus simples à construire. En d'autres termes, ceci revient à considérer une seule fonction de base $\mathbf{B_i}(s,t)$ à la fois. Le prix à payer est le stockage des matrices intermédiaires $A, A_{ss}, A_{st}, A_{tt}$, construites colonne par colonne, mais devant la simplicité d'implantation ainsi obtenue, nous avons estimé que la petite perte d'espace mémoire ainsi occasionnée était négligeable. Après le parcours de tous les noeuds de contrôle (et donc de toutes les fonctions de base), la matrice finale du système est finalement assemblée (Equation 4.20).

Les fonctions de base sont définies par morceaux dans un voisinage autour de la pré-image de chaque noeud de contrôle $\mathbf{P}_i$ (c.f. Figure 4.20). Chaque fonction de base $\mathbf{B}_i$ est définie par la donnée du T-maillage et des vecteurs de noeuds associés autour du point de contrôle $\mathbf{P}_i$. Lors de la construction de la matrice, pour retrouver ces paramètres, comme la surface est invariante par translation appliquée à l'espace paramétrique (seuls les intervalles de noeuds ont une influence), nous fixons des coordonnées arbitraires (s_0, t_0) associées à P_i. Ensuite, nous propageons itérativement les intervalles de noeuds jusqu'à ce que la région d'influence $D_i = [s_{i0}, s_{i4}] \times [t_{i0}, t_{i4}]$ soit complètement déterminée. La pré-image d'un noeud de contrôle et de la région d'influence associée à l'aspect que nous montrons sur la Figure 4.20.

Nous rappelons que la surface T-Spline est décrite sous forme de carreaux[9] polynomiaux de bi-degré 3, associés aux facettes du maillage de contrôle. Comme nous pouvons l'observer sur cette paramétrisation locale, le point de contrôle P_i influence les carreaux de T-Spline qui correspondent aux facettes intersectées par la région d'influence D_i. Le carreau de T-Spline associé à chaque facette du maillage de contrôle est mis en correspondance avec l'espace paramétrique canonique $[0,1]^2$, avec l'origine $(0,0)$ correspondant à l'un des coins du carreau. Ainsi, lorsque le point de contrôle P_i est pris en compte dans les matrices du système, sa pré-image doit subir une homotétie de manière à rester compatible avec les autres termes. Par exemple, dans la pré-image du T-maillage, d et e dénotent les dimensions du rectangle correspondant à une facette F présente dans la région d'influence, les vecteurs de noeuds en s et t doivent être pondérés par $1/d$ et $1/e$ respectivement. Une fois que les vecteurs de noeuds et la région d'influence D_i des fonctions de base D_i ont été

[9] patches

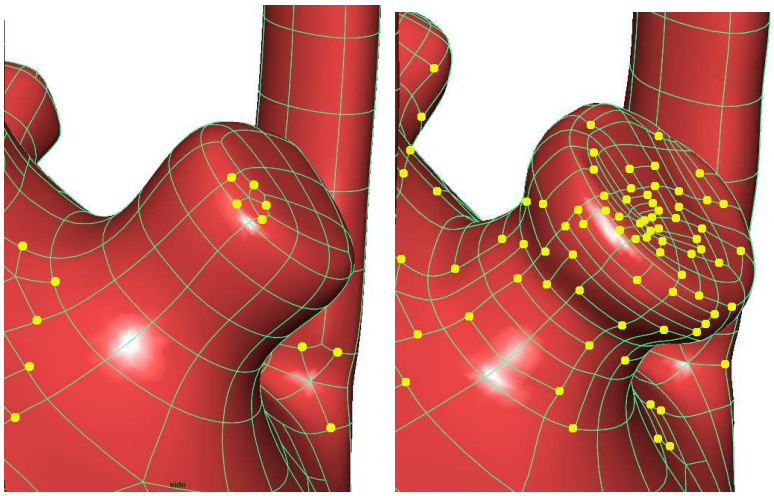


Figure 4.21: *Des subdivisions locales adaptatives permettent de mieux capturer la géométrie dans certaines zones.*

determinés, nous pouvons alors mettre à jour les colonnes correspondantes dans les matrices A, A_{ss}, A_{st} et A_{tt}. A la fin du parcours de tous les points de contrôle du maillage, la matrice globale et le second membre du système sont construits. Dans notre implantation, toutes les matrices sont représentées dans le format CCS[10]. Pour résoudre le système, nous utilisons le solveur direct creux TAUCS[MIT06], mentionné dans la section 2.2. Comme TAUCS calcule l'inverse de la matrice, celle-ci peut être réutilisée pour résoudre les trois systèmes indépendants en X, Y et Z (seul le second membre change). Nous remarquons que les solveurs directs creux fonctionnent si bien que de manière surprenante, inverser la matrice devient plus rapide que simplement résoudre un système linéaire avec le gradient conjugué pré-conditionné [BBK05], [Lé05].

Contrôle de la norme L^∞, ajustement adaptatif

Dans la section précédente, nous avons décrit l'algorithme permettant l'ajustement aux données en minimisant la norme L^2 de la déviation. Dans certaines situations, le maillage de contrôle automatiquement construit par notre méthode peut présenter un nombre de degrés de liberté trop faible pour parvenir à capturer une géométrie complexe. En conséquence, la déviation reste importante, et la surface T-Spline ainsi reconstruite n'est pas une bonne approximation de la surface initiale. Dans de telle situation, il est donc nécessaire d'ajouter des degrés de liberté (à savoir des noeuds de contrôle) supplémentaires *a posteriori*. En général, avec des surfaces Splines traditionelles (qui ne supportent pas les raffinements locaux), ceci est réalisé par raffinement *global* du maillage de contrôle. L'inconvénient principal de ces raffinement globaux est qu'ils rajoutent un grand nombre de noeuds de contrôle, y compris dans les régions où l'erreur d'approximation était faible. Dans notre cas, comme nous utilisons la représentation T-Spline qui supporte des opérations de raffinement locaux, de nouveaux points de contrôles peuvent être ajoutés *localement* dans les régions où l'erreur d'approximation est élevée (voir la Figure 4.21). De plus, comme les T-Splines ont la propriété du contrôle local, lors de ces opérations de raffinement locaux, la procédure d'ajustement aux données ne nécessite pas le re-calcul de

[10] Column Compressed Storage, i.e. stockage par colonnes compressées

[MIT06] Omer Meshar, Dror Irony, and Sivan Toledo. An out-of-core sparse symmetric indefinite factorization method. *ACM Transactions on Mathematical Software*, 32:445–471, 2006.

[BBK05] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for mesh processing. In *IMA Mathematics of Surfaces XI, Lecture Notes in Computer Science*, 2005.

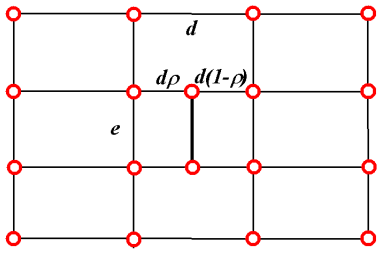


Figure 4.22: *Raffinement local*

toutes les positions des noeuds de contrôle, et se limite à un petit voisinage de la zone modifiée. Ainsi, l'opération de raffinement local nécessite la résolution d'un système linéaire de dimension bien plus réduite que le système global résolu dans la section précédente. Ce système ne combine que les points de contrôle affectés par l'opération de raffinement local. La métrique L^∞ que nous souhaitons minimiser s'exprime de la manière suivante:

$$L_\infty(\mathbf{S},\mathbf{M}) = max_S \|(\mathbf{S}(s,t) - \mathbf{M}(s,t))\|^2 \tag{4.21}$$

où $\mathbf{M}(s,t)$ dénote la paramétrisation de la surface originale. Comme précédemment pour la norme L_2, nous utilisons un échantillonage régulier dans l'espace paramétrique de chaque carte.

Nous appliqons itérativement les opérations de raffinement local jusqu'à ce que l'erreur d'approximation L_∞ du plus mauvais carreau passe en dessous d'un seuil défini par l'utilisateur.

Les possibilités de raffinement local représentent l'une des propriétés intéressantes des surfaces T-Splines. Ces raffinement locaux sont également appelés *insertion de points de contrôles*. De nouveaux points de contrôle sont tout d'abord insérés dans le T-maillage sans changer la géométrie de la surface T-Spline initiale. Ensuite, l'algorithme d'ajustement aux données local trouve la position de ces nouveaux points de contrôle qui minimise l'erreur d'approximation. Pour insérer ces nouveaux points de contrôle, il est également nécessaire d'assurer la préservation des critères de validité des surfaces T-Splines décrites en Section 4.4, en ajoutant des points de contrôle supplémentaire dans certains cas. L'algorithme complet consiste en les étapes suivantes:

1. Insérer le ou les nouveaux points de contrôle dans le T-maillage;
2. Vérifier et restorer la contrainte de validité dictée par la règle 1, en insérant de nouveaux noeuds de contrôle afin de respecter l'égalité des vecteurs de noeuds sur les arêtes opposées des quadrilatères de contrôle;
3. Si des bases de fonctions ont des noeuds de contrôle supplémentaire, mettre à jour le T-maillage de manière adaptés;
4. Répeter les étapes 2 et 3 jusqu'à ce que le T-maillage satisfasse les deux contraintes de validité.

Nous insérons une arête dans la facette du maillage de contrôle ayant la plus forte erreur d'approximation $_\infty$, ce qui décompose la facette en deux. Nous calculons ce raffinement local dans les deux directions (s et t), et choisissons la direction qui minimise le plus l'erreur d'approximation. La Figure 4.22 montre comment le maillage de contrôle et les intervalles de noeuds sont mis à jour. Le rapport des intervalles de noeuds ρ est positionné à 0.5 dans notre implantation[11]. Nous remarquons que cet algorithme d'insertion locale de points de

[11] il serait également possible de chercher à choisir ce paramètre de manière a minimiser l'erreur d'approximation, mais les dépendances non-linéaires qui en resulteraient seraient assez difficile à prendre en compte

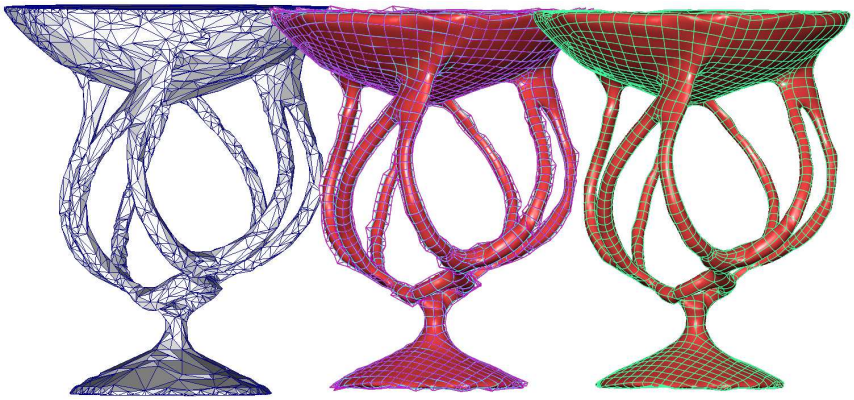


Figure 4.23: *Notre méthode appliquée à un objet de genre élevé: De gauche à droite: maillage initial, maillage de contrôle adapté à la surface, et surface T-Spline finale.*

contrôle est susceptible de se propager dans le T-maillage et d'insérer quelques points de contrôle supplémentaires.

L'opération de raffinement local des T-Spline préserve la géométrie de la surface. Dans notre cas, notre objectif en réalisant cette opération est de faire appaitre de nouveaux degrés de liberté et de choisir ces degrés de liberté de manière à minimiser l'erreur d'approximation. Pour ce faire, nous réalisons un ajustement aux données local. Comme les fonctions de base T-Spline ont des supports compacts, ceci ne requiert que la résolution d'un petit sytème linéaire. Comme expliqué dans la section 2.2.4, le vecteur X qui rassemble toutes les coordonnées X_i des points de contrôle est subdivisé en deux sous-ensembles, à savoir X_f, l'ensemble des points de contrôle influencé par le nouveau point de contrôle, et X_l, l'ensemble des points de contrôle qui ne tombent pas dans sa zone d'influence (et donc qui ne bougeront pas pendant la nouvelle étape d'ajustement aux données). Ces nouveaux degrés de liberté ainsi que les termes de couplage à la frontière de la zone d'influence sont déterminés par le motif des entrées non-nulles de la matrice A. Le terme d'ajustement aux données s'écrit:

$$F_{fit}(X_f) = \left\| \left[A_f | A_l\right] \left[\frac{X_f}{X_l}\right] - b \right\|^2$$

où A est décomposée en A_f et en A_l, suivant X_f et X_l. Les nouveaux degrés de liberté sont alors donnés par la solution du système linéaire suivant:

$$A_f^t A_f X_f = A_f^t A_l X_l - A_f^t b$$

Les termes provenant de l'énergie de lissage ont la même structure que dans la section précédente. Comme nous n'avons qu'un petit nombte de coefficients et comme la position des nouveaux points de contrôle est en générale assez proche de leur position précédente, un solveur itératif de type gradient conjugué (c.f. Section 2.2.2) converge de manière très efficace en quelques itérations.

4.4.4 Résultats

Nous montrons quelques résultats sur les Figures 4.23, 4.24 et 4.25. Notre méthode fonctionne pour une grande variété de formes de surfaces, et surtout pour des surfaces de genre arbitraire, ce qui est très difficile à réaliser avec les approches classiques. Notons que le culbuteur (Figure 4.25) est topologiquement équivalent à un tore. Par conséquent, il serait possible de créer un maillage de contrôle sans aucune singularité. Toutefois, nous pensons que le maillage de contrôle construit par notre méthode est plus naturellement adapté à cette forme, car il parvient à mieux prendre en compte les spécificités et détails géométriques (tels que les protrusions). Ceci est encore plus visible sur l'exemple de la

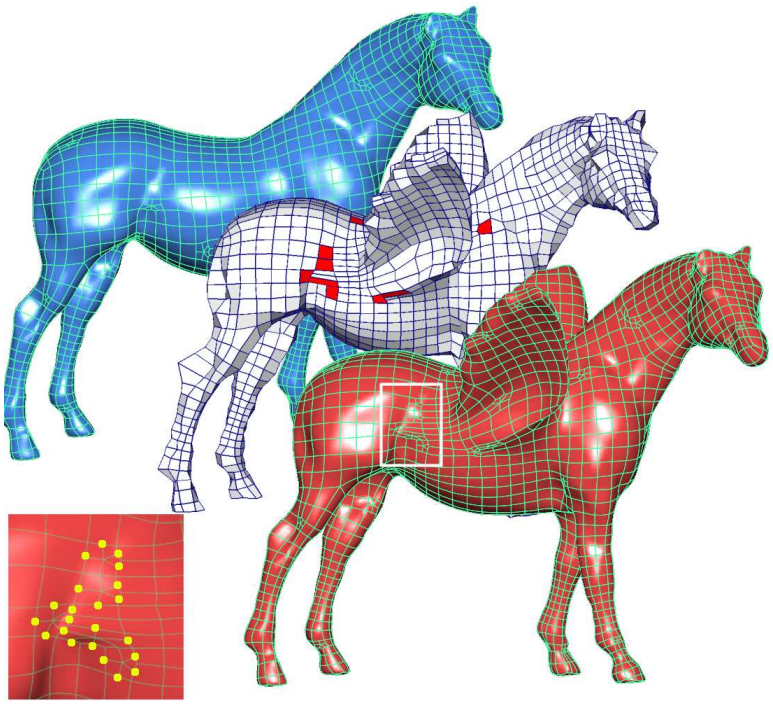


Figure 4.24: *Conversion de surfaces triangulées en T-Splines et édition interactive dans le modeleur commercial Maya. Cette figure montre que nous sommes parvenu à refermer la boucle entre objets réels, acquisition et modélisation 3D. Les zooms montrent comment les facettes de valence N sont remplacées par des surfaces T-NURCC.*

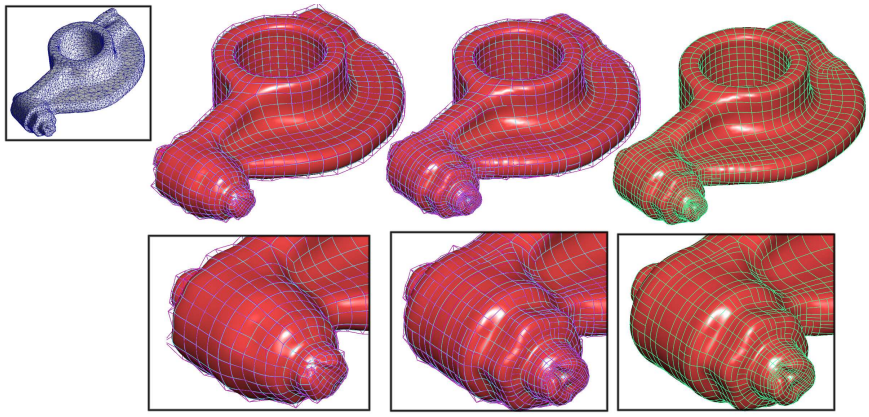


Figure 4.25: *Conversion d'un culbuteur scanné en surface T-Spline. De gauche à droite: ajustement L_2, ajustement L_∞ avec raffinement local (visualisés avec et sans le maillage de contrôle en surimpression).*

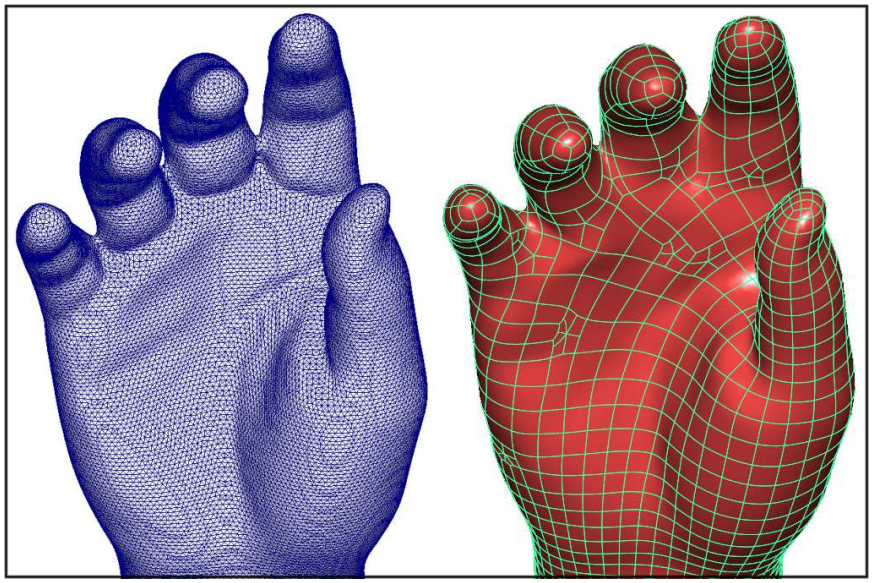


Figure 4.26: *Conversion d'une main scannée en T-Spline.*

main scannée de laFigure 4.26. Comme cette surface est un disque topolgique, nous pourrions utiliser une technique de paramétrisation 2D (par exemple, celles expliquées en Sections 3.3.3, 3.3.6), mais il semble naturel de faire apparaitre une singularité au sommet de chaque doigt, comme le ferait un designer qui souhaiterait créer cette forme dans un modeleur 3D. Ces singularités permettent de relaxer les contraintes dans les zones de fortes courbure, et ainsi d'optimiser les capacités d'approximation du maillage de contrôle, en répartissant les déformations de la paramétrisation de manière plus uniforme.

L'ensemble des méthodes présentées dans les Sections 4.2 (paramétrisation globale) et la section courante (ajustement de surfaces T-Spline) permet en quelque sorte de refermer la boucle objet réel - acquisition 3D - modélisation 3D - prototypage rapide. En particulier, ces méthodes contribuent à franchir le précipisse existant entre les étapes de l'acquisition 3D et de la modélisation 3D. Toutefois, comme nous avons pu le voir, ces méthodes restent extrèmemement complexes, délicates à implanter, et dépendent d'un ensemble de paramètres difficiles à règler de manière optimum pour l'utilisateur. Pour cette raisons, nous orientons à présent nos recherches vers d'autres classes de méthodes, susceptibles à la fois de fournir un formalisme plus élégant et des algorithmes plus simples.

4.5 Méthodes spectrales

Ces quelques dernières années, la paramétrisation des surfaces triangulées a été un sujet qui a connu une grande activité de recherche ces quelques dernières années. Tout d'abord, la communauté s'est interessée à la paramétrisation d'objets simples, tels des disques topologiques[Flo97] ou des sphères[GGS03a]. Plus récemment, des méthodes sont fondées par exemple sur la notion de 1-formes holomorphes[GY03b]. Un tel intérêt pour ces méthodes de paramétrisation peut s'expliquer par le fait qu'en construisant une représentation plus abstraite de l'objet, elles facilitent sa conversion sous différentes représentation (Section 4.4), ou encore elles donnent la possibilité d'attacher des propriétés à cet objet

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

[GGS03a] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM TOG (SIGGRAPH)*, 22:358–363, 2003.

[GY03b] Xianfeng Gu and Shing-Tung Yau. Global conformal parameterization. In *Symposium on Geometry Processing*, pages 127–137, 2003.

(comme dans le cas du plaquage de textures, c.f. Section 3.3.2). D'autres applications concernent l'utilisation de l'espace paramétrique pour y représenter des propriétés physiques, simulées par la méthode des éléments finis (c.f. Section 3.6).

Toutefois, des avancées récentes dans les domaines de la résolution d'équations aux dérivées partielles suggèrent qu'une paramétrisation n'est peut-être pas la meilleure représentation de la géométrie et des propriétés physiques. Par exemple, pour construire une représentation hiérarchique, la méthode CHARMS[GKS02a] suggère de concevoir la hiérarchie en termes de raffinement des *fonctions de base* plutôt que le point de vue raffinement des *éléments finis* communément utilisé. Cette vision centrée sur la représentation fonctionelle conduit à des calcules beaucoup plus simples et naturels. D'autre part, le *calcul extérieur* et ses équivalents discrets sont de plus en plus utilisés comme outil de base par la communauté du traitement numérique de la géométrie (c.f. le cours Siggraph de Matthieu Desbrun et Peter Schrörder présenté à la conférence ACM Siggraph en 2005).

Pour cette raison, plutôt que de chercher à construire une paramétrisation globale des objets (Section 4.2), nous nous intéressons à présent à des algorithmes permettant de définir des *bases de fonctions* attachées à un objet de topologie arbitraire. Les méthodes spéctrales (fondées sur le calcul des fonctions propres d'un opérateur, c.f. Section 2.3.3) semblent être un formalisme intéressant pour atteindre cet objectif. J'ai présenté cette idée de base lors d'un article invité à la conférence IEEE Shape Modeling International [Lé06]. Nous avons débuté l'étude de ces méthodes avec Bruno Vallet, doctorant de l'équipe. Nos premiers résultats ont été acceptés à la conférence Eurographics 2008 [VL].

Dans cette section, nous étudions un certain type de base de fonctions, définie comme les fonctions propres de l'opérateur de Laplace-Beltrami. En appliquant cette définition à des objets simples, nous retrouvons des fonctions de base classiques (transformée en cosinus discrète dans le carré et harmoniques sphériques sur la sphère). Pour des géométries et des topologies plus compliquées, ces notions permettent de définir une base de fonction naturelle de l'objet, permettant de décomposer tout signal en fréquences. En d'autres termes, ceci permet de réaliser l'analyse de Fourier appliquée à des signaux définis sur des variétés de genre arbitraire.

L'intérêt principal de cette approche est que cette analyse peut être réalisée en se passant de toute paramétrisation de l'objet. Ainsi, nous gravissons un echelon supplémentaire sur l'échelle de l'abstraction, en considérant une variété non plus comme un ensemble de triangles, ni comme un ensemble de cartes, mais comme une base de fonctions permettant de manipuler des signaux sans devoir faire référence au domaine géométrique.

En nous fondant sur des analogies physiques (modes de vibration propres), nous tenterons tout d'abord de donner une vision intuitive de ces notions. Nous expliquerons ensuite comment calculer les fonctions propres d'un opérateur différentiel, et comment projeter un signal sur ces fonctions propres, en utilisant le formalisme des éléments finis. Nous montrerons quelques applications possible au filtrage des signaux définis sur des variétés (ce qui comprend la géométrie de ces variétés). Nous commencerons par étudier le problème discret ainsi que ses différentes déclinaisons afin de saisir d'une manière intuitive la signification géométrique des fonctions propres. Nous établieront ensuite les liens avec l'analyse de Fourier et le filtrage dans le domaine fréquentiel.

[GKS02a] E. Grinspun, P. Krysl, and P. Schroder. Charms: A simple framework for adaptive simulation, 2002.

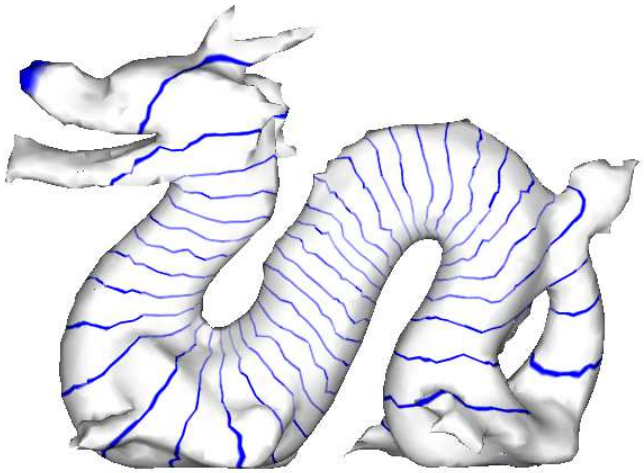


Figure 4.27: *Le vecteur de Fiedler définit un ordre naturel pour les noeuds d'un graphe. Les lignes de niveau montrent comment cet ordre suit naturellement la forme globale de ce dragon.*

4.5.1 Le mode discret: Laplaciens de graphes

Dans le domaine du traitement numérique de la géométrie, la théorie spectrale des graphes a été par exemple utilisée pour calculer un ordre des sommets d'un maillage facilitant son traitement par des algorithmes dits à mémoire externe[12], dont l'occupation en mémoire vive est bornée, et qui font appel au disque dur pour stocker des données temporaires[IL05]. Un tel ordre naturel peut être calculé à partir du vecteur de Fiedler du Laplacien du graphe, défini ci-dessous. Le Laplacien d'un graphe est une matrice $L = (a_{i,j})$ définie par:

$$\begin{array}{rcll} a_{i,j} & = & w_{i,j} > 0 & \text{if } (i,j) \text{ est une arête} \\ a_{i,i} & = & -\sum w_{i,j}) & \\ a_{i,j} & = & 0 & \text{sinon} \end{array}$$

où les coefficients $w_{i,j}$ sont des poids associés aux arêtes du graphe. Il est possible d'utiliser la pondération uniforme $w_{i,j} = 1$ ou des pondérations plus évoluées, calculées à partir du plongement du graphe (autrement dit, prenant mieux en compte la géométrie de la surface).

Le premier vecteur propre du Laplacien du graphe est le vecteur $(1,1\ldots1)$, associé à la valeur propre 0. Le second vecteur propre est appelé vecteur de Fiedler et présente des propriétés intéressantes, qui lui permettent de définir des ordres de parcours du graphe optimum pour certains calculs numériques[Fie73,Fie75]. En triant les sommet i par u_i croissants, où u_i dénote la $i$ème composante du vecteur de Fiedler, il est par exemple possible d'optimiser les accès au cache dans les solveurs numériques itératifs creux. D'autres applications concernent le traitement de la géométrie par des algorithmes en mémoire externe[IL05]. La Figure 4.27 montre l'aspect de la "coordonnée" ainsi définie. D'une certaine manière, cette coordonnée ressemble à une "analyse en composantes principales non-linéaire", capable de suivre la forme générale du maillage (même si celle-ci n'est pas

[12] ou OOC pour Out-of-Core

[IL05] Martin Isenburg and Peter Lindstrom. Streaming meshes. In *IEEE Visualization*, page 30, 2005.

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czech. Math. Journal*, 23:298–305, 1973.

[Fie75] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. Journal*, 25:619–633, 1975.

[IL05] Martin Isenburg and Peter Lindstrom. Streaming meshes. In *IEEE Visualization*, page 30, 2005.

rectiligne).

Nous allons à présent tenter de mieux comprendre comment le vecteur de Fiedler parvient à capturer une telle information géométrique globale. Une autre définition alternative du vecteur de Fiedler permet de mieux saisir ses propriétés. Le vecteur de Fiedler $u = (u_1 \dots u_n)$ est la solution du problème d'optimisation contrainte suivant:

$$\begin{aligned} &\text{Minimiser:} \quad F(u) = 1/2 u^t L u = 1/2 \textstyle\sum_{i,j} w_{i,j}(u_i - u_j)^2 \\ &\text{Avec les contraintes:} \quad \textstyle\sum_i u_i = 0 \quad \text{et} \quad 1/2 \textstyle\sum_i u_i^2 = 1 \end{aligned} \tag{4.22}$$

En d'autres termes, supposons qu'on soit en présence d'un maillage plongé dans un certain espace (par exemple $\mathbb{R}^3$, et supposons que les poids $w_{i,j}$ correspondent à la longueur des arêtes dans cet espace. Ainsi, le vecteur de Fiedler $(u_1 \dots u_n)$ définit un plongement 1D du graphe sur une ligne, tout en essayant de respecter les longeurs des arêtes du graphe. En quelque sorte, nous pouvons considérer ce problème comme une paramétrisation 1D (non-bijective) de la surface.

Les contraintes suppriment les ambiguités: la première contrainte place le centre de gravité à l'origine, et la deuxième contrainte permet d'éviter la solution dégénérée ($\forall i, u_i = 0$), en fixant le premier moment à 1.

Nous pouvons montrer facilement que la solution de ce problème est donnée par le vecteur de Fiedler, en écrivant le Lagrangien de ce problème (c.f. Section 2.2.6):

$$L = 1/2 u^t L u + \lambda_1 (1, 1 \dots 1) u + 1/2 \lambda_2 (u^t u - 1)$$

La solution u du problème d'optimisation contrainte doit annuler les dérivées du Lagrangien:

$$\begin{array}{llllll} (1) & \nabla_u L & = & L u + \lambda_1 (1, 1 \dots 1) + \lambda_2 u & = & 0 \\ (2) & \partial L / \partial \lambda_1 & = & (1, 1 \dots 1) u & = & 0 \\ (3) & \partial L / \partial \lambda_2 & = & 1/2 (u^t u - 1) & = & 0 \end{array}$$

Nous pouvons vérifier facilement que ces dérivées s'annulent en choisissant pour u le vecteur de Fiedler normalisé, en prenant $\lambda_1 = 0$ et en choisissant pour λ_2 l'opposée de la valeur propre associée au vecteur de Fiedler. La condition (1) et la condition (3) sont trivialement satisfaites. La condition (2) provient du fait que $(1, 1 \dots 1)$ est vecteur propre de L, et que L étant symétrique, ses vecteurs propres sont orthogonaux[13], donc le vecteur de Fiedler est orthogonal au vecteur $(1, 1 \dots 1)$ $\square$

Cette interpretation du vecteur de Fiedler en tant que plongement 1D minimisant les déformation conduit à se demander si les autres vecteurs propres du Laplacien de graphe permettent de définir des plongement dans des espaces de dimension supérieure à 1. Par exemple, un plongement du graphe dans un espace de dimension 2 correspondrait à un problème de paramétrisation classique. Ce problème général est bien connu de la communauté de recherche regroupée autour de la thématique de l'*apprentissage automatique*, à savoir une branche particulière de l'intelligence artificielle[14]. Dans cette discipline, une classe de méthode considère des nuages de points dans des espaces de grande dimension, et tentent d'organiser ou de simplifier ces données en identifiant des paramètres. Cette problématique est connue sous le nom d'*apprentissage variété*[15], ou encore de *réduction de la dimension*. En effet, ces méthodes tentent d'identifier la dimension intrinsèque de l'espace dans lequel vivent les données. Le lecteur souhaitant plus de détails sur ce sujet passionnant pourra se réferrer aux pages web de Martin Law http://www.

[13] c.f. Section 2.3 pour une démonstration de la version continue de ce théorème

[14] on dit à présent *représentation des connaissances* pour éviter les considérations existentielles liées à cette dénomination initiale de la discipline

[15] Manifold Learning

cse.msu.edu/~lawhiu/manifold/ et de Stephane Lafon http://www.math.yale.edu/~sl349/index.html.

Sur cette dernière page web, il est possible de voir une démonstration de classification d'un ensemble d'images d'un même objet photographié sous des conditions différentes (variations d'orientation, d'éclairage, de pose). Ces méthodes parviennent à partir d'un ensemble d'images fournies sous un ordre aléatoire à identifier ces paramètres et à classer les images suivant ces paramètres. D'un point de vue abstrait, ces images sont plongées dans un espace de très grande dimension (la dimension correspondant au nombre de pixels des images). L'algorithme essaye de trouver une paramétrisation de cet espace afin d'en réduire la dimension. La première étape construit un graphe en reliant chaque échantillon à ses voisins les plus proche, relativement à une fonction de distance donnée. Ceci permet d'organiser les échantillons sous forme d'un graphe. Ensuite, plusieurs classes de méthodes différentes permettent d'extraire de l'information depuis ce graphe. Nous en décrivons brièvement les principales classes:

Plongement Linéaire Local (LLE) [RS00] cette méthode tente de créer un plongement qui réalise la meilleure approximation possible des coordonnées barycentrique d'un sommet par rapport à ses voisins. Dans un certain sens, la méthode de Floater[Flo97] mentionnée dans le Chapitre 3 peut être considérée comme un cas particulier de cette approche.

Isomap[TdSL00] calcule les distances géodésiques entre chaque paire de sommets du graphe, et utilise la méthode MDS[You85] (*multidimensional scaling*) pour calculer un plongement qui réalise la meilleure approximation possible de toutes ces distances. La méthode MDS minimise une fonction objectif qui mesure la déviation entre les distances géodésiques mesurées dans l'espace initial de grande dimension, et les distances Euclidiennes mesurées dans l'espace d'arrivée de plus petite dimension. Cette fonction objectif appelée GDD (Geodesic Distance Deviation) est minimisée en calculant les vecteurs propres de la matrice $D = (d_{i,j})$ où $d_{i,j}$ dénote la distance géodésique mesnurée entre le sommet i et le sommet j. Ceci peut être considéré comme la version multivariée de l'Equation 4.22 qui caractérise le vecteur de Fiedler. Dans le contexte de la paramétrisation de surface, la méthode Isomap/MDS a été utilisée récemment[ZKK02], pour être ensuite améliorée sous la forme de la méthode dite *ISO-charts*[ZSGS04], intégrée à l'API DirectX 10 de Microsoft. L'arrangement en 2D des cartes est calculé par une variante de notre algorithme [LPRM02].

Nous comprenons maintenant que les vecteurs propres du Laplacien ou de la matrice des distances géodésiques jouent un rôle important dans la détermination de paramètres significatif à partir de données non-structurées. Si nous imaginons l'ACP (analyse en composantes principales), les vecteurs propres de la matrice de co-variance donnent les hyperplans les mieux appropriés pour projeter les données en minimisant la perte d'information. Dans le domaine de la reduction de dimension, nous cherchons des "vecteurs propres" plus souples, capables de s'adapter aux détails non-linéaires présents dans les données. Par exemple, dans le cas de la méthode MDS, les vecteurs propres sont calculés de manière à construire un espace qui immite au mieux la structure métrique de la surface initiale,

[RS00] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

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

[TdSL00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[You85] F. W. Young. Multidimensional scaling. *Encyclopedia of Statistical Sciences*, 5:649–658, 1985.

[ZKK02] Gil Zigelman, Ron Kimmel, and Nahum Kiryati. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2), 2002.

[ZSGS04] Kun Zhou, John Snyder, Baining Guo, and Heung-Yeung Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Symposium on Geometry Processing*, pages 47–56, 2004.

capturée par la matrice $D = (d_{i,j})$ de toutes les distances géodésiques entre les paires de sommets du graphe.

Plutôt que d'utiliser une matrice dense D, les méthodes fondées sur le Laplacien du graphe n'utilisent que des voisinages locaux. Par conséquent, la matrice correspondante est très creuse, et en extraire les vecteurs propres requiert moins de calculs. Dans le cas où le Laplacien de graphe est symétrique (à savoir $a_{j,i} = a_{i,j}$), ce qui est le cas pour la pondération uniforme ($a_{i,j} = 1$), les vecteurs propres forment une base orthogonale, et peuvent donc être utilisés pour encoder des fonctions. Ceci a été utilisé pour définir un encodage compact de la géométrie de la surface[KG00a]. L'idée de base de cette dernière méthode de compression consiste à encoder dans le fichier la topologie du maillage ainsi que les coefficients de la géométrie projetée sur les vecteurs propres. Le décoder reconstruit la base de vecteurs propres et les multiplie par les coefficients stockés dans le fichier. Une revue complète des méthodes de compression spectrale et de leur lien avec les méthodes de partitionnement de graphe a été écrite par les mêmes auteurs[Got03].

La théorie spectrale des graphes permet également de mettre en évidence de nouvelles méthodes pour construire des plongements de graphes valides. Par exemple, la notion de nombre de Colin-de-verdière[dV90] a été utilisée pour construire des plongements de graphes sphériques[GGS03b] pour des maillages de genre 0. D'autres méthodes ont également été décrite dans des objectifs de visualisation de grands graphes[Kor02]. La théorie spectrale peut également être utilisée pour calculer des invariants topologiques, tels que les nombres de Betti[Fei96]. Comme nous pouvons le voir dans ce petit état l'art des applications de la théorie spéctrale des graphes, il s'agit là d'un domaine très riche, permettant d'exploiter les informations à la fois topologiques et géométriques contenues dans les Laplaciens de graphes.

Toutefois, comme l'expliquent plusieurs auteurs[ZSGS04], utiliser seulement les connections du maillage peut créer des paramétrisation déformées. Les méthodes fondées sur MDS résolvent le problème en considérant la matrice D de toutes les distances géodésiques entre paires de sommet du graphe. Dans la suite de cette section, nous allons étudier une meilleure prise en compte de la géométrie dans les méthodes fondées sur un Laplacien de graphe, plus légères en calcules que MDS. En particulier, nous allons tenter de mieux comprendre comment la géométrie et la topologie globale de la forme émergent de l'interaction entre voisinage locaux.

Ceci fait typiquement référence à des notions empruntées à la formalisation *continue*, à savoir l'analyse fonctionelle et les opérateurs (brièvement introduits en Section 2.3). La prochaine section montre les liens avec l'opérateur de Laplace-Beltrami (déjà rencontré dans le Chapitre 3), qui apparait dans les équations d'ondes (équation de Helmholtz). Nous montrerons ainsi les liens entre les *ondes stationnaires*, l'analyse de Fourier, et la théorie spectrale des graphes.

[KG00a] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH*, pages 279–286, 2000.

[Got03] Craig Gotsman. On graph partitioning, spectral analysis, and digital mesh processing. In *Shape Modeling International*, pages 165–174, 2003.

[dV90] Y. Colin de Verdiere. Sur un nouvel invariant des graphes et un critere de planarite. *J. of Combinatorial Theory*, 50, 1990.

[GGS03b] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes, 2003.

[Kor02] Y. Koren. On spectral graph drawing, 2002.

[Fei96] J. Feidman. Computing betti numbers via combinatorial laplacians. In *Proc. 28th Sympos. Theory Comput.*, pages 386–391. ACM, 1996.

[ZSGS04] Kun Zhou, John Snyder, Baining Guo, and Heung-Yeung Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Symposium on Geometry Processing*, pages 47–56, 2004.

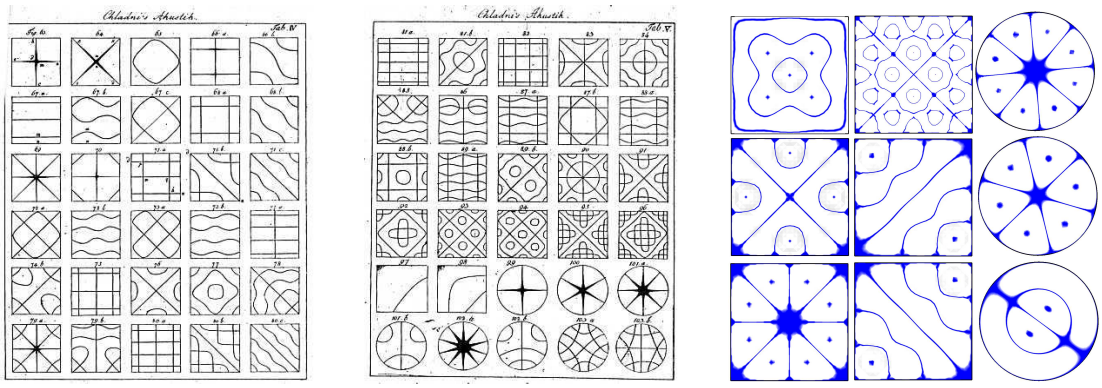


Figure 4.28: *A gauche: A la fin des années 1700, le physicien Ernst Chladni s'est étonné des motifs formés par du sable déposé sur des plaques métalliques vibrantes. A droite: l'équation d'Helmoltz permet de comprendre ces phénomènes et de reproduire les motifs qu'ils génèrent.*

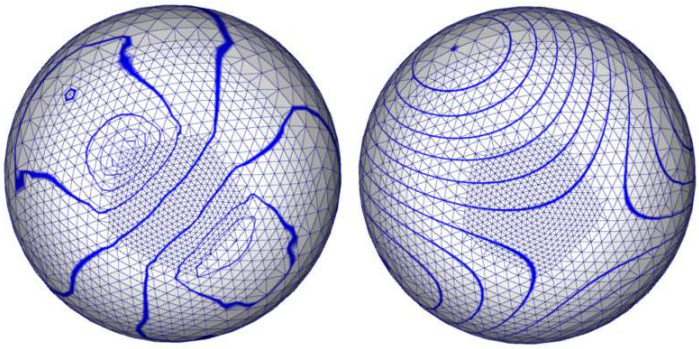


Figure 4.29: *Comparaison entre les lignes de niveau de la quatrième fonction propre du Laplacien sur une sphère échantillonée de manière irrégulière. A Gauche: Laplacien combinatoire; A Droite: Laplacien élément finis (cotangentes).*

4.5.2 Le mode continu

Comme le montre la Figure 4.29-gauche, si nous effectuons les calculs en nous fondant sur la seule connectivité du graphe, le résultat peut devenir fortement dépendant du maillage. Pour cette raison, nous nous intéressons à une formulation du Laplacien qui dépent de la géométrie de la surface (plutôt que de sa discrétisation). Les deux paragraphe suivants présentent un état de l'art de ces deux types de formulation.

Laplaciens combinatoires Il est bien connu que les vecteurs propres du Laplacien de graphe combinatoire définissent une base de fonction similaire à la base de Fourier. Ainsi, comme nous l'avons mentionné plus haut, Karni *et. al.*[KG00b] ont proposé d'utiliser cette idée pour compresser la géométrie. D'autre part, Zhang[Zha04] a étudié plusieurs variantes des Laplaciens de graphe combinatoires ainsi que leurs propriétés pour mettre en oeuvre une compression de maillages similaire à la norme JPEG. Toutefois, comme le remarquent Meyer *et. al*[MDSB03], cette analogie entre le Laplacien de graphes et la transformée en cosinus discrète suppose un échantillonage uniforme du maillage. Par conséquent, pour rendre l'espace de fonction indépendant des irrégularités du maillage, il est préférable de remplacer le Laplacien de graphe combinatoire par une discrétisation qui prend en compte la géométrie.

Discrétisations du Laplacien Le Laplacien combinatoire ne dépend que des connections entre les sommets. Par conséquent, deux plongements différents du même graphe engendrent les mêmes fonctions de base (il ne prend pas la géométrie en compte), et deux maillages différents du même objet engendre des fonctions de base différente (il n'est pas indépendant du maillage). La Figure 4.29-gauche montre le problème sur une sphère échantillonée de manière irrégulière.

Le Laplacien discret, à savoir les fameux poids en cotangentes[PP93b,MDSB03] (c.f. Section 3.2.5) ne souffre pas de ce problème (c.f. Figure 4.29-Droite). Ils ont récemment été utilisés[DBG+06] pour calculer une fonction propre et l'utiliser pour guider un processus de re-maillage quadrilatère. Dans notre contexte, pour séparer les différentes fréquences de la forme, nous avons besoin de calculer de multiples fonctions propres. Dans notre contexte, le Laplacien discret semble perdre une propriété importante de sa version continue: comme ses coefficients ne sont pas symétriques, ses vecteurs propres ne sont plus orthogonaux. Ceci rend la transformée en fréquence difficile à calculer (système linéaire dense à résoudre au lieu d'une simple projection). J'ai proposé une solution ad-hoc[Lev06], consistant à symétriser la matrice de manière artificielle. Des fondations théoriques plus générales ont été utilisées par Reuter *et al.*[RWP05], qui utilisent le formalisme des éléments finis (c.f.

[KG00b] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Zha04] H. Zhang. Discrete combinatorial laplacian operators for digital geometry processing. In *Proc. SIAM Conference on Geometric Design and Computing*, 2004.

[MDSB03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

[PP93b] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1), 1993.

[DBG+06] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1057–1066, New York, NY, USA, 2006. ACM Press.

[Lev06] Bruno Levy. Laplace-beltrami eigenfunctions: Towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications*, 2006.

[RWP05] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-spectra as fingerprints for shape matching. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 101–106, New York, NY, USA, 2005. ACM Press.

Section 2.3.3) pour calculer le spectre (à savoir les valeurs propres), et l'utiliser comme signature pour des applications en classification de formes. D'autre travaux[KR05,KR06] mentionnent également la possibilité d'utiliser le formalisme des éléments finis. Nous montrons plus loin comment cette formulation va nous permettre d'assurer les deux propriétés à la fois, à savoir l'indépendance par rapport au maillage et l'orthogonalité des fonctions propres.

Avant d'aller plus loin sur ces aspects formels, nous allons tenter de donner plus d'intuition sur la nature de ces fonctions propres, en nous inspirant d'analogies avec des phénomènes physiques.

En 1787, le physicien Ernst Chladni publia le livre "Discoveries Concerning the Theories of Music"' (découvertes concerant la théorié de la musique). Dans ce livre, il rapporte les nombreuses observation qu'il a effectuées en mettant en vibration une plaque mince métalique à l'aide d'un archet et en saupoudrant du sable sur cette même plaque. Le sable s'accumule dans certaines zones, ce qui forme des motifs structurés et complexes (c.f. Figure 4.28).

Ce comportement peut s'expliquer par la théorie des *ondes stationnaires*. Quand la plaque métallique vibre, certaines zones restent immobiles, et le sable s'accumule naturellement dans ces zones. Ce phénomène peut se modéliser de la manière suivante, par la composante spatiale de l'équation de Helmholtz qui régit les phénomènes vibratoires:

$$\Delta f = \lambda f \tag{4.23}$$

Dans cette équation, Δ dénote l'opérateur de Laplace, que nous avons déjà rencontré dans le Chapitre 3. Avant d'étudier les fonctions propres de cet opérateur, nous allons tout d'abord expliquer les liens entre ces phénomènes vibratoires et l'analyse de Fourier. Nous verrons ensuite comment généraliser l'analyse de Fourier à des fonctions définies sur des variétés de genre arbitraire.

Analyse de Fourier

Comme dans l'article de Taubin[Tau95b] (à la base du traitement du signal géométrique), nous commencerons par étudier le cas d'une courbe fermée. Toutefois, dans notre cas, nous nous intéressons plus particulièrement à la formulation continue, permettant de mieux prendre la géométrie en compte. Etant donnée une fonction périodique de carré intégrable $f : x \in [0,1] \mapsto f(x)$, ou encore, de manière équivalente, une fonction f définie sur une courbe fermée paramétrée par l'abscisse curviligne, il est bien connu que f peut se décomposer en une suite infinie de sinus et de cosinus de fréquences croissantes:

$$f(x) = \sum_{k=0}^{\infty} \tilde{f}_k H^k(x) \quad ; \quad \left\{ \begin{array}{lcl} H^0 & = & 1 \\ H^{2k+1} & = & \cos(2k\pi x) \\ H^{2k+2} & = & \sin(2k\pi x) \end{array} \right. \tag{4.24}$$

où les coefficients $\tilde{f}_k$ de la décomposition sont donnés par:

$$\tilde{f}_k = < f, H^k > = \int_0^1 f(x) H^k(x) dx \tag{4.25}$$

et où $< .,. >$ dénote le produit interne (i.e. le "produit scalaire" pour les fonctions définies sur $[0,1]$), défini par $< f,g > = \int f(s)g(s)ds$ (c.f. Section 2.3). La base des "harmoniques

[KR05] ByungMoon Kim and Jarek Rossignac. GeoFilter: Geometric Selection of Mesh Filter Parameters. *Computer Graphics Forum*, 24(3):295–302, 2005.

[KR06] ByungMoon Kim and Jarek Rossignac. Localized bi-Laplacian Solver on a Triangle Mesh and Its Applications. *Technical Report*, 2006.

[Tau95b] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1995. ACM Press.

circulaires" H^k est orthonormale par rapport au produit interne $< .,. >$: $< H^k, H^k >= 1$, $< H^k, H^l >= 0$ si $k \neq l$.

L'ensemble des coefficients $\tilde{f}_k$ (Equation 4.25) est appelé la transformée de Fourier de la fonction f. Etant donnés les coefficients $\tilde{f}_k$, la fonction f peut être reconstruite, en appliquant la transformée de Fourier inverse (Equation 4.24). Notre objectif est à présent de généraliser ces notions à des fonctions définies sur des variétés de genre arbitraire. Pour ce faire, nous pouvons considérer les fonctions H^k de la base de Fourier comme les fonctions propres de l'opérateur $-\partial^2/\partial x^2$[16]: les fonctions propres H^{2k+1} (resp. H^{2k+2}) sont associées à la valeur propre $(2k\pi)^2$:

$$\frac{\partial^2 H^{2k+1}(x)}{\partial x^2} = -(2k\pi)^2 \cos(2k\pi x) = -(2k\pi)^2 H^{2k+1}(x)$$

Cette construction peut se généraliser à des variétés de genre arbitraire, en généralisant la notion de dérivée seconde. Cette généralisation correspond à l'opérateur de Laplace et à ses variantes, présentées plus loin.

L'opérateur de Laplace et ses variantes

L'opérateur de Laplace (ou Laplacien) joue un rôle fondamental en physique et en mathématiques. Dans $\mathbb{R}^n$, il est défini comme la divergence du gradient:

$$\Delta = \text{div grad} = \nabla.\nabla = \sum_i \frac{\partial^2}{\partial x_i^2}$$

Intuitivement, le Laplacien généralise la dérivée seconde à des dimensions supérieures, et caractérise les irrégularités d'une fonction. Ainsi, $\Delta f(P)$ mesure les différences entre $f(P)$ et sa valeur moyenne sur un petit voisinage de P.

Généraliser le Laplacien sur des surfaces courbes requiert des calculs difficiles. Ces calculs peuvent être simplifiés par un outil mathématique appelé *calcul extérieur*, présenté brièvement dans la Section 4.1.2. Nous rappelons que le calcul extérieur généralise le gradient par la dérivée extérieure d, et la divergence par la co-dérivée δ, les deux pouvant être définis indépendament de tout système de coordonnées.

En utilisant le calcul extérieur, la définition du Laplacien peut se généraliser à des fonctions définies sur une variété $\mathscr{S}$ munie de la métrique g. Le Laplacien ainsi généralisé est appelé opérateur Laplace-Beltrami:

$$\Delta = \text{div grad} = \delta d = \sum_i \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial x_i} \sqrt{|g|} \frac{\partial}{\partial x_i}$$

où $|g|$ dénote le déterminant de la métrique g. La dérivée extérieure d généralise la notion de gradient et la co-différentielle $\delta = *d*$ généralise la divergence (voir la Section 4.1.2 sur le calcul extérieur). Le terme $\sqrt{|g|}$ peut s'interpreter comme un facteur d'échelle, permettant de prendre en compte l'expression de l'élément différentiel d'aire dA sur $\mathscr{S}$, défini par $dA = \sqrt{|g|} dx_1 \wedge dx_2$.

Finalement, par souci d'exhaustivité, nous mentionnons que l'opérateur de Laplace peut s'étendre à des k-formes différentielles. Cette généralisation s'appelle alors l'opérateur de Laplace-de Rham, défini par $\Delta = \delta d + d\delta$. Nous remarquons que pour les fonctions (i.e. les 0-formes différentielles), le second terme $d\delta$ disparait et le premier terme δd correspond à la première définition[17].

Nous donnons à présent les définitions relatives aux fonctions propres de l'opérateur de Laplace, et expliquons comment ceci permet de généraliser des concepts importants à des variétés de genre arbitraire et à des surfaces triangulées.

[16] en notation complexe, il est également possible de considérer l'operateur dérivée pour les fonctions analytique, qui combine le sinus et le cosinus dans une unique fonction propre.

[17] au signe près. Malheureusement, la convention utilise le même symbole Δ pour les opérateurs Laplace-Beltrami et Laplace-de Rham, ce qui peut causer des confusions de signe.

Figure 4.30: *Quelques fonctions de la base des harmoniques variétés sur le maillage "Gargoyle" du projet AIM at Shape.*

Fonctions propres du Laplacien

Les fonctions propres et valeurs propres du Laplacien sur une variété S sont les paires (H^k, λ_k) qui satisfont:

$$-\Delta H^k = \lambda_k H^k \tag{4.26}$$

A noter que dans cette équation, le Laplacien est précédé d'un signe "$-$", ce qui permet de rendre les valeurs propres positives.

Sur une courbe fermée, les fonctions propres de l'opérateur de Laplace définissent la base de fonctions (sinus et cosinus) de l'analyse de Fourier, comme nous l'avons rappelé en Section 4.5.2. Sur un carré, les fonctions propres correspondent à la base de fonctions de la DCT (Discrete Cosine Transform ou transformée en cosinus discrete), utilisée par exemple par le format d'images JPEG. Finallement, les fonctions propres de l'opérateur Laplace-Beltrami sur la sphère définissent la base des harmoniques sphériques, très utilisée en simulation numérique de la lumière.

Dans ces trois cas simple, deux raisons font des fonctions propres une base de fonction intéressante pour l'analyse spectrale de fonctions définies sur des variétés:

1. Comme le Laplacien est symétrique ($< \Delta f, g >=< f, \Delta g >$), ses fonctions propres sont orthogonales (c.f. Section 2.3), et il est extrèmement simple de projeter une fonction sur la base, à savoir calculer l'équivalent de la transformée de Fourier de cette fonction.

2. Comme nous l'avons mentionné au début de cette Section, pour les physiciens, le problème aux valeurs propres (Equation 4.26) est appelé l'équation Helmoltz, et ses solutions H^k sont les ondes stationnaires. Ceci signifie que les fonctions H^k ont une longueur d'onde ω_k (ou fréquence spatiale) constante définie par $\omega_k = \sqrt{\lambda_k}$.

Ainsi, utiliser les fonctions propres du Laplacien pour construire une base de fonction sur la variété est une manière naturelle d'étendre l'analyse spectrale usuelle à cette variété.

Dans notre cas, la variété est représentée par un maillage, nous allons donc devoir traduire cette construction dans le contexte discret. La première idée venant à l'esprit consiste à appliquer l'analyse spectrale à un Laplacien discret, par exemple la matrice des "cotangentes" calculée en Section 3.2.5.

Toutefois, le Laplacien discret n'est pas une matrice symétrique: le dénominateur du coefficient $a_{i,j}$ correspond à l'aire du voisinage du sommet i, qui ne correspond en général pas à l'aire du voisinage du sommet j. Ainsi, nous perdons la symétrie de la version continue du Laplacien, et les vecteurs propres ne sont plus orthogonaux. Ceci rend difficile la projection d'une fonction sur la base d'harmoniques. Pour cette raison, nous allons clarifier les relations entre le contexte continu (avec des fonctions et des opérateurs) et le contexte discret (avec des vecteurs et des matrices).

4.5.3 Harmoniques Variétés

Pour être capable de projeter des fonctions sur la base des fonctions propres, nous avons besoin de résoudre notre problème aux valeurs propres (Equation 4.26) d'une manière qui préserve leur orthogonalité. Les méthodes à base d'éléments finis (c.f. Section 2.3.3), fondées sur la théorie des espaces de Hilbert, structure les espaces de fonctions considérés en utilisant la notion de produit interne $< .,. >$. Ainsi, en utilisant ce type de formalisme, nous parviendront à mieux comprendre ce que devient l'orthogonalité des fonctions propres lors de la discrétisation.

Elements Finis

La formulation par élément finis de notre problème aux valeurs propre utilise un ensemble de *fonctions de base* utilisées pour exprimer la solution et un ensemble de *fonctions de test* sur lesquelles notre problème aux valeurs propre (Equation 4.26) sera projeté. Nous suivons l'approche classique, consistant à utiliser le même ensemble de fonctions $\Phi^i (i = 1 \dots n)$ pour définir les fonctions de base et les fonctions de test. Nous utilisons Nous utilisons les fonctions "chapeau" (également appelées fonctions P_1 dans la littérature liée aux éléments finis). Ces fonctions sont linéaires par morceaux sur les triangles, et sont telles que $\Phi^i(i) = 1$ et $\Phi^i(j) = 0$ si $i \neq j$. Géométriquement, la fonction Φ^i correspond aux coordonnées barycentriques associées au sommet i dans les triangles qui lui sont incidents. Comme nous l'avons vu dans la Section 2.3.3, la formulation par éléments finis de l'Equation 4.26 relative aux Φ^i revient à trouver les coefficients H_i^k satisfaisant:

$$\begin{aligned} H^k &= \textstyle\sum_{i=1}^n H_i^k \Phi^i \\ \forall j, < -\Delta H^k, \Phi^j > &= \lambda_k < H^k, \Phi^j > \end{aligned}$$

ou sous forme matricielle:

$$-Q\mathbf{h}^k = \lambda B \mathbf{h}^k \tag{4.27}$$

avec $Q_{i,j} =< \Delta\Phi^i, \Phi^j >$, $B_{i,j} =< \Phi^i, \Phi^j >$ et où $\mathbf{h}^k$ dénote le vecteur $[H_1^k, \dots H_n^k]$. La matrice Q est appelée *matrice de rigidité*, et la matrice B *matrice de masse*. Les coefficients de ces matrices sont obtenues de la même manière que dans la Section 3.2.5, traitant de la discrétisation du Laplacien, et sont rappelés ci-dessous:

$$\begin{aligned} &\begin{cases} Q_{i,j} &= \left(\text{cotan}(\beta_{i,j}) + \text{cotan}(\beta'_{i,j})\right)/2 \\ Q_{i,i} &= -\sum_j Q_{i,j} \end{cases} \\ &\begin{cases} B_{i,j} &= (|t| + |t'|)/12 \\ B_{i,i} &= (\sum_{t \in St(i)} |t|)/6 \end{cases} \end{aligned} \tag{4.28}$$

où t, t' sont les deux triangles partageant l'arête (i, j), et où $|t|$ et $|t'|$ dénotent leurs aires. $\beta_{i,j}$ et $\beta'_{i,j}$ correspondent aux deux angles opposés à l'arête (i, j) dans les triangles t et t', et $St(i)$ dénote l'ensemble des triangles incidents au sommet i (voir la Figure 3.13 dans la Section 3.2.5).

Une pratique commune consiste à remplacer cette équation par une approximation, plus facile à calculer:

$$-Q\mathbf{h}^k = \lambda D \mathbf{h}^k \quad \left(\text{or} \quad -D^{-1}Q\mathbf{h}^k = \lambda \mathbf{h}^k\right) \tag{4.29}$$

où la matrice de masse B is remplacée par la matrice de masse condensée D définie par:

$$D_{i,i} = \sum_j B_{i,j} = (\sum_{t \in St(i)} |t|)/3. \tag{4.30}$$

Dans notre cas spécifique (Equation 4.26), non seulement cette approximation simplifie les calculs, mais les erreurs d'approximation qu'elle introduit compense celles relatives à la

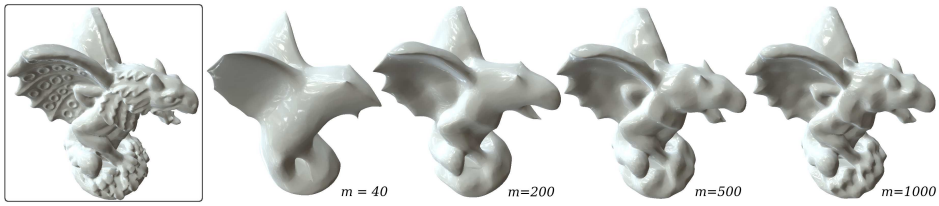


Figure 4.31: *Reconstructions obtenues avec un nombre croissant de fonctions propres*

discrétisation, l'effet global étant d'améliorer la précision du résultat[Dye06,Pra99]. La Figure 4.30 montre quelques une de ces fonctions calculées sur une surface triangulée.

Remarque: Comme nous l'avons remarqué en Section 3.2.5, la matrice $D^{-1}Q$ (Equation 4.29) correspond exactement au Laplacien discret classique[PP93b,MDSB03]. Comme nous le verrons dans la section suivante, la formulation par éléments finis offre dans notre cas l'avantage d'expliquer comment l'orthogonalité des fonctions propres peut être préservée (alors que le fait que la matrice $D^{-1}Q$ ne soit pas symétrique implique que ces vecteurs propres ne soient pas obligatoirement orthogonaux).

Orthogonalité des fonctions propres

La transformée en harmonique variétés que nous développons ici se fonde sur une projection sur la base des fonctions propres, grandement facilitée par le fait que ces fonctions propres soient orthogonales. Nous devons donc clarifier la manière dont l'orthogonalité du Laplacien continu Δ se traduit dans le contexte discret, bien que la matrice de l'Equation 4.29 ne soit pas symétrique. Ceci s'explique simplement en faisant la distinction entre le produit scalaire (discret) entre deux vecteurs, et le produit interne (continu) entre deux fonctions $< G,H >$:

$$< G,H >=< \sum_{i=1}^{n} G_i\Phi^i, \sum_{j=1}^{n} H_j\Phi^j >$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} G_iH_j < \Phi^i,\Phi^j >= \sum_{i=1}^{n}\sum_{j=1}^{n} G_iH_jB_{i,j} = \mathbf{g}^\top B\mathbf{h}$$

Le produit scalaire et le produit interne ne se correspondent que si la matrice de masse B est égale à l'identité, à savoir si la base des fonctions de tests (Φ^i) est orthonormale. Nous remarquons que ce n'est pas le cas dans notre configuration (les fonctions "chapeau" P1 ne sont pas orthogonales). Par conséquent, nous devons utiliser le produit interne $\mathbf{g}^\top B\mathbf{h}$.

Lorsque nous utilisons l'approximation masse condensée, le produit interne devient $\mathbf{g}^\top D\mathbf{h}$, et deux vecteurs $\mathbf{g}$ et $\mathbf{h}$ associés à deux valeurs propres différentes satisfont $\mathbf{g}^\top D\mathbf{h} = 0$. En plus d'être orthogonale relativement à la matrice D, il est facile de rendre la base des fonctions propres orthonormale, en divisant chaque vecteur $\mathbf{h}^k$ par sa norme relative à D $\|\mathbf{h}^k\|_D = (\mathbf{h}^{k\top} D\mathbf{h}^k)^{1/2}$.

En résumé, calculer les fonctions propres du Laplacien en utilisant une formulation de type éléments finis et l'approximation masse condensée se formule sous la forme de l''Equation 4.29. Résoudre cette équation revient à chercher les vecteurs propres et les

[Dye06] Ramsey Dyer. Mass weights and the cot operator (personal communication). Technical report, Simon Fraser University, CA, 2006.

[Pra99] G. Prathap. Towards a science of fea: Patterns, predictability and proof through some case studies. *Current Science*, 1999.

[PP93b] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1), 1993.

[MDSB03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

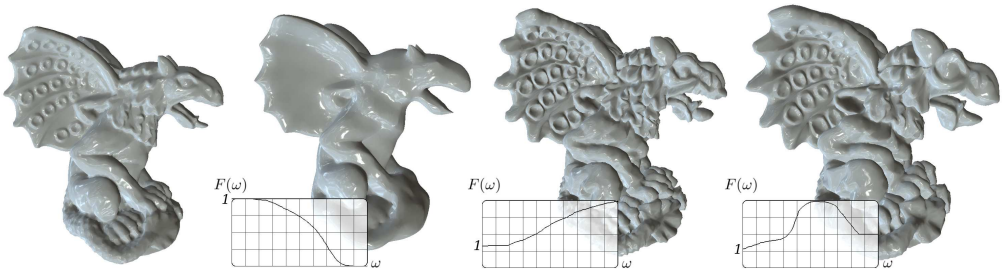


Figure 4.32: *Filtre passe-bas, passe-haut et passe-bande. Le filtre peut être changé interactivement par l'utilisateur, et la surface est mise à jour en temps réel.*

valeurs propres d'une grande matrice creuse. Nous avons mis au point un algorithme numérique pour résoudre ce problème pratique, décrit dans notre article [VL]. Cet algorithme calcule une série de vecteurs propres et valeurs propres $(\mathbf{h}^k = [H_1^k, H_2^k, \ldots H_n^k], \lambda_k)$, que nous appelons la base des harmoniques variétés (BHV). La BHV associée au produit interne nous permet à présent de définir la transformée en harmoniques variété (THV) et son inverse (THV^{-1}).

4.5.4 Transformée en Harmoniques Variétés

Maintenant que nous avons vu comment calculer une base d'harmoniques variété en résolvant l'Equation 4.29 et que nous avons compris la différence entre le produit scalaire et le produit interne, nous pouvons en déduire l'expression de la THV (qui va de l'espace géométrique vers l'espace fréquentiel) et de la THV inverse (qui va de l'espace fréquentiel vers l'espace géométrique). Nous expliquerons également comment utiliser ces notions pour implanter un algorithme de filtrage géométrique.

4.5.5 Calcul de la THV

La géométrie x (resp. y,z) d'une surface triangulée S peut être considérée comme une fonction linéaire par morceaux, définie comme combinaison linéaire des fonctions de base Φ^i: $x = \sum_{i=1}^n x_i \Phi^i$ où x_i dénote la coordonnée x du sommet i.

Calculer la THV de la fonction x revient à convertir x depuis la base des fonctions "chapeau" (Φ^i) (espace géométrique) vers la base des harmoniques variété (H^k) (espace fréquentiel). Comme cette dernière base est orthonormale, ceci peut être simplement réalisé en projetant x sur la base, à l'aide du produit interne. La THV de x est un vecteur $[\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_m]$, défini par:

$$\tilde{x}_k = < x, H^k > = \mathbf{x}^\top D \mathbf{h}^k = \sum_{i=1}^n x_i D_{i,i} H_i^k \tag{4.31}$$

où $\mathbf{x}$ dénote le vecteur $x_1, x_2, \ldots x_n$.

Calcul de la THV inverse

La THV inverse, qui transforme depuis l'espace fréquentiel vers l'espace géométrique, se déduit simplement de l'expression de x dans la base (H^k). La coordonnée x reconstruite en un sommet i devient alors:

$$x_i = \sum_{k=1}^m \tilde{x}_k H_i^k \tag{4.32}$$

La Figure 4.31 montre la géométrie reconstruite à partir de la THV d'une surface, en utilisant un nombre variable m d'harmoniques. Comme nous pouvons le voir, les premières

fonctions H^k capturent la forme générale de la surface, et les harmoniques suivantes correspondent aux détails. Nous montrons à présent comment implanter un filtrage géométrique en modifiant la THV inverse.

Filtrage géométrique

Une fois que la géométrie est convertie dans la base des harmoniques variété, chaque composante $(\tilde{x}_k, \tilde{y}_k, \tilde{z}_k)$ de la THV correspond à une fréquence spatiale individuelle ω_k. Dans le cas d'une courbe fermée, comme nous l'avons expliqué en Section 4.5.2, nous avons $-\partial^2 \sin(\omega x)/\partial x^2 = \omega^2 \sin(\omega x)$. Nous en déduisons la relation entre la fréquence spatiale ω_k et la valeur propre associée λ_k, à savoir $\omega_k = \sqrt{\lambda_k}$. Ceci reste valable dans le cas des fonctions propres de l'opérateur de Laplace-Beltrami défini sur une surface.

Un filtre géométrique est une fonction $F(\omega)$ qui définit le facteur d'amplification devant être appliqué à chaque fréquence spatiale ω.

Comme toutes les fréquences spatiales sont parfaitement séparées par la THV, appliquer un filtre $F(\omega)$ revient à effectuer un produit dans l'espace fréquentiel. Ainsi, les coordonnées filtrées x_i^F (resp y_i^F, z_i^F) du sommet i sont données par:

$$x_i^F = \sum_{k=1}^{m} F(\omega_k)\tilde{x}_k H_i^k = \sum_{k=1}^{m} F(\sqrt{\lambda_k})\tilde{x}_k H_i^k$$

En pratique, comme nous tronquons la base des harmoniques variété à la fréquence $\omega_m = \sqrt{\lambda_m}$, les détails géométriques plus petits ne sont pas représentés dans la THV. Toutefois, il est possible de manipuler l'information de haute fréquence, en stockant dans chaque sommet la différence x_i^{hf} (resp. y_i^{hf}, z_i^{hf}) entre la géométrie originale et sa projection sur la base d'harmonique. Cette différence est donnée par:

$$x_i^{hf} = \sum_{k=m+1}^{\infty} \tilde{x}_k H_i^k = x_i - \sum_{k=1}^{m} \tilde{x}_k H_i^k$$

Il est alors possible d'appliquer le filtre aux composantes de haute fréquence, en les réinjectant dans la THV inverse, de la manière suivante:

$$x_i^F = \sum_{k=1}^{m} F(\omega_k)\tilde{x}_k H_i^k + f^{hf} x^{hf} \text{where } f^{hf} = \frac{1}{\omega_M - \omega_m} \int_{\omega_m}^{\omega_M} F(\omega) d\omega \tag{4.33}$$

Dans cette équation, le terme f^{hf} dénote la valeur moyenne du filtre F sur l'intervalle $[\omega_m, \omega_M]$, où ω_M dénote la fréquence maximume présente dans le maillage, à savoir la fréquence de Nyquist, qui correspond au double de la longueur des arêtes. Les composantes de hautes fréquences se comportent comme un paquet d'ondes qui peut être filtré comme une entité unique, mais qui ne peut plus être décomposé en fréquences individuelles. Dans nos expériences, nous avons utilisé 10 fois la longueur moyenne des arêtes pour définir la fréquence de coupure ω_m.

La Figure 4.32 montre l'application d'un filtre passe-bas, passe-haut et passe-bande. Des fréquences arbitraires peuvent alors être filtrées, sans introduire d'effet de perte de volume dont les méthodes classiques soufrent. De plus, seule la THV inverse dépend du filtre F. Par conséquent, en stockant la base d'harmonique variété et les coefficients de la THV, la solution peut être mise à jour en temps réel lorsque l'utilisateur modifie le filtre F.

Nous avons implanté un algorithme efficace pour calculer la base des fonctions harmponiques et pour réaliser ces opérations de filtrage [VL]. Cet algorithme permet de manipuler de grands maillages, jusqu'à 1 million de sommet, ce qui repousse de deux ordres de magnitude les limites courament admises pour le traitement spectral de la géométrie (limité jusqu'à présent à quelques milliers de sommets). La Figure 4.33 montre les résultats obtenus avec ces grands maillages.

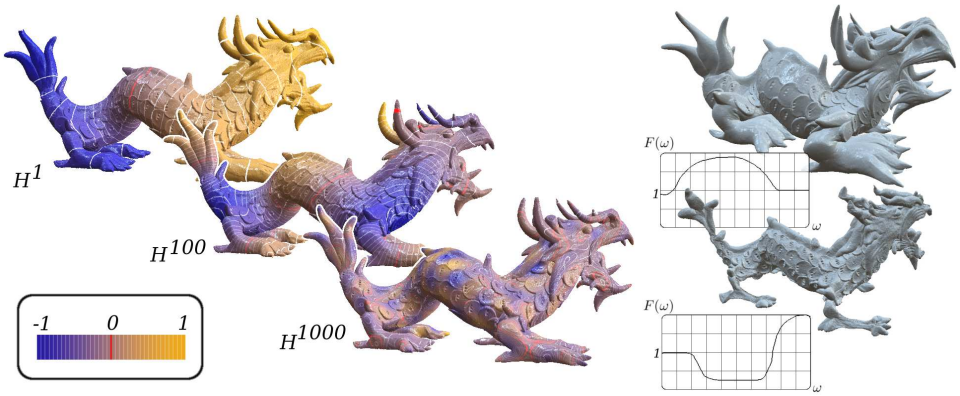


Figure 4.33: *Passage à l'échel du traitement spectral de la géométrie: la base d'harmoniques variété calculée sur un maillage comportant 1 million de sommets, et le filtrage en mémoire externe.*

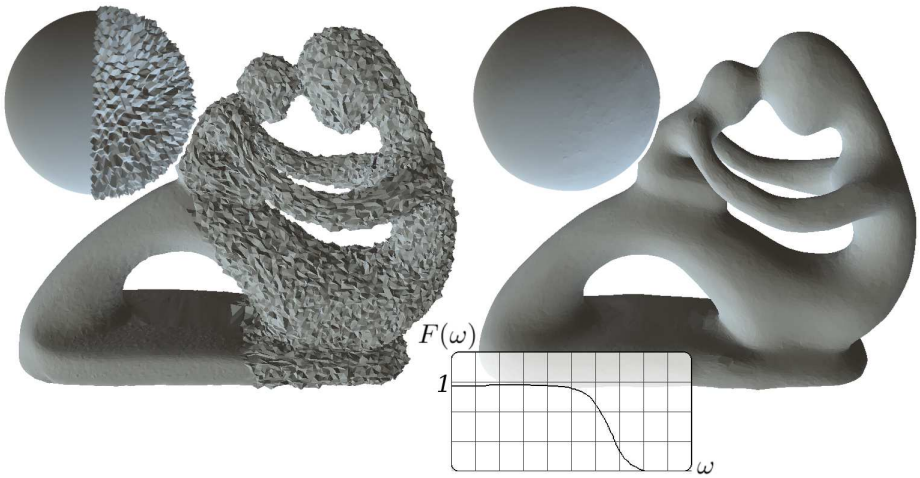


Figure 4.34: *A gauche: une sphère et un objet de genre 4 avec du bruit aléatoire ajouté. A droite: le résultat de l'application d'un filtre passe-bas.*

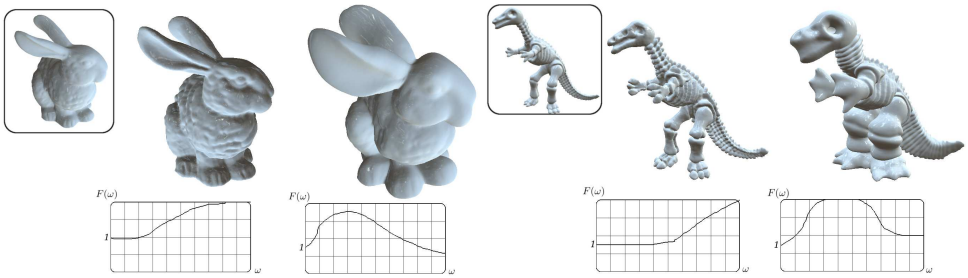


Figure 4.35: *Filtrage du lapin de Stanford et du dinosaure de Cyberware. Nous obtenons des résultats similaires à l'approche "Geofilter" de Kim* et. al*, avec l'interactivité et sans les problèmes de perte de volume.*

Figure 4.36: *Filtrage des couleurs attachées aux sommets d'un objet de topologie arbitraire.*

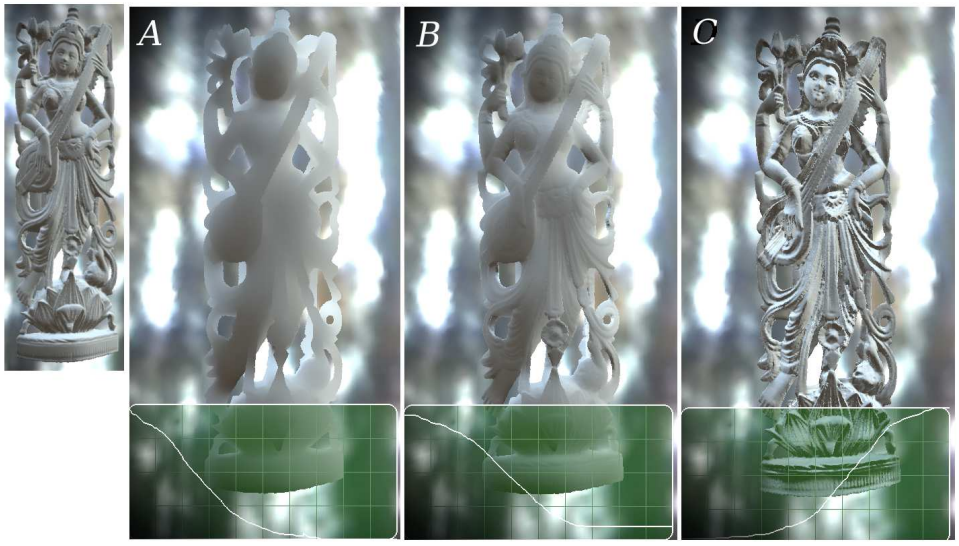


Figure 4.37: *Une approche à base de traitement du signal pour la mise au point de modèles d'éclairement. A: forte diffusion; B: diffusion modérée; C: exagération des détails.*

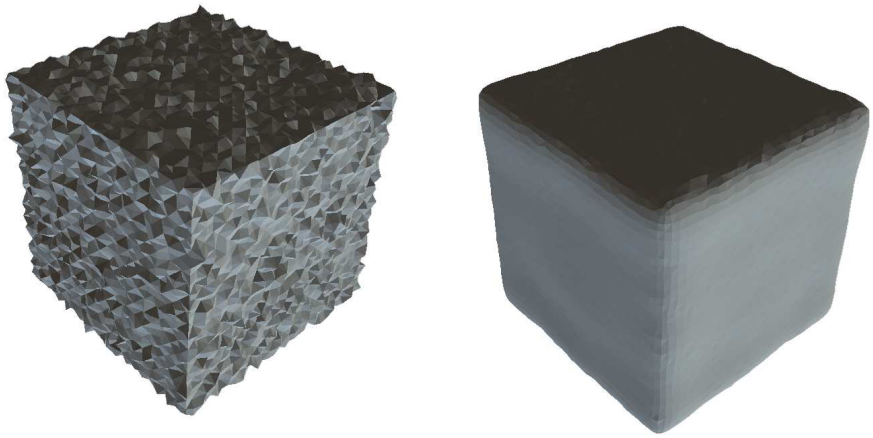


Figure 4.38: *Les arêtes vives génèrent un grand nombre d'harmoniques de fréquences différentes, et sont donc très difficiles à préserver lors du filtrage.*

Notre méthode de filtrage fondée sur les harmoniques variétés peut s'appliquer à des objets de topologie arbitraire. La Figure 4.34 montre un filtre passe-bas utilisé pour enlever du bruit de haute fréquence d'une sphère et d'un objet de genre 4. Le filtre passe-bas préserve quasiment la symétrie de la sphère. La Figure 4.35 démontre l'utilisation de notre méthode dans un contexte similaire à celui de la méthode *geofilter*[KR05]. Dans notre cas, le filtrage s'effectue interactivement. De plus, contrairement à cette dernière méthode, notre filtrage dans le domaine fréquentiel n'introduit pas de pertes de volumes parasites.

Nous montrons également notre méthode appliquée à différents attributs attachés aux sommets d'une surface (Figure 4.36) avec un filtre passe-haut et un filtre passe-bas. La Figure 4.37 montre comment notre méthode appliquée au vecteur normal permet de simuler différents effets lumineux. Appliquet un filtre passe-bas au vecteur normal est approximativement équivalent au filtrage de l'intensité lumineuse. Ceci permet d'obtenir une approximation très simple des phénomènes de diffusion sub-surface. L'utilisateur peut facilement changer le profil du filtre interactivement. Une fois que l'utilisateur est satisfait du résultat, seul un vecteur supplémentaire par sommet est nécessaire pour effectuer le rendu. L'effet est simplement obtenu en remplaçant le vecteur normal par sa version filtrée. Réciproquement, appliquer un filtre passe-haut aux vecteurs normaux permet d'obtenir un résultat similaire à la méthode d'*éclairement exagéré*[RBD06], utilisée pour mettre en évidence les détails. L'utilisateur peut, en modifiant interactivement la fonction de filtrage, obtenir n'importe quel type d'éclairement intermédiare entre ces deux extrèmes.

Une limitation classique des méthodes de filtrage fondées sur l'analyse de Fourier concerne la prise en compte des arêtes vives (c.f. Figure 4.38). De telles arêtes vivent génèrent un grand nombre d'harmoniques de fréquences différentes dans le spectre, et les préserver devient donc un problème spécialement difficile. Une extension de notre méthode, fondée sur une version anisotrope de l'opérateur de Laplace-Beltrami, est susceptible de donner de meilleurs résultats pour ce type de surface, en améliorant la localisation en fréquence des arêtes vives.

Notre méthode est susceptible de trouver des applications dans différents contextes, tels que la segmentation, le "tatouage" de maillage (permettant de cacher de l'information dans un maillage), ou la reconstruction de surfaces. Comme notre solveur (décrit dans notre article [VL]) permet de calculer des milliers vecteurs propres pour des maillages comportant des millions de sommets, nous l'avons appliqué au Laplacien discret combinatoire,

[KR05] ByungMoon Kim and Jarek Rossignac. GeoFilter: Geometric Selection of Mesh Filter Parameters. *Computer Graphics Forum*, 24(3):295–302, 2005.

[RBD06] Szymon Rusinkiewicz, Michael Burns, and Doug DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

afin d'expérimenter la méthode de compression spectrale de Karni *et al.*shortciteKarni00 sans partitionner l'objet. Dans nos expérience, il est apparu qu'un nombre conséquents de coefficients THV étaient nécessaires pour reconstruire la géométrie avec suffisament de précision. Ceci peut s'expliquer par le fait que la base des harmoniques variété n'est pas localisée spatialement. Ainsi, non seulement le partionnement du maillage effectué par Karni *et. al* permet de réduire les temps de calcul, mais de manière plus importante, ceci permet d'obtenir une base de fonction localisée à la fois dans le domaine fréquentiel et dans le domaine spatial[18], le prix à payer étant la perte de la continuité des fonctions de base. Ceci nous conduit à penser que la définition de fonctions de bases continues pour les maillages, localisées à la fois dans le domaine spatial et dans le domaine fréquentiel[GKS02b], que nous pourrions dénommer *ondelettes variété*, sera un sujet de recherche important et passionant dans un avenir très proche.

[18] c'est également pour cette raison que le format JPEG utilise des petits blocs plutôt que d'appliquer la transformée cosinus discrète à toute l'image

[GKS02b] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms: a simple framework for adaptive simulation. In *Processings SIGGRAPH*, 2002.

Chapitre 5

Perspectives

Les résultats de recherche présentés dans ce mémoire s'inscrivent dans la démarche présentée au début de ce mémoire, à savoir:

- ◇ étudier des problèmes géométriques,
- ◇ les formaliser sous forme de problèmes variationels ou d'équations aux dérivées partielles,
- ◇ caractériser, quand cela se révèle possible, les propriétés de la solution,
- ◇ mettre au point un algorithme d'optimisation

Cette approche a été efficace pour permettre à mon équipe d'inventer plusieurs algorithmes efficaces et les publier dans les principaux journaux et conférences du domaine du graphisme. Toutefois, pour aller plus loin, il me parait important de parvenir à prendre plus de recul, en étudiant l'historique du domaine, et d'en déduire quels sont les aspects les plus important. Nous verrons ainsi le rôle majeur joué par la notion d'*abstraction*, et dégagerons un programme de recherche, à savoir la réalisation d'algorithmes permettant de calculer ce que nous appelons des *bases de fonctions mobiles*, qui structurera les activités de l'équipe

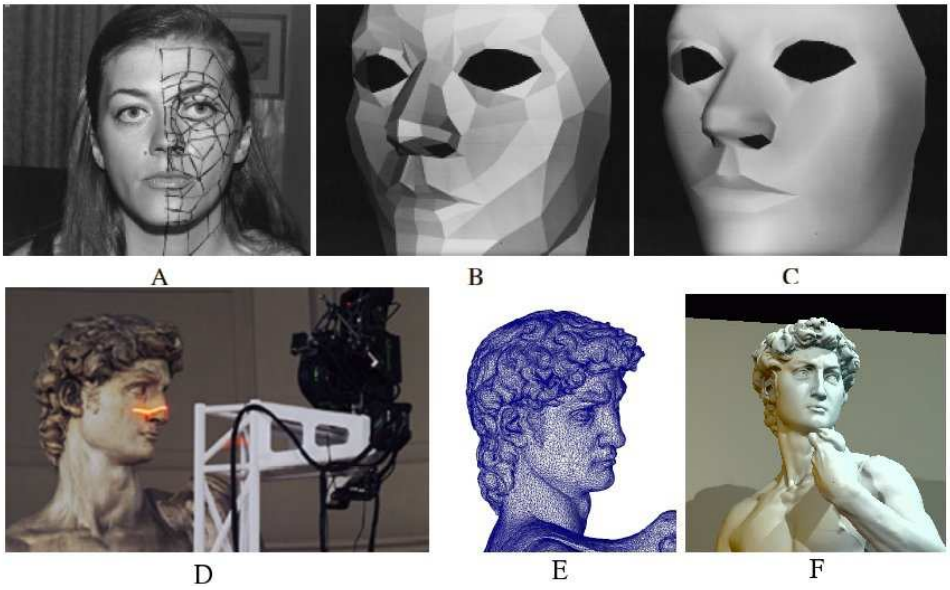


Figure 5.1: *Le graphisme par ordinateur dans les années 70 (A,B,C) et à présent (D,E,F). Une classe entière de problèmes, revenant à construire la bonne abstraction des données, reste ouverte.*

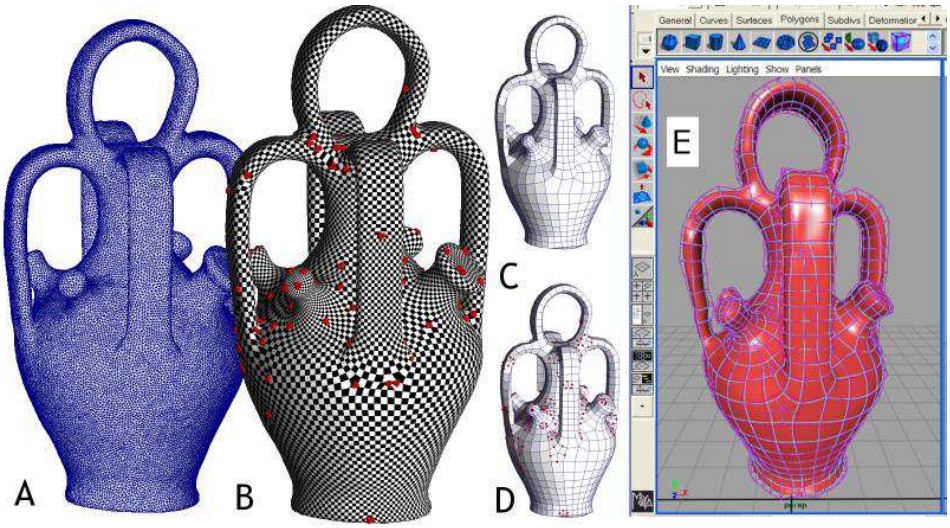


Figure 5.2: *Retrouver une équation d'un objet à partir d'un échantillonage: Notre méthode de paramétrisation globale périodique permet à partir d'un objet scanné (A) de reconstruire une abstraction géométrique (B) pouvant ensuite être instanciée sous forme de surface de haut niveau (C,D,E).*

pour les quelques années à venir.

5.1 Importance de l'abstraction

Au tout début de l'émergence du graphisme par ordinateur, il était extrèmement difficile d'acquérir des données. La Figure 5.1-A, extraite de la thèse d'Henri Gouraud, montre un tel processus d'acquision, réalisé à la main en dessinant directement des polygones sur le visage à digitaliser, et en mesurant les coordonnées des points sur des photographies. Ensuite, les premières recherches effectuées dans le domaine ont concerné l'amélioration du rendu graphique des images de synthèses réalisées à partir de ces modèles (Figure 5.1 B,C). A présent, l'acquisition et la visualisation de modèles complexes sont devenues bien plus facile, avec d'une part la puissance de calcul accrue des ordinateurs, et d'autre part l'avènement des techniques d'acquisition 3D. La figure 5.1 D,E,F montre une digitalisation 3D réalisée par le "Digital Michelangelo Project" de Stanford.

Toutefois, une classe entière de problèmes reste ouverte. Ces problèmes concernent le choix de la bonne *représentation* des objets. Par exemple, un scanner 3D est capable de produire à partir d'un objet réel une représentation informatique de cet objet (Figure 5.1-E). Malheureusement, cette représentation est disponible sous forme brute, peu structurée (à savoir, des millions de points reliés par des triangles). Ces représentations sont moins bien adaptées à la forme de l'objet que les triangles judicieusement choisis par Henri Gouraud lors de la digitalisation manuelle de sa femme (Figure 5.1-A). Ceci rend difficile (voire impossible) la visualisation en temps réel de ces objets, et surtout l'utilisation de ces objets dans le cadre de simulation numérique (par exemple par éléments finis), sensible à la forme des triangles. L'un des problèmes ouverts en modélisation 3D consiste donc à retrouver de manière automatique une représentation de plus haut niveau de l'objet, par exemple sous la forme d'un ensemble déquations. Henri Gouraud (cité plus haut) et Malcolm Sabin (un autre pionnier du domaine), mentionnent encore tous deux dans leurs conférences ce problème comme un problème ouvert.

Une manière plus générale de considérer cette problématique est de considérer l'ensemble d'équations ainsi reconstruit comme une *abstraction* de l'objet initial. L'importance de l'abstraction en image de synthèse est souvent mentionnée par Pat Hanrahan, l'un des pionniers du domaine. Le projet de recherche que je souhaite développer avec mon équipe peut alors se résumer ainsi: *Etant donné un objet 3D et ses propriétés physiques, définir la bonne abstraction ainsi que l'algorithme pour la construire*. Nous détaillons ci-dessous ces différents aspects:

- ◇ *les propriétés physiques:* notre recherche ne se limite pas au seul domaine de la géométrie. En effet, comme nous avons pour objectif de développer des applications en visualisation scientifique et en simulation numérique, il convient de considérer également des propriétés physiques des objets. Comme nous sommes passionnés par l'image de synthèse, et que l'un des aspects fondamentaux de cette discipline concerne les interactions lumière-matière, il est naturel de nous intéresser particulièrement aux propriétés photométriques des objets;
- ◇ *la bonne abstraction:* par "bonne", nous entendons "optimale pour une application donnée". L'application en question peut être par exemple l'affichage interactif, ou encore la simulation numérique de phénomènes physiques. Pour ces applications, il est relativement facile de formaliser un critère d'optimalité: les applications d'affichage souhaitent réaliser un compromis entre fidélité et vitesse, alors que les applications de calcul numérique recherchent la précision et la stabilité numérique;
- ◇ *l'algorithme pour construire cette abstraction:* une fois que le critère d'optimalité est défini, il reste à inventer un algorithme pour convertir la représentation brute des données en une représentation abstraite, satisfaisant le critère d'optimalité.

5.2 Bases de fonctions mobiles

Une fois ces objectifs définis, nous pouvons progresser dans la formalisation du problème. Nous considérons le problème de l'approximation des solutions d'équations aux dérivées partielles linéaires (ou d'équation intégro-différentielles dans le cas de l'illumination globale). Dans sa forme la plus générale, cette classe de problème peut s'exprimer sous la forme $Lf = g$, où L est un opérateur linéaire, f est la fonction inconnue recherchée, et la fonction g est le second membre. La formulation par élements finis classique (Galerkin) projette cette équation sur une base de fonctions (ϕ^k). Dans notre cas, cette base de fonction (ϕ^k) est également utilisée pour représenter la solution, ce qui conduit à la formulation suivante:

$$\begin{aligned} f &= \sum_{k=1}^{n} \alpha_k \phi^k \\ \forall j, < Lf, \phi^j > &= < f, \phi^j > \end{aligned}$$

qui peut s'exprime sous forme matricielle:

$$Q\alpha = \lambda B\alpha$$

avec $Q_{i,j} =< L\phi^i, \phi^j >$, $B_{i,j} =< \phi^i, \phi^j >$ et où α dénote le vecteur $[\alpha_1, \alpha_2, \ldots \alpha_n]$. La matrice Q est appelée *matrice de rigidité*, et B est appelée *matrice de masse*.

Dans le cas par exemple de la simulation numérique de la lumière, la solution f peut exhiber de nombreuses discontinuités (près des frontières d'ombre) et des variations importantes (près des reflets spéculaires). Pour bien adapter la base de fonctions ϕ^i aux variations de la fonction f, une méthode courament utilisée consiste à raffiner adaptativement le support, suivant un critère d'erreur calculé localement. Ceci revient à appliquer la méthode de Galerkin en utilisant une base d'ondelettes. L'inconvénient majeur de ce type de méthode de raffinement est que dans le cas où les variations et discontinuités de la fonction ne sont

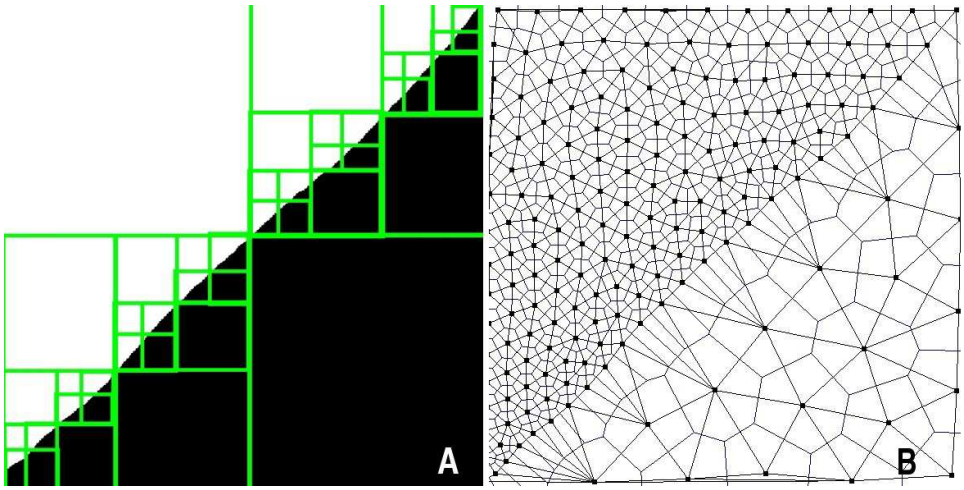


Figure 5.3: *A: les méthodes de simulation par éléments finis et raffinement hiérarchique ont des difficultés près des discontinuités et fortes variations. B: nous développons une nouvelle famille de méthodes, fondées sur la notion de base de fonction mobile.*

pas orientées comme le support (Figure 5.3-A), un grand nombre de fonctions de base peut être généré (dans le cas de la figure, il faut théoriquement raffiner jusqu'à l'infini afin de capturer la diagonale).

Dans notre approche, l'approximation de la solution est également représentée dans une base de fonctions (ϕ^k), mais toutes ces fonctions (ϕ^k) dépendent d'un vecteur de paramètre supplémentaires p, *inconnus*. Par exemple, supposons que la fonction f soit une fonction de deux variables linéaire par morceaux, définie sur les facettes d'une triangulation de Delaunay. Dans cette configuration, nous avons exactement une fonction de base ϕ^k par sommet k de la triangulation (les fonctions ϕ^k sont appelées $P1$ dans la littérature des éléments finis). Le vecteur de paramètres supplémentaire p correspond alors aux coordonnées (x_k, y_k) de tous les sommets de la triangulation. La Figure 5.3-B montre l'idée de cette méthode: les fonctions de base ϕ^k vont s'adapter à la discontinuité, alors représentée sous forme bien plus compacte. La fonction f s'exprime alors sous la forme $f(x) = \sum \alpha_i \phi^i(p,x)$. Nous allons tout d'abord étudier le problème consistant à minimiser le résidu $F(\alpha, p) = \|Lf - g\|^2$:

$$F(\alpha, p) = \|Lf - g\|^2 = \alpha^t M \alpha - 2\alpha^t c + < g, g >$$

$$\text{avec} \quad m_{i,j} = < L\phi^i, L\phi^j > \quad ; \quad c_i = < L\phi^i, g >$$

La difficulité principale provient des dépendances non-linéaires introduites par le vecteur de paramètres supplémentaires p. La deuxième difficulté est que, dans le cas général, l'expression – à savoir l'équation – correspondant à la fonction objectif F dépend de la valeur des paramètres p (la fonction F est elle même définie par morceaux, dans l'espace des paramètres p). Pour calculer les points fixes de F, nous avons introduit un schéma général, fondé sur l'algorithme de Newton:

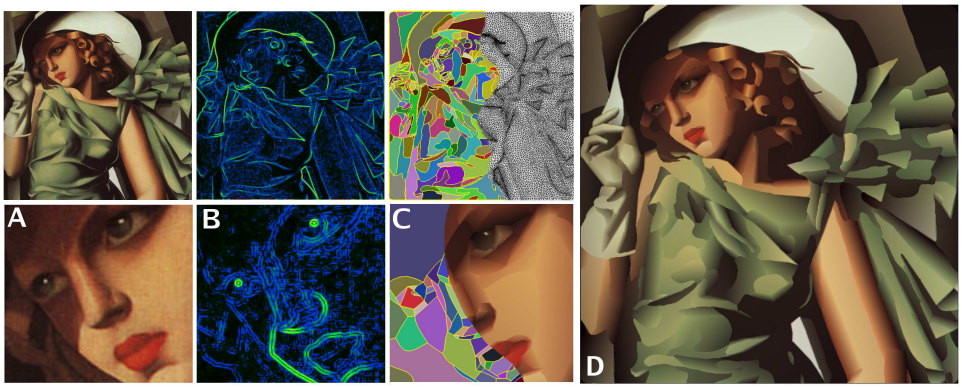


Figure 5.4: *Approximation et vectorisation d'images avec des bases de fonctions mobiles. En partant d'une image bitmap (A), notre algorithme ARDECO calcule une image de salience (B), puis une triangulation et une segmentation adaptées à cette image de salience (C). Le résultat est une "équation de l'image", bien plus compacte (150 équations au lieu d'un million de pixels), et compatible avec les formats d'images vectorielles modernes (PDF, SVG, Flash . . .).*

$\alpha \leftarrow \alpha_0 \quad ; \quad p \leftarrow p_0$
tant que $\|\nabla F(\alpha, p)\| > \varepsilon$

$$\textbf{résoudre} \quad \begin{pmatrix} \nabla^2_{\alpha,\alpha} F & \nabla^2_{\alpha,p} F \\ \nabla^2_{p,\alpha} F & \nabla^2_{p,p} F \end{pmatrix} \begin{pmatrix} \delta_\alpha \\ \delta_p \end{pmatrix} = \begin{pmatrix} -\nabla_\alpha F \\ -\nabla_p F \end{pmatrix}$$

$\alpha \leftarrow \alpha + \delta_\alpha \quad ; \quad p \leftarrow p + \delta_p$
fin$//tantque$

En pratique, nous allons instancier ce schéma général dans différentes configurations, et ainsi définir de nouveaux algorithmes intéressants. Nous pouvons ainsi imaginer un programme de recherche, consistant à étudier des configurations de difficultés croissantes:

- ◇ 1D: Equation de Laplace, Equation de Poisson ;
- ◇ 1D + t: Equation de la chaleur ;
- ◇ 2D,$L = Id$: problème d'approximation numérique d'images, vectorisation d'images ;
- ◇ 2D + t: simulation de fluides, Navier stokes ;
- ◇ 3D, $L = Id$: conversion de maillages en surfaces polynomiales ;
- ◇ 3D: simulation de la lumière; optimiser les paramètres p revient à calculer une approximation complexe de visibilité par une approche numérique;
- ◇ 3D + t: simulations dynamiques

J'ai débuté la réalisation de ce programme de recherche avec les membres de mon équipe:

Le cas $2D, L = Id$, correspondant à un problème d'approximation d'image, nous a permis avec Gregory Lecot, doctorant de l'équipe, de développer l'algorithme ARDECO (Automatic Region Detection and Conversion) [LL06]. En partant d'une image bitmap (Figure 5.4-A), notre algorithme calcule tout d'abord une *image de salience* (Figure 5.4-B) mesurant la densité de détails de l'image, puis une triangulation dont la densité est adaptée à cette image de salience. Enfin, l'algorithme regroupe les triangles en se fondant sur le fait que les variations de la couleurs dans les triangles d'une même région peuvent être approximées par la même équation polynomiale (Figure 5.4-C). Ceci permet de convertir l'image bitmap initiale en un ensemble d'équations polynomiales, délimitées par des Splines cu-

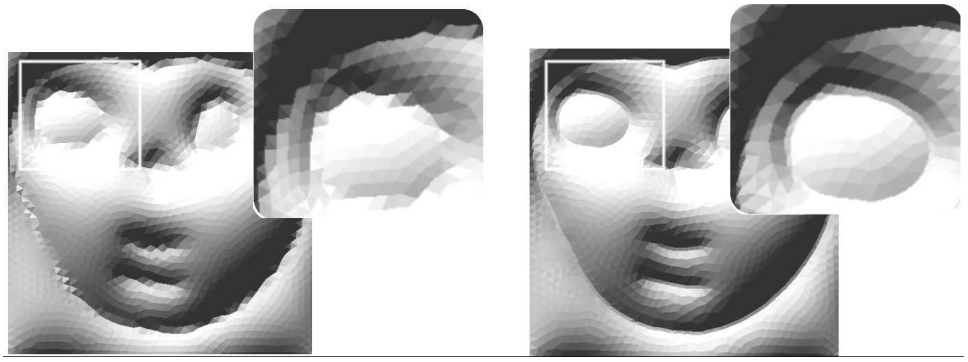


Figure 5.5: *Résolution par élément finis de l'équation de Poisson, à gauche sans bases de fonctions mobiles, et à droites avec des bases de fonctions mobile. L'amélioration de la discrétisation est très nette, en particulier près des frontières.*

biques (5.4-D). Cette représentation, plus compacte, correspond aux formats de fichiers vectoriels (PDF, SVG, Flash . . .), et également utilisée dans les interfaces graphiques modernes (MacOS X, Windows Vista et AIGLX dans le monde Unix).

Avec Bruno Vallet, doctorant de l'équipe, nous nous penchons actuellement sur le cas $2D, L = \Delta$ avec un second membre (équation de Poisson $\Delta f = g$) et conditions de Dirichlet. La Figure 5.5 montre le résultat obtenu (le second membre correspond à quelques lignes tracées à la main, symbolisant "la tête à toto"). Avec les bases de fonctions mobiles, l'algorithme présenté dans cette section permet de mieux capturer les fortes variations de la solution, en alignant naturellement les triangles de manière optimum le long de ces zones de forte variations.

5.3 Bilan et perspectives

5.3.1 Bilan

Tout au long de ce mémoire, nous avons vu des méthodes utilisant un formalisme de plus en plus *abstrait*. Depuis ma thèse, et avec les chercheurs de l'équipe-projet ALICE, j'ai eu pour objectif de gravir ces différents niveaux d'abstractions, avec l'intuition qu'aborder les problèmes de modélisation au bon niveau d'abstraction était susceptible d'aboutir à des résultats de recherche intéressants. Comme fil conducteur, nous avons manipulé toujours le même type d'objet géométrique, à savoir des maillages avec des valeurs associées aux sommets, représentant soit la géométrie, soit les propriétés attachées à cette géométrie. Pour manipuler ces objets, nous avons ainsi étudié les formalismes suivants:

- ◇ *Optimisation numérique:* Pour manipuler ces valeurs associées aux sommets de maillages, il parait naturel de les regrouper en vecteurs, et étudier les fonctions de plusieurs variables prenant ces vecteurs en paramètres. Ainsi les outils d'optimisation numérique (Section 2.2) nous ont permis de mettre au point des algorithmes d'optimisation géométrique (Section 3.5).
- ◇ *Paramétrisation des surfaces:* Cette notion qui permet de construire une "carte" de l'objet, nous a permis d'étudier différents algorithmes, en particulier dans le domaine du plaquage de textures et de la conversion entre représentations (Section 4.4).
- ◇ *Eléments finis et bases de fonctions:* Les méthodes par éléments finis sont un formalisme très efficace pour la simulation de phénomènes physiques (Section 3.6), et nous commençons à l'appliquer également à des problèmes de modélisation géométrique (Section 4.5).

◇ *Bases de fonctions mobiles:* Finalement, l'orientation actuelle de nos recherches nous pousse à mettre au point des algorithmes qui construisent une base de fonction, bien adaptée au problème étudié.

Ainsi, nous avons pris comme point de départ l'étude de la représentation des maillages, et des outils d'optimisation numérique permettant de manipuler leur géométrie et leurs propriétés.

Tout d'abord, la notion de *paramétrisation* nous a permis de franchir un niveau d'abstraction, en considérant une forme géométrique par l'intermédiaire d'un système de coordonnées curviligne défini sur cette forme. Ceci permet par exemple de découpler les propriétés colorimétriques du support géométrique (plaquage de textures), ou encore de simplifier certains calculs liés à l'optimisation géométrique [Lé03].

Ensuite, le formalisme des *éléments finis* nous a permis de mieux caractériser les différentes fonctions que nous manipulons, et de développer de nouvelles méthodes de simulation [LLAP05]. L'*analyse fonctionelle*, à savoir l'un des fondements de la théorie des éléments finis, nous laisse entrevoir le niveau d'abstraction supérieur que nous allons aborder dans la suite de nos recherches.

5.3.2 Perspectives

L'analyse fonctionelle étudie la structure (par exemple la topologie) des espace de fonctions, et les propriétés des application définies sur ces espace, appelées des *opérateurs* (autrement dit, des fonctions qui prennent des fonctions en paramètres et qui renvoient d'autres fonctions). Ceci nous permet de manipuler les objets géométriques d'une manière plus abstraite, et ainsi de mettre au point des outils plus puissants. Les premiers résultats de cette approche sont l'algorithme de vectorisation ARDECO [LL06], permettant de vectoriser une image bitmap. Nous projetons d'expérimenter ce formalisme appliqué à la compression de flux vidéo. D'autre part, dans certaines configurations particulières, plus de connaissances *a priori* sur la structure du problème permettent de spécifier de manière plus précise le critère d'optimalité de la base de fonction. Par exemple, j'étudie actuellement particulièrement les fonctions propres d'opérateurs différentiels linéaires symétriques, qui naturellement définissent des bases de fonctions orthogonales. Il est ainsi possible de généraliser l'analyse de Fourier à des formes arbitraires, et leur appliquer tous les outils de filtrage disponibles dans ce formalisme [Lé06]. La suite de cette section donne plus de détails sur cette approche, que j'ai présentée dans cet article (invité à la conférence IEEE Shape Modeling International), et qui va structurer un ensemble d'activités de recherche dans mon équipe.

La Figure 5.6 montre l'allure de la première fonction propre (également appelée *vecteur de Fiedler* en théorie des graphes). Nous pouvons remarquer que cette fonction propre définit une paramétrisation naturelle de l'objet, qui suit sa forme principale. D'une certaine manière, ceci définit une "analyse en composantes principales curviligne", qui parvient à capturer la forme générale de l'objet. Nous montrons également sur la Figure 5.7 l'aspect des autres fonctions propres. Ces fonctions définissent sur l'objet l'équivalent de la base de Fourier (d'une certaines manières, ces fonctions correspondent à une version bidimentionelle de la fonction sinus définie sur l'objet). Enfin, la Figure 5.8 montre comment le signal géométrique (à savoir la forme d'un objet) peut être décomposé sur cette base de fonctions. Les premiers éléments de la base capturent la forme générale de l'objet, et les éléments suivants des détails de plus en plus fin. Ces premiers résultats, appliqués à de petits objets (quelques milliers de facettes), nous semblent prometteurs.

Nous souhaitons par la suite étendre ces résultats, à des objets de tailles plus conséquentes (millions de facettes), et généraliser la méthode à des éléments finis de degré supérieur. Ceci va nous permettre d'adapter à des surfaces de forme et de topologie arbitraire les outils de filtrage destinés initialement à des domaines plus simple (transformée

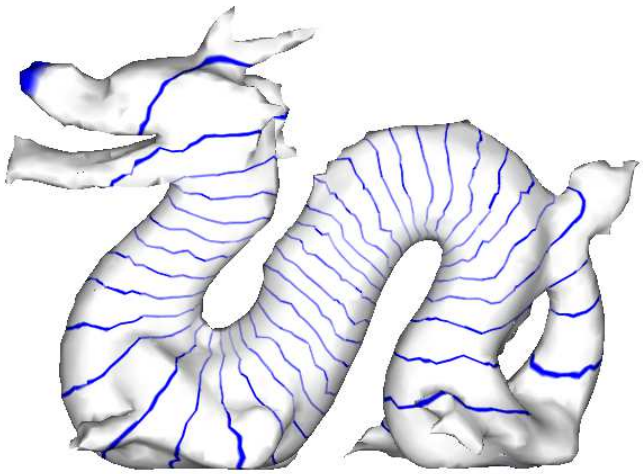


Figure 5.6: *Le vecteur de Fiedler fournit un ordre naturel pour ordonner les sommets d'un maillage. Les lignes bleues correspondent aux iso-valeurs de la première fonction propre. Nous montrons ici comment l'ordre induit par le tri des sommets par valeurs croissante suit naturellement la forme de l'objet, même si elle est curviligne.*

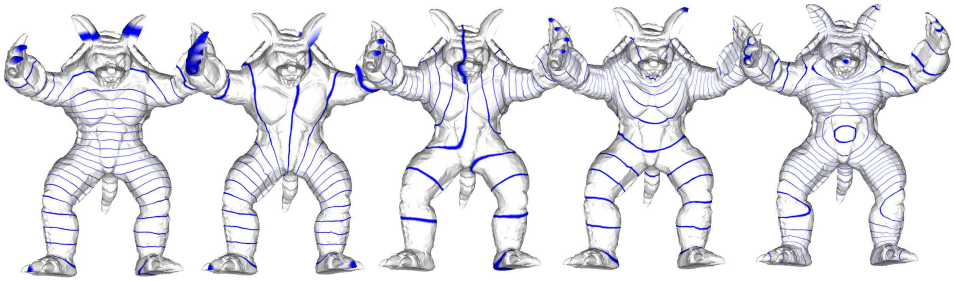


Figure 5.7: *Lignes de niveaux des premières fonctions propres de l'opérateur Laplace-Beltrami. Nous pouvons remarquer que les protrusions et les symétries de l'objet sont capturées. Les fonctions propres capturent la manière subtile dont la topologie et la géométrie de l'objet se combinent.*

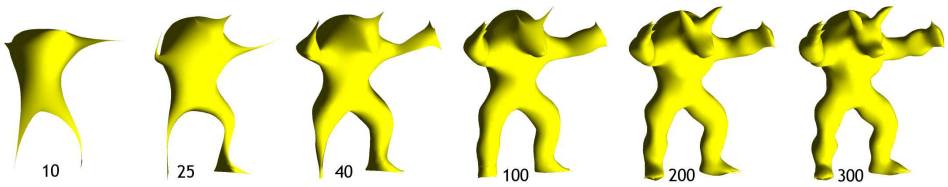


Figure 5.8: *Reconstructions de la géométrie obtenues en utilisant un nombre croissant de fonctions propres.*

de Fourier). Nous souhaitons par la suite étendre ce formalisme à des bases de fonction irrégulières, supportant le raffinement adaptatif (base d'ondelettes).

Pour développer ce nouveau formalisme, nous allons devoir acquérir de nouvelles connaissances, concernant à la fois des branches modernes des mathématiques, et des aspects liés à la technologie graphique:

- ◇ **Géométrie différentielle et topologie:** cette branche récente des mathématiques offre un formalisme très général et très puissant. Nous avons débuté l'étude de cette thérorie (et plus particulièrement du calcul extérieur et de la co-homologie), qui nous a permis de mettre au point de nouvelles méthodes de visualisation de champs de vecteurs [LVRL06]. Nous explorons actuellement des rafinements plus poussés de ces théories, afin de généraliser la notion de bases d'ondelettes à des objets de genre arbitraire (en utilisant en particulier la notion de matrice de période, liée aux bases de co-homologie).
- ◇ **Eléments finis d'ordre supérieur** le domaine scientifique du traitement numérique de la géométrie auquel nos recherches s'apparente utilise essentiellement des fonctions linéaires par morceaux pour attacher des attributs à des surfaces triangulées (les morceaux correspondant aux triangles). Afin de nous affranchir des artefacts liés à la discrétisation du maillage, nous développons actuellement une théorie de la géométrie numérique fondée sur le formalisme des éléments finis. Nous avons débuté les expérimentations avec des fonctions d'ordre plus élevé. Ces premières expérimentations sont susceptibles de fournir des résultats importants, qui servirons comme base théorique pour plusieurs développements futurs de l'équipe (bases de fonctions quasi-harmoniques, plaquage de textures d'ordre supérieur, simulation dynamique sur des objets complexes).
- ◇ **Programmation des cartes graphiques (GPU):** Les cartes graphiques continuent leur évolution très rapide. Afin de rester à la pointe de la technique, nous nous documentons sur ces aspects. J'ai développé un moteur de rendu exploitant ces nouvelles technologies (shaders, rendu 'High Dynamic Range', ...). D'autre part, nous expérimentons également les nouvelles API, qui permettent d'utiliser la carte graphique en tant que processeur de calcul.
- ◇ **Solveurs numériques:** A la base des implantations pratiques de nos algorithmes, les solveurs numériques jouent un rôle très important pour l'équipe. Une bonne connaissance des solveurs linéaires nous a assuré une bonne avance dans les années 90-2000. Au début des années 2000, nous avons attaqué le monde non-linéaire, ce qui nous a permis de développer de nouveaux algorithmes (dont la méthode ABF++ [SLMB04], en coopération avec University of British Columbia). Nous nous intéressons à présents aux solveurs pour problèmes aux valeurs propres, à la base de l'analyse harmonique, l'un des aspects de la géométrie différentielle que nous étudions. La loi de Moore ayant atteint ses limites, elle ne peut être actuellement dépassée que par l'utilisation de plusieurs unités de calcul en parallèle. Nous explorons en coopération avec le projet GRAAL des solveurs parallèles efficaces (MUMPS) pour résoudre ces problèmes numériques sur de très grands maillages, avec des éléments finis d'ordre élevé.

La combinaison de ces connaissance nous donnera la maitrise des outils mathématiques fondamentaux, et ainsi de formaliser *quelle équation* nous devons résoudre. La maitrise des techniques de programmation et du hardware nous permettra de savoir *comment résoudre* ces équations. Nous pensons que la combinaison de ces deux types de connaissances est susceptible de permettre d'importantes avancées dans le domaine de la géométrie numérique.

Publications

1. JOURNAUX INTERNATIONAUX AVEC COMITE DE LECTURE

[RLL+06] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Transaction on Graphics*, 2006.

[LVRL06] Wan Chiu Li, Bruno Vallet, Nicolas Ray, and Bruno Lévy. Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization '06), 2006-10-31*, 2006.

[CMCL06] Laurent Castanié, Christophe Mion, Xavier Cavin, and Bruno Lévy. Distributed shared memory for roaming large volumes. *IEEE Transactions on Visualization and Computer Graphics (proceedings Visualization 2006)*, 12(5), 2006.

[CLB05b] Laurent Castanié, Bruno Lévy, and Fabien Bosquet. Advances in seismic interpretation using new volume visualization techniques. *First Break Journal*, October 2005.

[SLMB04] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. ABF++ : Fast and robust angle based flattening. *ACM Transactions on Graphics*, Apr 2004.

[CLCP04] Guillaume Caumon, Bruno Lévy, Laurent Castanie, and Jean-Claude Paul. Advanced visualization for various unstructured grids using ah hoc topological structures. *Computers and Geosciences*, Sep 2004.

[Lé03] Bruno Lévy. Dual domain extrapolation. *ACM Transactions on Graphics (SIGGRAPH)*, 22(3):364–369, Jul 2003. SIGGRAPH 2003 Session : Parameterization.

[ACSD+03] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics (SIGGRAPH)*, 22(3):485–493, Jul 2003. SIGGRAPH 2003 Session : Surfaces.

[RSLC02] Nicolas Rémy, Arben Shtuka, Bruno Lévy, and Jeff Caers. Gstl : a geostatistical template library in c++. *Computers and Geosciences*, 28(8):971–979, October 2002.

2. JOURNAUX NATIONAUX AVEC COMITE DE LECTURE

[DLM00] Matthieu Dazy, Bruno Lévy, and Jean-Laurent Mallet. Ensembles simpliciaux hiérarchisés et intersections de surfaces. *Revue internationale de CFAO et d'Informatique graphique*, 15(1):11–23, June 2000.

[LJL00] Bruno Lévy and Mallet Jean-Laurent. Paramétrisation des surfaces triangulées. *Revue Internationale de CFAO et d'Informatique Graphique*, 15(1):25–42, June 2000.

3. CONFERENCES INTERNATIONALES AVEC ACTES ET COMITE DE LECTURE

[BPK+07] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff, C. Röossl, "Geometric Modeling Based on Polygonal Meshes - SIGGRAPH 2007 Course Notes", 2007.

[HLS+07] K. Hormann, B. Lévy, A. Sheffer, "Mesh Parameterization: Theory and Practice - SIGGRAPH 2007 Course Notes", 2007.

[TLP07] R. Toledo, B. Lévy, J.-C. Paul, "Iterative Methods for Visualization of Implicit Surfaces on GPU", *in : ISVC, International Symposium on Visual Computing*, Lake Tahoe United States, 2007.

[TWL07] R. Toledo, B. Wang, B. Lévy, "Geometry Textures", *in : SIBGRAPI*, Belo Horizonte Brazil, 2007.

[ZLS07] R. Zayer, B. Lévy, H.-P. Seidel, "Linear Angle Based Parameterization", *in : ACM/EG Symposium on Geometry Processing*, Barcelone Spain, 2007.

[BCL07] L. Buatois, G. Caumon, B. Lévy, "Concurrent Number Cruncher : An Efficient Sparse Linear Solver on the GPU", *in : High Performance Computation Conference*, Houston United States, 2007.

[LRL06] Wan Chiu Li, Nicolas Ray, and Bruno Lévy. Automatic and interactive mesh to t-spline conversion. In *EG/ACM Symposium on Geometry Processing, 2006-06-26*, Italy, 2006.

[LL06] Gregory Lecot and Bruno Lévy. Ardeco: Automatic region detection and conversion. In *Eurographics Symposium on Rendering, 2006-07-30*, Chypre, 2006.

[Lé06] Bruno Lévy. Laplace-beltrami eigenfunctions: Towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications, 2006-06-14*, invited talk, Japan, 2006.

[BCL06] Luc Buatois, Guillaume Caumon, and Bruno Lévy. Gpu accelerated isosurface extraction on tetrahedral grids. In *International Symposium on Visual Computing, 2006-11-06*, USA, 2006.

[CLB05a] Laurent Castanie, Bruno Lévy, and Fabien Bosquet. VolumeExplorer: Roaming large volumes to couple visualization and data processing for oil and gas exploration. In *Visualization conf. proc., to appear*. IEEE, 2005.

[LLAP05] Gregory Lecot, Bruno Lévy, Laurent Alonso, and Jean-Claude Paul. Master-element vector irradiance for large tessellated models. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, 2005.

[LLP05] Wan Chiu Li, Bruno Lévy, and Jean-Claude Paul. Mesh editing with an embedded network of curves. In *Shape Modeling International conference proceedings*, 2005.

[Lé05] Bruno Lévy. Numerical methods for digital geometry processing. In *Israel Korea Bi-National Conference - Invited talk*, November 2005.

[RL03] Nicolas Ray and Bruno Lévy. Hierarchical least squares conformal maps. In *11th Pacific Conference on Computer Graphics and Applications 2003 - PG'03, Canmore, Canada*, pages 263–270, Octobre 2003.

[RUCL03] Nicolas Ray, Jean-Christophe Ulysse, Xavier Cavin, and Bruno Lévy. Generation of radiosity texture atlas for realistic real-time rendering. In *Eurographics 2003, Granada, Espagne*, September 2003.

[DMLC02] Jean-Michel Dichler, Karl Maritaud, Bruno Lévy, and Djamchid Chazanfarpour. Texture particles. In *Eurographics 2002 - EG 2002, Saarbrücken, Germany*, Sep 2002.

[LP02] Bruno Lévy and Sylvain Petitjean. Least squares conformal maps. In *Fifth International Conference on Curves and Surfaces 2002 , Saint-Malo, France*, Jun 2002.

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In ACM, editor, *SIGGRAPH*, Jul 2002.

[Lé01] Bruno Lévy. Constrained texture mapping. In *SIGGRAPH*. ACM, Addison Wesley, Aug 2001.

[LCCC01] Bruno Lévy, Guillaume Caumon, Stephane Conreaux, and Xavier Cavin. Circular incident edge lists : A data structure for rendering complex unstructured grids. In *IEEE Visualization 2001, San-Diego, USA*. IEEE, Oct 2001.

[LM98] Bruno Lévy and Jean-Laurent Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *SIGGRAPH*. ACM, 1998.

4. **ARTICLES SOUMIS**

[RVL$^+$] Nicolas Ray, Bruno Vallet, Wan Chiu Li and Bruno Lévy. N-Symmetry Direction Fields on Surfaces of Arbitrary Genus Accepted pending minor revisions, *ACM Transaction on Graphics*.

[VL] Bruno Vallet and Bruno Lévy. Geometry Filtering with Manifold Harmonics, Accepted pending minor revisions, *EUROGRAPHICS, Computer Graphics Forum*

[RLW+] Nicolas Ray, Bruno Lévy, Huamin Wang and Greg Turk, Material-Space Texturing Accepted pending revisions, *Computer Graphics Forum*.