Conversion automatique de maillages en surfaces splines

Wan Chiu Li

Département de formation doctorale en informatique

Institut National Polytechnique de Lorraine

École doctorale IAEM Lorraine

Conversion automatique de maillages en surfaces splines

THÈSE

présentée et soutenue publiquement le jeudi 16 novembre 2006

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(spécialité informatique)

par

Wan-Chiu Li

Composition du jury

Président :	Pr. Jean–Laurent Mallet	INPL
Rapporteurs :	Pr. Dominique Bechmann	ULP Strasbourg I
	Pr. Leif Kobbelt	Lehrstuhl Informatik VIII
Examinateurs :	Dr. Alexander Belyaev	MPII
	Dr. Bruno Lévy	INRIA
Directeur de thèse :	Pr. Jean–Claude Paul	CNRS

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

Je tiens tout d'abord à remercier Jean–Claude Paul pour tous ses efforts énormes de rendre cette thèse possible administrativement, pour être mon directeur de thèse, et pour avoir toujours soutenu mon travail.

Je remercie également les membres du jury, Jean–Laurent Mallet, Dominque Bechmann, Leif Kobbelt, et Alexander Belyaev, qui m'ont fait l'honneur d'être les rapporteurs et examinateurs de cette thèse.

Je remercie beaucoup Bruno Lévy pour la confiance qu'il m'a accordée, pour les choses qu'il m'a apprises, et pour les opportunités qu'il m'a données tout au long de ces cinq ans. D'ailleurs, je lui suis reconnaissant pour m'avoir beaucoup inspiré avec sa passion pour l'infographisme, et avec sa générosité de partager ses savoir-faires.

Dans le même ordre d'idée, je voudrais aussi remercier beaucoup Nicolas Ray pour son support académique, technique, mental et fraternel, et pour m'avoir permis de participer à ses innovant projets de recherche qui sont des excellentes démonstrations de sa créativité.

Je tiens à remercier les collègues qui m'ont aidé sur la rédaction en français. Puis, un merci spécial à Denis Roegel pour sa classe thloria pour LaTeX, qui a facilité la vie de nombreux thésards du laboratoire.

Je remercier beaucoup également les membres de l'ancien ISA et d'ALICE qui ont toléré gentiment mon français primitif !

To my dear parents,
sister and
Regina

Résumé

Afin de convertir un maillage triangulaire en une surface spline de CAGD/CAM, cette thèse adresse l'un des problèmes les plus cruciaux du processus de conversion : extraire un "bon" maillage de contrôle quadrilatéral de la surface. Ce que nous entendons par "bon" est que les arêtes du maillage de contrôle se croisent perpendiculairement et sont alignées avec les principales directions de la courbure de la surface. Ces deux propriétés du maillage de contrôle permettent de fournir une bonne approximation de la surface avec peu de points de contrôles. D'ailleurs, ils aident considérablement à réduire des oscillations non-désirées sur la surface spline finale.

Pour résoudre ce problème, nous proposons un nouvel algorithme automatique, appelé *paramétrisation globale périodique*. L'idée fondamentale de cet algorithme est de trouver une paramétrisation qui ait un "sens d'un point de vue géométrique", pour ce faire, elle doit être guidée par la courbure de la surface, représentée par une paire de champs de direction orthogonaux. Les iso-lignes de cette paramétrisation sont ensuite extraites pour définir un maillage de contrôle qui ait les propriétés requises.

Ce maillage de contrôle, nous permet de construire une approximation en surface T-spline de la surface triangulée initiale. Nous exposons plusieurs résultats de cette conversion d'un maillage triangulée en surface spline. Les résultats montrent que, grâce aux maillages de contrôle anisotropes, les surfaces spline finales ont beaucoup moins d'oscillations que celles construites par les méthodes précédentes qui ne tiennent pas compte de l'anisotropie de la surface.

Mots-clés: infographie, traitement géométrie, surface paramétrisation, surface spline, visualisation, surface topologie

Abstract

Aiming at converting a triangular mesh into a CAGD/CAM spline surface, this thesis focuses on one of the most crucial problems of the conversion process, i.e. extracting a "good" quadrilateral control mesh of the surface. What we mean by good is that the edges of the control mesh should be orthogonal and aligned with the principal directions of curvature of the surface. These two properties make the control mesh optimum in an approxmiation point of view, and greatly help to reduce unwanted oscillations on the final spline surface built from it.

To solve this problem, we propose a new automatic algorithm, called *periodic global parameterization*. The basic idea is to find a "geometry-meaningful" parameterization guided by a pair of orthogonal anisotropic direction fields. Then, the iso-value lines of this parameterization will be extracted to define an initial control mesh, that satisfies the two criteria of a good control mesh.

With the initial control mesh, we explain how to construct a T-spline approximation of the initial triangulated surface. We show several examples of the triangular mesh to T-spline conversion. The results show that thanks to the anisotropic control meshes, the final spline surfaces generated have much less oscillations as compared to results of previous methods, that do not take into account of the anisotropy.

Keywords: computer graphics, geometry processing, surface parameterization, spline surface, visualization, surface topology

Table des matières

Annexe H

Introduction

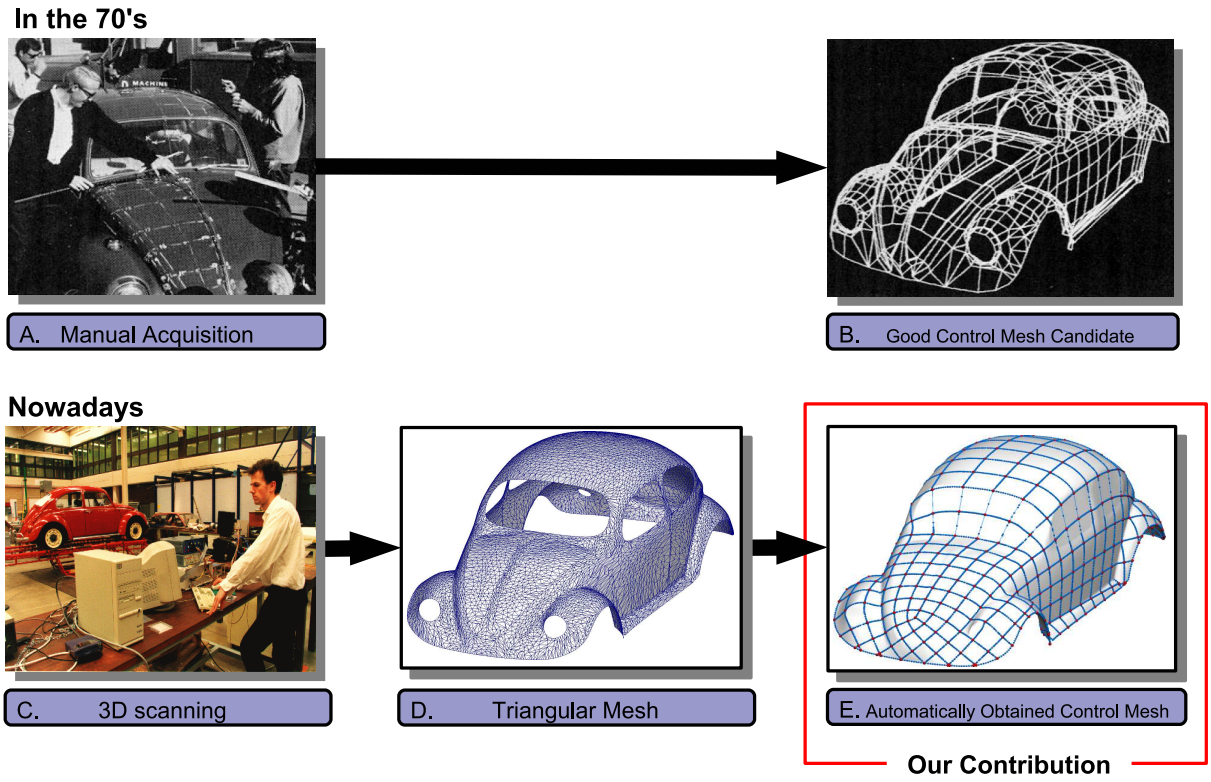


FIG. 1: *Convertir un objet réel en un modèle utilisable dans le domaine du CAGD/CAM est une tâche qui requiert beaucoup d'interactions de la part d'un utilisateur (A,B). De nos jours, la technologie des scanners 3D permet d'automatiser le processus (C). Malheureusement, les données brutes qui peuvent être extraites de la sorte ne sont pas directement utilisables dans le contexte de la CAGD/CAM (D). Cette thèse présente de nouveaux algorithmes capables de convertir les données brutes extraites par scanner en des objets splines utilisés en CAGD/CAM (E).*

1 Définition du problème

Les représentations paramétriques des surfaces et plus précisément les surfaces splines (*tensor-product splines*), [Far02] sont les plus utilisées dans les domaines de la CAGD (*Computer-Aided Geometric Design*), du CAM (*Computer-Aided Manufacturing*) et du graphisme par ordinateur. Dans ces représentations, la géométrie du maillage est définie par un maillage de contrôle quadrilatéral. De telles représentation permettent de modifier la forme de la surface par un simple déplacement des points de contrôle du maillage, tout en préservant la continuité (généralement C^2) de la surface. De plus, dans le contexte des méthodes par éléments finis et de la modélisation automobile la forme analytique de ces surfaces permet de calculer précisément leur normale et leurs dérivées en n'importe quel point. Par exemple, dans le contexte du design automobile, les effets de lumières sur la carrosserie sont directement liés à la normale de la surface, et requièrent la manipulation de surfaces au moins 2 fois dérivables (C^2). Ainsi, la surface de la Class-A [ICE] doit être au moins 2 fois dérivable pour ne pas créer d'artefacts visuels lors de son affichage.

Dans les années 1970, la représentation digitale des objets ne pouvait généralement être acquise que par des procédés manuels (Figure 1-A). Les maillages étaient alors bien adaptés aux caractéristiques des objets afin de limiter le nombre de polygones et de pouvoir ainsi être manipulé par les ordinateurs de cette époque. Ces maillages optimisés pour suivre les lignes caractéristiques des objets (Figure 1-B) étaient très bien adaptés pour construire des maillages de contrôles.

De nos jours, grâce aux progrès réalisés dans les technologies de scanners (Figure 1-C), il est désormais possible d'obtenir automatiquement une surface triangulée à partir d'un objet réel (Figure 1-D). Cependant, ces maillages ne peuvent généralement pas être directement utilisés comme maillage de contrôle de surface paramétriques pour les raisons suivantes :

- Les cellules sont des triangles plutôt que des quadrilatères, ce qui ne permet pas de définir les splines les plus usitées dans les applications CAGD/CAM.
- Les maillages créés par scanner sont très denses puisque seul les sommets permettent de définir la géométrie de la surface, or il est préférable d'avoir des maillages moins denses mais dont les cellules soient placées de manière intelligente. En effet, la géométrie des surfaces splines peut être capturée grâce aux degrés des équation qui les définissent.

L'objectif de ce travail est de mettre au point un algorithme capable de convertir des données brutes (issues d'un scanner) en un bon maillage de contrôle sur lesquels des splines pourront être définies (Figure 1-E) (une définition plus rigoureuse de la qualité sera donnée dans la section suivante et développée dans l'annexe F). Ces maillages de contrôles pourront ensuite être

exploités pour créer des surfaces splines, approximant au mieux la surface initiale, directement utilisable dans les application de type CAGD/CAM.

La prochaine section présente les principales contributions de cette thèse.

2 Contributions

Étant donné :

– un maillage triangulé de topologie arbitraire,

– et un seuil d'erreur d'approximation entre le maillage triangulé initial et la surface spline,

notre objectif est de construire une surface spline (Figure 2-F) à partir du maillage triangulé (Figure 2-A) en respectant l'erreur autorisée donnée par l'utilisateur. L'algorithme suit les étapes indiquées dans la figure 2.

A : Entrée de notre algorithme : un maillage triangulaire scanné

Pour obtenir une représentation tridimensionnelle d'un objet réel, il est possible d'utiliser un scanner laser (voir par exemple [LPC+00, Cyb, Rap]). Un scanner 3d crée un nuage de points qui est ensuite triangulé par un logiciel de reconstruction *ad hoc* (par exemple [KBH06, HK06]). Ce maillage triangulé très dense constitue la donnée d'entée de notre méthode.

B : Génération d'un champ de direction de guidage

Les surfaces splines nécessitent un maillage de contrôle quadrilatéral des objets. Comme expliqué dans [d'A00] par d'Azevedo, les arêtes d'un maillage quadrilatéral optimal doivent se croiser de forme orthogonales et être alignées avec les directions de courbure principales de la surface (adaptées à l'anisotropie[1]). Par exemple, il serait plus naturel de dessiner les arêtes du maillage de contrôle d'un cylindre parallèles et perpendiculaires à l'axe du cylindre (Figure 3-A). De plus, l'algorithme doit éviter la configuration montré dans la Figure 3-B, ce qui causerait des oscillations non désirées dans la surface spline finale. Notons que les travaux précédents (par exemple [EH96]) ne prennent pas ce problème en considération.

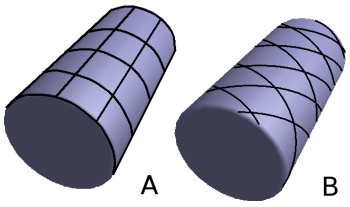


FIG. 3: *Maillage de contrôle adapté ou non à l'anisotropie de la surface.*

Afin de corriger ce problème, notre idée est de définir un *champ de direction de guidage* qui va orienter les des arêtes du maillage de contrôle pour l'adapter à l'anisotropie de la surface.

[1] Pour améliorer la lisibilité de l'exposé, le terme "anisotropie de la surface" sera employé à la place de "anisotropie du tenseur de courbure de la surface".

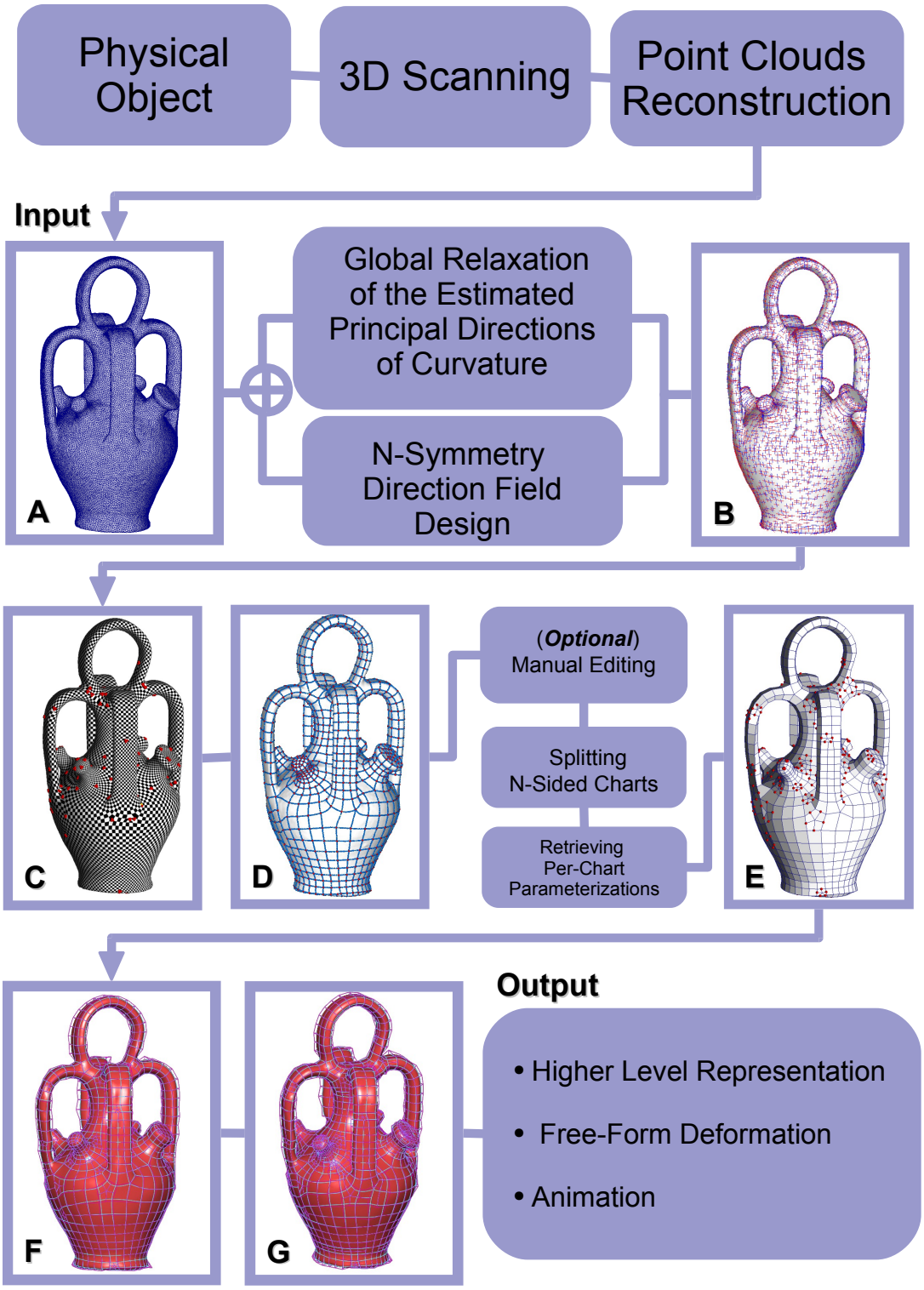


FIG. 2: *Étapes successives de notre processus de conversion. A : maillage triangulé d'entrée ; B : champs de direction de guidage ; C : Paramétrisation globale ; D : extraction de l'agencement des cellules en utilisant une structure de donnée en plongement du complexe cellulaire ; E : agencement des cellules compatible avec notre représentation sous forme de splines ; F : surface spline construite en utilisant comme maillage de contrôle notre agencement de cellule extrait à la dernière étape ; G : surface spline ajustée à la surface initiale.*

Nous avons proposé deux nouveaux algorithmes pour produire de tels champs de direction de guidage :

- *Relaxation globale d'une estimation des directions principales de courbure :*
  trouver deux champs de direction qui correspondent aux direction principales de courbure de la surface en estimant le tenseur de courbure, puis, lisser les champs de direction par une approche de relaxation globale. Cette méthode donne des champs de direction de guidage lisses avec un ensemble de singularités significatives générées automatiquement. (Pour plus de détails, voir l'annexe D.4, ainsi que notre article publié à *ACM Transactions on Graphics* [RLL$^+$].)
- *Contrôle des singularités :*
  Les singularités d'un champ de direction (ou pôles) et leurs indices caractérisent la topologie du champ. Notre idée est de construire un champ de direction en spécifiant sa topologie à travers le placement et les indices de telles singularités. De plus, nous contrôlons la géométrie du champ à travers des contraintes directionnelles pour rendre ce champ adapté à l'anisotropie. (Cette partie de la thèse a été réalisée en collaboration avec le doctorant Bruno Vallet. Pour plus de détails, se référer à l'annexe E, à notre revue technique [RVLL06] ainsi qu'à l'article publié dans *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization '06)* [LV$^+$06].)

C : Paramétrisation périodique globale

Existe une différence fondamentale entre un maillage et notre représentation cible par des splines. Un maillage est un échantillonnage de la géométrie, alors qu'une surface spline nécessite plus de structure, plus précisément, la construction d'une spline nécessite une *paramétrisation* de l'objet.

Dans cette étape, nous comblerons l'écart entre ces deux représentations en construisant une représentation abstraite de l'objet, en tant que surface paramétrée. Il existe de nombreuses méthodes pour paramétrer une surface [FH05], mais, afin de produire une paramétrisation cohérente avec la géométrie et son anisotropie, nous avons proposé un nouvel algorithme dans lequel la paramétrisation est guidée par une paire de champs de direction orthogonaux et adaptés à l'anisotropie (le résultat de l'étape B) ce qui produit une paramétrisation globale du maillage triangulé initial.

Étant donné que la paramétrisation est adaptée à la géométrie, les lignes d'iso-valeur définissent un *maillage contrôle quadrilatéral naturel* de la surface. Notez que cet algorithme est indépendant de la topologie (du genre) de l'objet. (Pour plus de détails, se référer à l'annexe

F.2, et à notre article publié dans *ACM Transactions on Graphics* [RLL$^+$].)

D : Extraction de l'agencement des cellules
Une fois obtenue la représentation abstraite paramétrée (résultat de l'étape C), nous en extrayons les lignes d'iso-valeur. Ceci définit un maillage de contrôle initial. Pour ce faire, nous avons proposé une structure de donnée en plongement de complexe cellulaire pour représenter un maillage de contrôle et ses intersections avec le maillage triangulé initial. De plus, nous avons proposé des outils interactifs pour améliorer manuellement le résultat automatique si besoin est. (pour plus de détails, consulter les annexes F.3.3 et C, ainsi que notre article publié dans *Proceedings of IEEE International Conference on Shape Modeling and Applications '05* [LLP05].)

E, F, G : Ajustement
Une fois la représentation abstraite de la surface construite et le maillage de contrôle extrait, nous appliquons les contraintes de validité du maillage de contrôle requises par la représentation spline voulue (E), et ajustons la surface spline à la surface triangulée initiale (F). Un raffinement adaptatif local est ensuite appliqué (G). (Pour plus de détails, voir les annexes F.3.3, F.3.4 et G, ainsi que notre article publié dans *Proceedings of EG/ACM Symposium on Geometrie Processing '06* [LRL06].)

Le reste de cette thèse est organisée comme suit : en chapitres 1–5, nous détaillons les étapes de l'algorithme ; en chapitre 6 nous concluons cette thèse et nous discutons plusieurs travaux futurs.

Chapitre 1

Étape A : Donnée d'entrée et objectif de l'algorithme

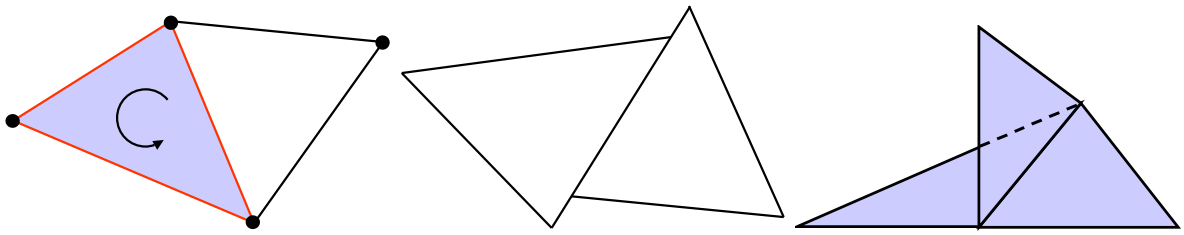


FIG. 1.1: *Gauche : les points sont les 0-simplex, les lignes sont les 1-simplex, et les triangles sont les 2-simplex. La flèche à l'intérieur du triangle bleu indique l'orientation des simplex. Les trois lignes rouges indiquent le bord du triangle bleu ; Milieu : pas un complexe simplicial ; Droite : pas une 2-variété.*

1.1 Maillage triangulaire

La donnée d'entrée de l'algorithme est un maillage triangulaire issue d'un objet scanné, qui doit être converti dans notre représentation par des splines.

Les maillages sont la représentation classique des surfaces dans R^3 dans le graphisme par ordinateur. Quand toutes facettes du maillage sont les triangles, c'est à dire un maillage triangulaire, nous avons un 2D *complexe simplicial* K. Dans un 2D complexe simplicial, les éléments sont un point (0-simplex), une ligne segment (1-simplex), ou un triangle (2-simplex). Nous supposons toujours que des orientations locales sont données. C'est à dire, chaque élément du maillage a été donné une orientation particulière. L'opérateur de bord ∂ d'un k-simplex est la

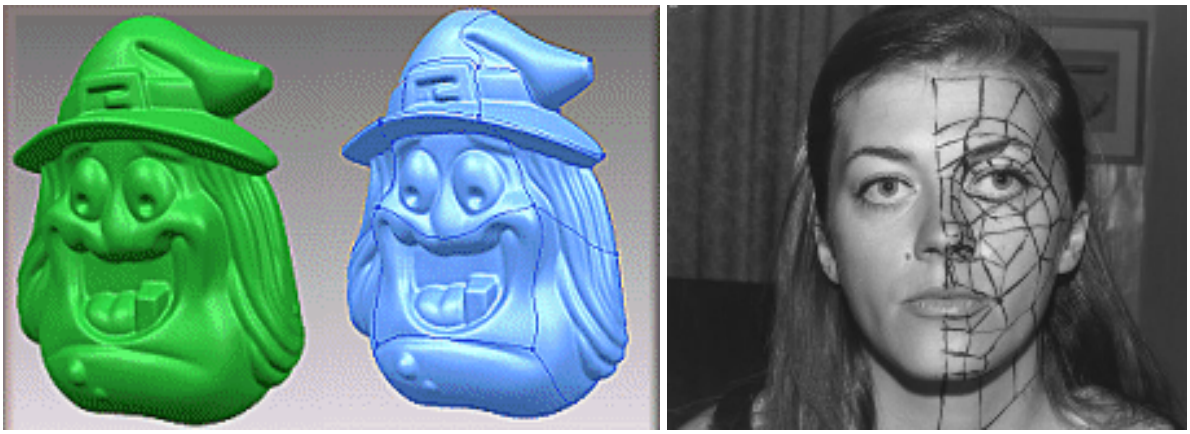


FIG. 1.2: *Gauche : construction manuelle d'un maillage de contrôle CAGD/CAM [Rap] à partir d'un maillage triangulaire ; Droite : numérisation d'un objet réel par acquisition manuelle (l'image est tiré de la thèse de Henri Gouraud [Gou71b]).*

chaîne de toutes ses $(k-1)$-faces. Un complexe simplicial doit satisfaire les deux règles suivantes (Figure 1.1) :

1. chaque face de chaque simplex dans K est aussi dans K ;
2. l'intersection de deux simplex quelconques dans K est vide, ou une face commune entière.

Noter qu'à cette thèse, nous sommes seulement intéressés par les complexes simplicial avec une réalisation topologique de 2-variété (Figure1.1-Droite). Nous employons la structure de demi-arrête [Wei85] pour représenter les complexes simplicials dans notre implantation.

1.2 Objectif de l'algorithme

Pour convertir un maillage triangulaire en une surface spline, nous devons extraire un maillage de contrôle CAGD/CAM quadrilatéral, c'est à dire adapté à l'anisotropie de la surface triangulée.

De nos jours, la façon la plus simple de créer un tel maillage de contrôle est de laisser l'utilisateur le construire manuellement, ce qui est souvent le cas dans les logiciels industriels [Cyb, Rap]. Grâce à la flexibilité du dessin manuel, et à sa capacité à adapter spontanément à l'anisotropie, il est maintenant systématiquement utilisé dans l'industrie. Cette méthode de construction manuelle des maillages de contrôle, implémentée dans un modeleur 3D, est effectivement assez similaire au processus de digitalisation par acquisition manuelle des années 70 (Figure 1.2-Droite). En pratique, l'utilisateur dessine le maillage de contrôle sur une repré-

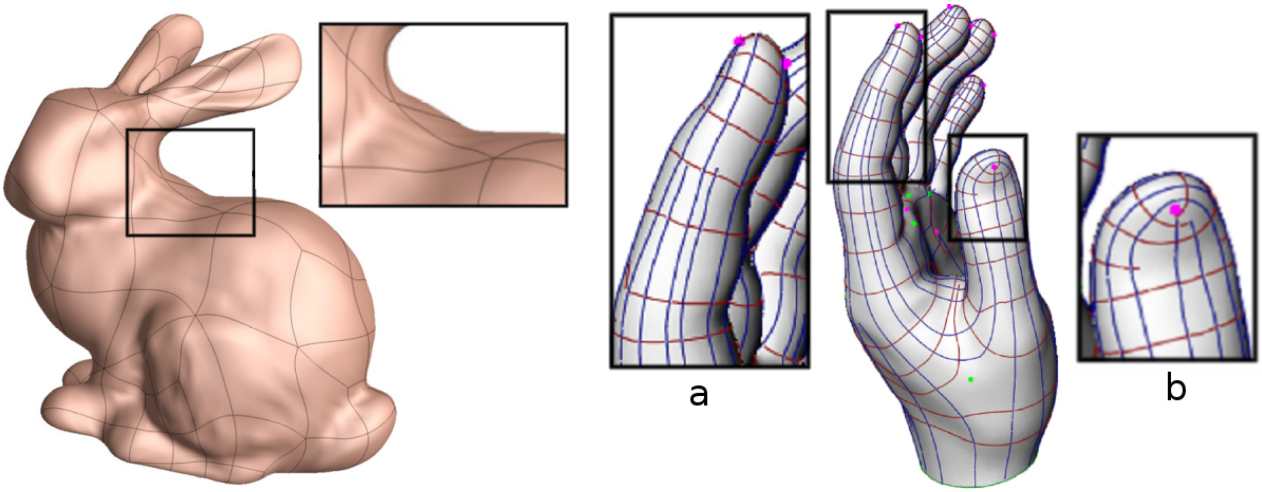


FIG. 1.3: *Gauche : un résultat de la méthode de Eck and Hoppe, [EH96] des ondulations apparaissent sur la surface splineà cause du fait que le maillage de contrôle ne soit pas adapté à l'anisotropie ; Droite :un résultat de la méthode de Alliez* et al. *[ACSD+03],(a) espacement irrégulier arêtes ; (b) boucles ouvertes.*

sentation virtuelle de l'objet (Figure 1.2-Gauche). Dans les deux cas, on dessine les arêtes du maillage à la main est on essaye de les adapter aux caractéristiques de l'objet.

Bien que plus flexible, le dessin manuel est très coûteux en temps et fastidieux pour des surfaces de géométrie et/ou de topologie complexe. C'est pour cette raison que beaucoup de méthodes automatiques ont été proposées, comme les premières approches de Eck and Hoppe [EH96]. Malheureusement, on observe assez souvent des oscillations parasites sur les surfaces construites à partir de maillages de contrôle obtenus par leur méthode automatique (Figure 1.3-Left). Cela s'explique par le fait que les maillages de contrôle ne sont pas adaptés à l'anisotropie. Récemment, Alliez *et al.* [ACSD+03] ont proposé une méthode adaptée à l'anisotropie pour obtenir un maillage de contrôle en intégrant explicitement les lignes de champs des directions principales de courbure sur la surface. Bien qu'elle soit adaptée à l'anisotropie, leur méthode ne donne pas de résultats satisfaisants en terme de qualité du remaillage, et cela à cause de deux inconvénients majeurs : l'espacement irrégulier des arêtes du maillage de contrôle, ainsi que des boucles ouvertes (Figure 1.3-Droite).

Nous proposons une nouvelle approche pour l'extraction automatique de maillages de contrôle. Nos maillages de contrôle sont non seulement adaptés à l'anisotropie, mais présentent également un espacement régulier de leurs arêtes.

Chapitre 2

Étape B : Génération de champs de direction de guidage

2.1 Introduction

Pour créer nos maillage de contrôle, nous commençons par créer un champ de direction qui orientera le placement des arêtes. En d'autres termes, ce champ de direction capture l'anisotropie de la surface.

Un champ de direction $\vec{u}$ est un champ de vecteurs unitaires tangents : $\vec{u} \cdot \vec{u} = 1$, $\vec{u} \cdot \vec{n} = 0$, où $\vec{n}$ est la normale de la surface. Les champs de direction que nous manipulons à cette étape ont la propriété d'être à symétrie d'ordre N, ce qui est caractérisé mathématiquement de la façon suivante (une définition plus rigoureuse est donnée au Chapitre E). En un point de la surface, une direction à symétrie d'ordre N, est un ensemble de directions (vecteur unitaire tangent) invariant par rotation de $2k\pi/N$:

$$\vec{d} = \{\vec{u}_k = R_{\vec{n}}(\vec{u}_0, 2k\pi/N)\}$$

où $R_{\vec{n}}(., \theta)$ dénote la rotation d'angle θ du vecteur $\vec{u}_0$ autour de la normale $\vec{n}$ à la surface.

Cette étape génère automatiquement un champ de direction de guidage adapté à l'anisotropie afin de servir de base à l'étape suivante (la paramétrisation globale périodique). Pour produire un tel champ de directions de guidage, nous proposons deux nouvelles approches :

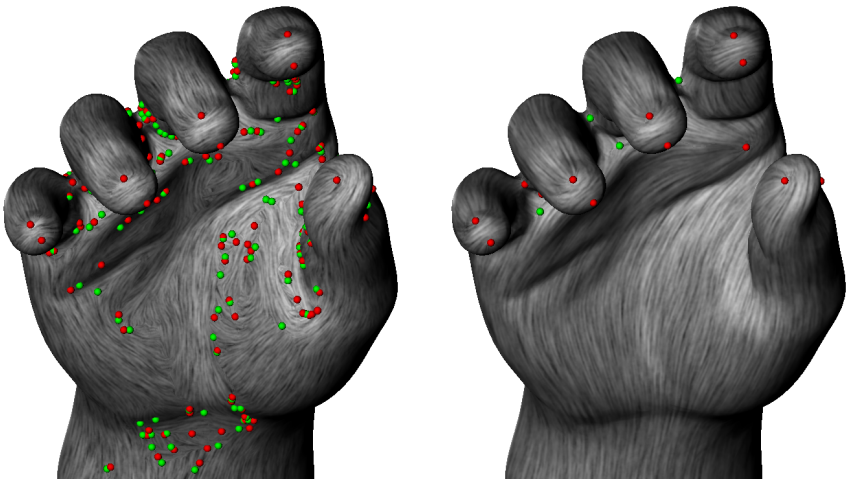


FIG. 2.1: *Gauche : estimation des directions principales ; Droite : directions principales lissées. (les points indiquent des singularités du champ)*

2.2 Relaxation globale de l'estimation des directions principales de courbure

Nous commençons par trouver deux champs de directions orthogonaux qui correspondent aux deux directions de courbure principales de la surface. Pour ce faire, nous commençons par estimer le tenseur de courbure en chaque sommet du maillage triangulé. Il existe de nombreuses méthodes pour estimer le tenseur de courbure en un sommet (voir [Pet01] pour un récapitulatif). Nous avons choisi la méthode présentée dans [CSM03] qui intègre le tenseur du courbure sur une région arbitraire autour du sommet au lieu de n'utiliser que son voisinage immédiat. Les deux directions principales sont alors trouvées en calculant les deux vecteurs propres k_1 et k_2 qui correspondent aux deux plus grandes valeurs propres du tenseur de courbure. L'anisotropie des surfaces dont nous parlons est définie par ces deux directions principales. Son amplitude A est donné par le ratio $\left|\frac{k_1}{k_2}\right|$. Comme ce sont des vecteurs propres, ils peuvent être changés en leurs opposés, et donc le champ de direction correspondant à ces directions principales est à symétrie d'ordre 2.

La paire de champs de directions orthogonaux sera utilisée pour guider notre méthode de paramétrisation globale périodique (PGP). Cependant, ces champs estimés ne peuvent pas être utilisés en pratique pour deux raison :

1. les directions de courbure principales de la surface sont indéfinies dans les régions isotropes,

2. le champ de tenseurs présente des oscillations parasites, générant des détails micro-topologiques.

Afin de résoudre ces deux problèmes, nous lissons le champ de directions principales tout en extrapolant l'anisotropie aux régions isotropes. Il existe des méthodes de lissage du champ de direction qui prennent en compte les symétries du champ, comme, par exemple, celles présentées dans [WL01] et [HZ00]. Cependant, ces deux méthodes sont limitées car très conteuses en terme de temps de calcul, à cause de l'approche par relaxation locale pour l'une, et de la non-linéarité de la fonction objectif pour l'autre.

Nous présentons une nouvelle approche par relaxation globale qui utilise une simple procédure de minimisation quadratique. Plus précisément, nous introduisons une procédure qui lisse un champ de direction donné et les extrapole dans les parties du maillage sur lesquelles les directions associées sont mal définies (Figure 2.1). Nous introduisons énergie suivante, avec pour variables les α_i définis comme les angles entre la direction du champ $\vec{K}_i$ et une direction de référence $\vec{H}_i$ dans le plan tangent du sommet i :

$$R = (1-\rho)\underbrace{\sum_{\forall v_i} A_i \left\| \begin{pmatrix} \cos\alpha_i \\ \sin\alpha_i \end{pmatrix} - \begin{pmatrix} \cos\alpha_i^0 \\ \sin\alpha_i^0 \end{pmatrix} \right\|^2}_{\text{terme d'ajustage}} + \underbrace{\rho \sum_{\forall T} R_T}_{\text{terme de lissage}}$$

où le coefficient ρ peut être défini par l'utilisateur et correspond à l'intensité du lissage (dans tous nos exemples, $\rho = 0.8$). Le terme de lissage R_T sur un triangle T minimise les variations de α sur T et est donné par :

$$R_T = \sum_{i=0}^{2} \lambda_i^T \left\| \begin{pmatrix} \cos\alpha_{i\oplus 2} \\ \sin\alpha_{i\oplus 2} \end{pmatrix} - \begin{pmatrix} \cos\beta_i & \sin\beta_i \\ -\sin\beta_i & \cos\beta_i \end{pmatrix} \begin{pmatrix} \cos\alpha_{i\oplus 1} \\ \sin\alpha_{i\oplus 1} \end{pmatrix} \right\|^2$$

où, β_i dénote l'angle entre $\vec{H}_{i\oplus 1}$ et $\vec{H}_{i\oplus 2}$, et où $\oplus$ dénote l'addition modulo 3 (qui "tourne autour" des 3 arêtes des triangles). Les termes constats λ_i^T utilisés pour définir les intégrales sur les triangle, dépendent seulement de la géométries des triangles, et sont explicités dans l'équation F.14 du chapitre F.

Nous minimisons la fonctionnelle d'énergie par rapport aux inconnues $(\cos\alpha_i, \sin\alpha_i)$. Ceci rend possible d'utiliser des solveurs numériques efficaces et permet d'obtenir une convergence plus rapide comparé à l'approche globale précédente. Étant donné $(\cos\alpha_i, \sin\alpha_i)$ nous recalculons le champ de direction $\vec{K}_i = \cos\alpha_i \vec{H}_i + \sin\alpha_i \vec{H}_i \times \vec{N}_i$ and $\vec{K}_i^{\perp} = \vec{N}_i \times \vec{K}_i$, où $\vec{N}_i$ est le vecteur normal.

En raison de la symétrie d'ordre 2 de notre champ de direction, nous satisfaisons l'égalité modulo π la résolvant pour les variables intermédiaires $\tilde{\alpha}_i = 2\alpha_i$. En pratique, ceci correspond

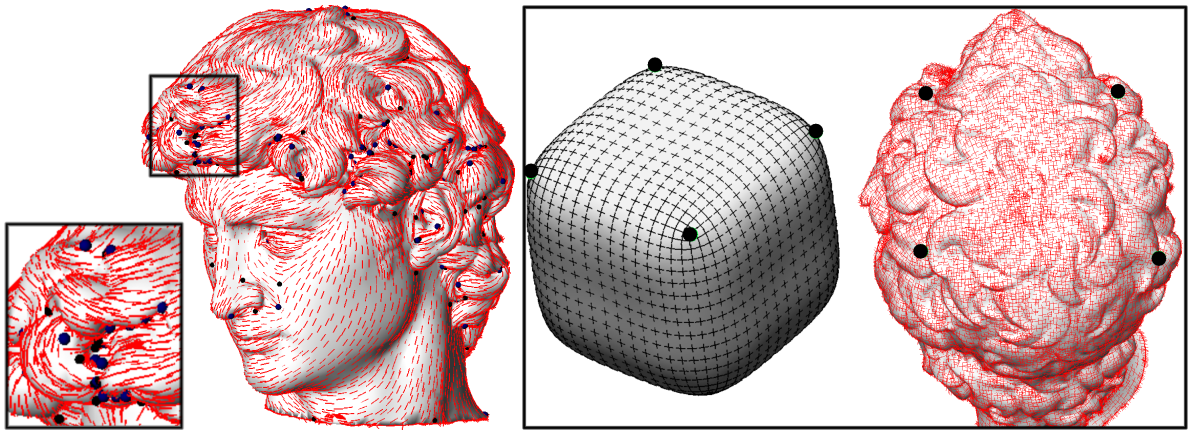


FIG. 2.2: *Gauche : un champ de directions principales à symétrie d'ordre 2 lissé sur un maillage avec de fortes variations géométriques locales ; Droite : un champ de directions principales à symétrie d'ordre 4 avec un ensemble de 4 singularités d'indice 1/4 définies par l'utilisateur.*

simplement à diviser tous les α_i^0 et les β_i par 2, puis minimiser la fonctionnelle d'énergie, et enfin multiplier les α_i par 2.

Nous pouvons aussi lisser le champ de directions avec une symétrie d'ordre 4 pour obtenir un champ plus "lisse" grâce à de symétries (voir Figure 2.2-Droite). De même que dans le cas de symétries d'ordre 2, ceci est obtenu en résolvant l'équation avec des variables intermédiaires $\tilde{\alpha}_i = 4\alpha_i$.

Lisser les champs de direction principales de courbures en utilisant la fonctionnelle d'énergie définie ci-dessus, (en symétrie d'ordre 2 ou 4) produit en général des champs de direction de guidage raisonnablement bons, avec un ensemble de singularités bien placées. C'est typiquement idéal pour des surfaces avec de faibles variations géométriques locales.

Cette partie de l'algorithme est expliquée en détail dans la section D.4 et a fait partie de notre article publié dans *ACM Transactions on Graphics* [RLL$^+$].

2.3 Contrôle des singularités : construction de champs de direction d'ordre N

Pour des surfaces avec de fortes variations géométriques locales la méthode de lissage peut mal fonctionner puisque l'on ne peut pas contrôler explicitement la position des singularités. Ces singularités correspondent à des pôles (voir plus bas pour une définition plus formelle). Par exemple, considérant la tête de la statue de David (du projet Digital Michelangelo de Stanford)

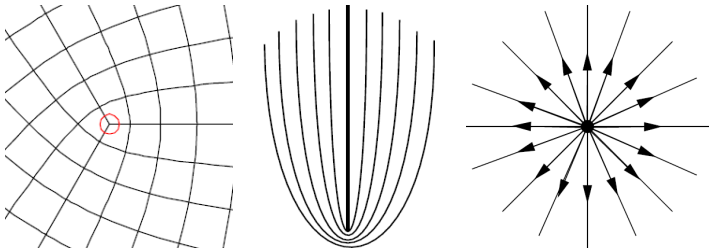


FIG. 2.3: *Gauche : une singularité d'index 1/4 ; Milieu : une singularité d'index 1/2 ; Droite : une singularité d'index 1.*

[LPC+00]), toutes les singularités générées par la méthode n'ont pas nécessairement une signification globale La figure 2.2-Gauche montre un champ de directions principales d'une surface lissé avec symétrie d'ordre 2. Nous pouvons voir de nombreuses singularités sans réelle signification (points noirs) dans la région des cheveux qui ne peuvent pas être lissées à cause des fortes variations géométriques.

En considérant que la géométrie globale de la tête est similaire à un cube unité, l'ensemble des singularités que l'on désire peut être similaire à celui montré sur la figure 2.2-Droite, qui est un champ de direction à 4 symétries avec seulement 4 singularités d'index 1/4 chacune dans la région des cheveux. Pour obtenir un tel champ de direction, nous avons besoin d'un contrôle *explicite* des singularités du champ de direction et de leurs indices plutôt que d'essayer simplement de les éliminer par lissage.

Avant de continuer l'exposé, nous allons rappeler brièvement la définition d'une singularité d'un champ de vecteur tangent défini sur une surface S. Une singularité est un point $p \in S$ tel que $\vec{v}(p) = \vec{0}$. Les singularités peuvent être de divers types, en fonction du comportement du champ autour de la singularité.

Ces "comportements" du champ de direction autour de la singularité sont classifiés par un *index* en topologie des champs de vecteurs (voir par exemple [Tri02]), et qui correspond intuitivement au nombre de tours que le vecteur fait sur lui même autour de la singularité. La figure 2.3 montre trois singularités d'index index 1/4, 1/2 et 1 respectivement. (l'index d'un champ de direction d'ordre N est un multiple de $1/N$). Notons que sur une surface de genre g, le théorème de Poincaré-Hopf affirme que la somme des indices des singularités vaut nécessairement $2-2g$.

Le problème résolu par notre algorithme peut être formalisé de la façon suivante : étant donné une surface triangulée S de genre g, et un sous ensemble de sommet $V_s = \{v_i\} \in V$ avec un ensemble d'indices désirés I_i (de telle sorte que $\sum I_i = 2-2g$), notre algorithme est capable de générer un champ de vecteur tel que l'index des sommets soit égal à I_i pour les éléments de V_s, et soit nul sinon. Notons que le champ de vecteurs généré par notre algorithme est de norme

unitaire (c'est un champ de directions).

Les constructions classiques de champs de direction ont été très étudiées, comme par exemple dans [TSH00, The02, ZMT04]. Notons aussi les travaux de [ZHT05] par exemple qui présentent des méthodes de construction des champs de direction à symétrie d'ordre 2 (ou champs de tenseurs). Cependant, les méthodes existantes ne contrôlent pas les singularités des champs générés. Quant aux constructions de champs de direction à symétrie d'ordre 4 sur des surfaces de genre arbitraire, il n'en existe pas à notre connaissance dans la littérature actuelle. Ceci est dû au manque de définitions formelles pour la symétrie d'ordre N en général, et au manque d'outils mathématiques pour expliquer le lien entre la topologie de tels champs de direction (constituée par les singularités et leurs indices) et la topologie de la surface sur laquelle ils sont définis, c'est à dire qu'il n'existe pas de généralisation du théorème de Poincaré Hopf pour les champs de symétries d'ordre N en général.

Pour définir un mécanisme de construction d'un champ de direction d'ordre N, nous étudions la structure topologique fondamentale. Nous capturons la structure topologique d'un champ de direction à symétrie d'ordre N $\vec{d}$ en définissant la notion de *nombre de tours* $T_{\vec{d}}(\gamma)$ d'un cycle γ,

$$T_{\vec{d}}(\gamma) = \frac{1}{2\pi} \oint_{\gamma} (\kappa_{\vec{d}} - \kappa_{\gamma}) ds \tag{2.1}$$

où, $\kappa_{\vec{d}}$ est la courbure géodésique de la direction et κ_{γ} est la courbure géodésique du vecteur tangent au cycle.

Nous montrons que cette quantité topologique caractérise les singularités du champ de direction en calculant le "nombre de tours" d'un cycle autour d'un point singulier Le nombre de tours est lié à la définition de l'*index* en topologie des champs de vecteurs par la simple équation suivante (prouvée dans la section E.2.6) :

$$I_{\vec{d}}(P) = 1 + T_{\vec{d}}(\partial\Omega(P)) \tag{2.2}$$

où P est un point singulier et $\partial\Omega(P)$ est le bord d'un disque $\Omega(P)$ recouvrant P.

En utilisant ce nouvel outil de compréhension de la topologie des champs de vecteurs, nous introduisons la notion de *saut de période* (définie ci-dessous) et nous l'utilisons pour construire une nouvelle représentation discrète pour les champs de direction. Le champ de direction est défini de la façon suivante sur un maillage triangulé :

- faces : la direction est définie par l'angle θ qu'elle forme avec une direction de référence donnée par la direction de l'une des arêtes ($\vec{d}_0$) du triangle ;
- arête (duale) : un entier que nous appelons saut de période k qui indique comment la direction varie le long de l'arête duale entre deux faces qui partagent une arête ;

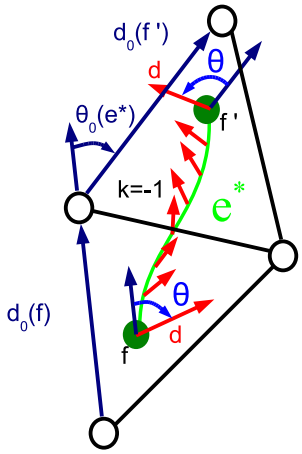


FIG. 2.4: *Structure de données de notre représentation discrète de champ de direction.*

– sommet : la direction reste indéfinie (le sommet peut être conceptualisé comme un trou de la surface)

En utilisant cette représentation discrète, nous montrons que les indices du champ de direction désiré en chaque sommet est lié aux indices des sommets d'un champ de base $\vec{d}_0$, et à la somme des sauts de période le long du bord de leur face duale de la façon suivante (voir section E.3.2) :

$$I_{\vec{d}}(v) = I_0(v) + \sum_{e^* \in \partial v^*} \frac{k(e^*)}{N} \tag{2.3}$$

où v est le sommet, $I_0(v)$ est l'indice du champ de base $\vec{d}_0$ calculé en utilisant les équations 2.1 et 2.2, ∂v^* est le bord de la face duale v^*, et e^* est une arête duale de ∂v^*.

L'idée de base de notre mécanisme de construction est de trouver les bonnes valeurs de sauts de période k et d'angles θ sur tout le maillage, ce qui correspond respectivement à définir la topologie et la géométrie du champ de direction. La procédure de construction se décompose donc en deux étapes :

1. *Étape topologique*

   Dans cette étape, l'utilisateur spécifie un ensemble de sommets singuliers ainsi que leurs indices, qui doivent nécessairement être des multiples de $1/N$ dont la somme soit égale à la caractéristique d'Euler du maillage (puisque les indices vérifient le théorème de Poincaré-Hopf). En utilisant l'équation 2.3, nous obtenons les valeurs des sauts de périodes k, ce qui fixe la topologie du champ de direction.

2. *Étape géométrique*

   Une fois fixée la topologie du champ de direction, l'utilisateur spécifie la géométrie du champ de direction par l'intermédiaires de contraintes directionnelles sur certaines faces

du maillage. De cette façon, le champ de direction peut être adapté à l'anisotropie du maillage triangulé.

Les valeurs des angles *theta* sont déterminées de façon à ce que le champ généré soit le plus lisse possible pour la topologie spécifiée et les contraintes directionnelles données. Ceci est obtenu en minimisant la forme quadratique suivante :

$$E = \sum_{e^*=(f,f')\in E^*} \left(\theta(f') - \theta(f) + \theta_0(e^*) + \frac{2\pi k(e^*)}{N} \right)^2$$

où f et f' sont les faces qui partagent l'arête orientée e, et $\theta_0(e^*)$ est l'angle entre $\vec{d_0}(f')$ et $\vec{d_0}(f)$ mesuré après avoir aplati la paire de triangles par rotation autour de leur arête commune e.

2.4 Conclusions

En ce chapitre nous avons proposé deux nouvelles approches pour produire automatiquement des champs de direction de guidage adapté à l'anisotropie afin de servir de base à l'étape suivante (la paramétrisation globale périodique).

La première approche estime les directions courbures principales de la surface (2-symétrie) et les emploie comme champs de direction de guidage. Nous avons expliqué la nécessité de lisser les champs estimés de direction des régions anisotropes aux régions isotropes. Des méthodes locales et globales existantes de relaxation de champs de direction sont revues. Puis, nous avons présenté notre méthode globale de relaxation qui emploie une énergie plus simple fonctionnelle que l'approche globale existante et par conséquent réalisons une convergence plus rapide. D'ailleurs, nous avons discuté les avantages de lisser un champ principal de direction de 2-symétries avec la 4-symétrie afin de réaliser même des champs de guidage "plus lisse" pour le but d'extraction de maillage de contrôle.

Cette méthode donne habituellement des bons champs de direction de guidage avec l'ensemble de singularités bien-placé. Elle est en général idéale pour des surfaces avec des petites géométrie variations locale. Néanmoins, pour des surfaces avec des variations géométriques locales élevées, par exemple la région de cheveux de David de Michaelangelo, la méthode ne pourrait fonctionner bien puisqu'on ne peut commander la distribution des singularités. Cet inconvénient mène à la recherche de la deuxième approche qui permet de contrôler directe de la topologie et de la géométrie du champ de direction de guidage en indiquant un placement d'un ensemble de singularités du champ et d'un ensemble de contraintes géométriques respectivement.

Les deux approches sont expliquées plus en détail dans l'annexe D.4 et E respectivement. La première approche a fait partie de notre article publié dans *ACM Transactons on Graphics* [RLL+] et la deuxième approche a fait partie de notre rapport technique [RVLL06] ainsi que notre article publié dans *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization '06)* [LV+06].

Chapitre 3

Étape C : Paramétrisation Globale Périodique

3.1 Introduction

A partir du champ de direction calculé à l'étape B, nous allons voir comment générer un système de coordonnées de l'objet dont les axes suivent le champ de direction.

L'idée principale de notre approche est de trouver une paramétrisation qui ait "une signification géométrique". Dans les étapes suivantes, les lignes d'iso-valeurs de cette paramétrisation permettront d'extraire un maillage de contrôle initial. Cette approche d'extraction du maillage de contrôle fondée sur l'extraction d'iso-valeurs présente les trois avantages suivants :

1. Il existe une différence fondamentale entre les maillages et les représentations splines : un maillage est un échantillonnage désordonné de la géométrie tandis que les splines requièrent la définition d'un espace paramétrique associé a l'objet. La génération d'un domaine paramétrique lié au maillage permet de rendre compatible ces deux familles d'objets.
2. Il existe de nombreuses façon de paramétrer une surface. Afin d'en prendre une qui ait un "sens d'un point de vue géométrique", il est intéressant de suivre des directions caractéristiques de la surface telles que sa courbure. Les lignes d'iso-valeurs de la paramétrisation sont ainsi guidées par une paire de champs de directions qui suivent l'anisotropie de la surface. La paramétrisation étant bien adaptée à la géométrie, les lignes d'iso-valeurs définissent un maillage de contrôle quadrilatéral de la surface qui est assez naturel i.e. similaire à un maillage qui aurait été créé manuellement. De plus, [d'A00] confirme que ce type de maillages de contrôle est optimal d'un point de vue théorique

3. La paramétrisation est définie par une minimisation *globale* d'une énergie définie par des fonctions trigonométriques (et donc périodiques) des paramètres s et t (coordonnées dans l'espace paramétrique) de la surface (voir Section F.2). Cette approche permet d'assurer une distribution régulière des arêtes du maillage de contrôle sur l'ensemble de l'objet, contrairement aux approches de suivit de ligne de flux [ACSD+03].

3.2 La paramétrisation

3.2.1 Les entrées

Hormis le maillage original, notre algorithme utilise les entrées suivantes :

– deux champs de direction (2- ou 4-symétrie) orthogonaux $\vec{K}$ et $\vec{K}^{\perp}$ qui correspondent aux champs de courbures principales de la surface. (En pratique, $\vec{K}^{\perp}$ peut être déterminé par le produit vectoriel de $\vec{K}$ par la normale de la surface.)

– la longueur moyenne des arêtes du maillage de contrôle ω, souhaitée par l'utilisateur

3.2.2 Le principe

L'idée générale de notre méthode est de trouver une paramétrisation abstraire de la surface en générant deux fonctions périodiques s et t définies sur la surface S, dont les gradients soient respectivement alignés avec $\vec{K}$ et $\vec{K}^{\perp}$. Les fonctions s et t sont trouvées en minimisant l'énergie objectif suivante :

$$F \quad = \quad \int_S \left(\|\nabla s - \omega \vec{K}\|^2 + \|\nabla t - \omega \vec{K}^{\perp}\|^2 \right) dS$$

Nous optimisons l'énergie définie précédemment par une simple procédure de minimisation quadratique avec les variables périodiques $(u_1, u_2) = (\cos s, \sin s)$ (resp $(v_1, v_2) = (\cos t, \sin t)$). Nous utilisons une intégration implicite des paramètres s et t, ce qui permet de s'affranchir des problèmes d'espacement des lignes de flux et de la génération de lignes infinies obtenues par intégration explicite (e.g. [ACSD+03]). Nous avons définit l'énergie quadratique suivante (échantillonnée sur les arêtes de la triangulation).

$$F_{edge} \quad = \quad \sum_{\vec{e}_{ij} \in E} \left\| \begin{pmatrix} u_{1i} \\ u_{2j} \end{pmatrix} - \begin{pmatrix} \cos(\beta_{ij}) & -\sin(\beta_{ij}) \\ \sin(\beta_{ij}) & \cos(\beta_{ij}) \end{pmatrix} \begin{pmatrix} u_{1j} \\ u_{2j} \end{pmatrix} \right\|^2$$

$$\vec{K}_{ij} = 0.5(\vec{K}_i + \vec{K}_j) \quad ; \quad \beta_{ij} = \omega \vec{K}_{ij} \cdot \vec{e}_{ij}$$

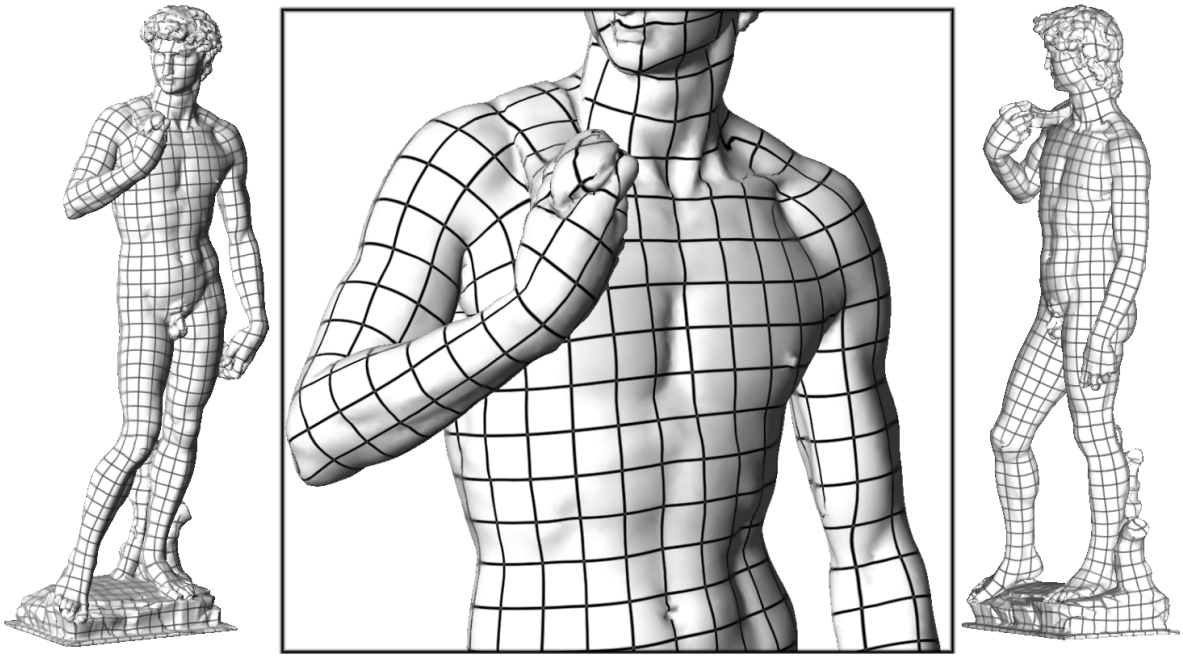


FIG. 3.1: *Paramétrisation globale périodique du modèle du "David" (lignes d'iso-valeurs).*

Remarque : pour des champs de direction à symétrie d'ordre N, une variante de cette équation est utilisée à la place, ainsi que nous l'expliquons dans la section F.2.4.

Une fois les variables périodiques $(u_1,u_2) = (\cos s, \sin s)$ (resp. $(v_1,v_2) = (\cos t, \sin t)$) calculées, nous devons récupérer la variable s (resp. t) correspondante. Pour ce faire, il faut lever l'ambiguïté sur la valeur de s entre les différentes valeurs possibles $s + 2k\pi$ (en déterminant k).

Ceci est fait individuellement pour chaque triangle en commençant à un sommet du triangle, et en propageant le long des 3 arêtes. Nous choisissons parmi les valeurs possibles pour s_j la plus proche de la valeur optimale $s_i + \beta_{ij}$, où $\beta_{i,j} = \omega \vec{K}_{ij} \cdot \vec{e}_{ij}$ correspond au déplacement optimal le long de l'arête (i, j).

Une fois la paramétrisation reconstruite dans chaque triangle individuel nous devons vérifier certaines contraintes de validité Plus précisément, les angles autour d'un sommet doivent avoir une somme de 2π, les angles autour d'un triangle doivent avoir une somme de π, et les voisinages des sommets doivent satisfaire la condition de "compatibilité en roue" (voir par exemple [SdS01]). Les sommets et triangles qui violent des conditions seront nommés "singuliers". Par exemple, chaque singularité du champ de direction de guidage (étape B) générera automatiquement de tels sommets et triangles.

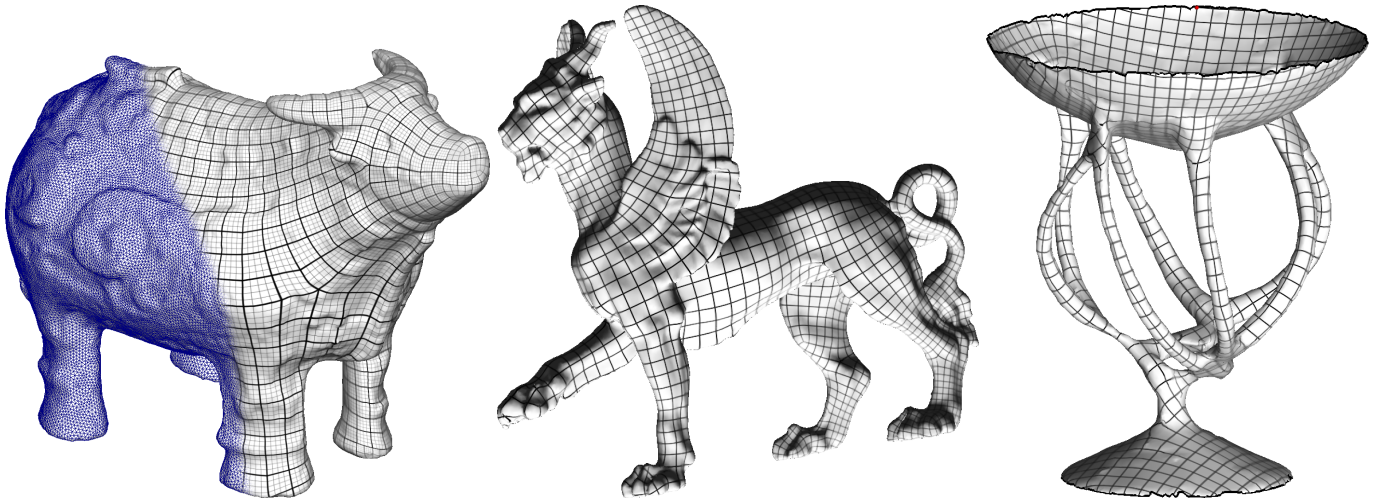


FIG. 3.2: *Paramétrisation Globale Périodique avec différent genre (iso $k\pi$ lignes).*

3.3 Resultats

La méthode paramétrisation globale périodique de (PGP) a été implémenté dans la module de traitement de géométrie dans le logiciel Graphite [Gra].

Figures 3.1 et 3.2 montrent la paramétrisation finale avec les iso $k\pi$ lignes. Ces figures montrent que notre méthode marche sur des modèles de n'importe quel genre, aussi bien que des modèles avec des frontières. D'ailleurs, même sur des complexes modèles le nombre de singularités produites par notre méthode reste très petit ($2-3\%$ des triangles).

Table 3.1 donne les statistiques et mesure de temps de notre méthode. Les temps sont mesurés sur une machine de 1.7 GHz.

3.4 Conclusion

En ce chapitre, visant à extraire les maillage de contrôle quadrilatérals de haute qualité des surfaces triangulées du genre arbitraire, nous avons proposé une nouvelle paramétrisation globale périodique. Un agencement de cellules peut être extrait comme iso-lignes de la paramétrisation globale (nous allons expliquer cela au chapitre suivant). Cet agencement de cellules est utilisé comme le maillages de contrôle de la surface triangulée originelle. L'avantage principal de notre paramétrisation globale périodique par rapport aux techniques paramétrisation globales précédentes est sa capacité d'aligner la paramétrisation avec les champs orthogonaux de direction de guidage. En utilisant les directions courbures principales en tant que champs de direction de guidage, l'agencement de cellules est adapté à l'anisotropie.

Model	♯Δ	Algorithm	Stretch	Shear	time
Horse	20K	Gu et al.	6.777	0.07	NA
		PGP	1.07	0.20	45 s.
Bunny	25K	Gu et al.	2.65	0.042	NA
		PGP	1.029	0.167	58 s.
Bull	34.5K	Sander et al.	1.030	0.1558	1 min. 11 s.
		PGP	1.064	0.1774	1 min. 26 s.
Camel	78K	Sander et al.	1.053	0.227	3 min. 51 s.
		PGP	1.048	0.1596	5 min. 46 s.
David	200K	PGP	1.121	0.2398	17 min. 35 s.
Lion	400K	PGP	1.123	0.1728	33 min. 42 s.

TAB. 3.1: *Statistiques et mesure de temps de notre méthode. Les nombres sont comparés, quand les données sont disponibles, à la méthode paramétrisation globale de Gu* et al.*, et à celle de Sander* et al.*. Le nombre de singularités pour Sander* et al. *est le nombre de sommets sur la coupure.*

La paramétrisation est obtenue par une optimisation globale de deux fonctions scalaires périodiques de sorte que leurs gradients soient aussi tangentiels comme possible aux champs de direction de guidage. De cette façon, la qualité de nos maillages de contrôle sont beaucoup améliorés par rapport aux ceux produits des méthodes précédentes (par exemple [ACSD+03]) : l'espacement des arêtes est irrégulier, ainsi que des boucles fermés.

Cette partie de l'algorithme est expliquée plus en détail dans l'annexe F et a fait l'objet d'un article publié à *ACM Transactions on Graphics* [RLL+].

Chapitre 4

Étape D : Extraction d'agencement de cellules

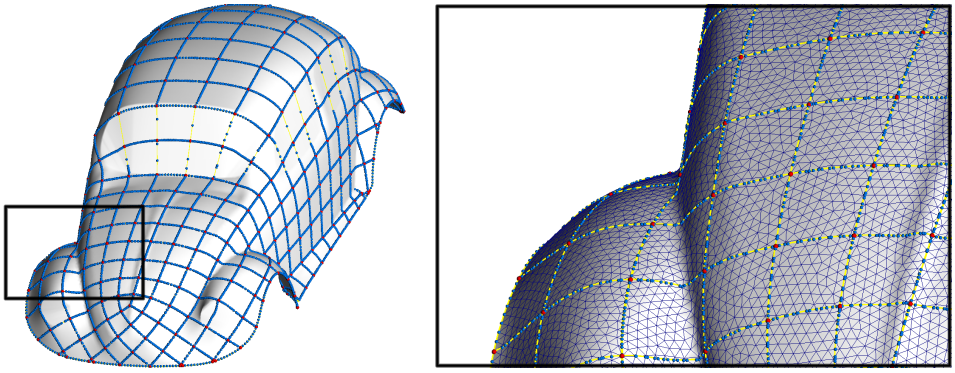


FIG. 4.1: *Extraction d'un agencement de cellules par détermination des courbes de niveaux correspondant aux lignes d'iso-*$2k\pi$.

4.1 Introduction

Une fois calculée la paramétrisation locale dans chaque triangle T, nous construisons l'agencement de cellules (Figure 4.1). Dans le cadre étudié, les bords des cellules correspondent aux iso-$2k\pi$ lignes pour les paramètres s et t. Ces lignes définissent un ensemble de segments dans chaque triangle. Il faut noter que si un triangle T est traversé par une iso-$2k\pi$ ligne de s (resp. de t), le même triangle déplacé de $(2a\pi, 0)$ dans l'espace paramétrique sera traversé à cet endroit par une iso-$2(k+a)\pi$ ligne de s (resp. de t). Les extrémités de ces segments indépendants se recollent sur les bords du maillage, et les segments forment des lignes polygonales continues.

Algorithm 1 Construction de l'agencement de cellules

calculer l'agencement de cellules :
pour chaque triangle T
  si Test non-singulier
    pour $k \in \mathbb{N}$ tels aue $2k\pi \in [\min_T(s), \mathrm{Max}_T(s)]$
      Ligne $l \leftarrow$ ligne d'équation$(s = 2k\pi)$
      Segment $S \leftarrow l \cap T //dans\ l'espace\ paramtrique$
      stocker S in T
      stocker les extrémités de S dans les arêtes correspondantes de T
    fin$//pour$
    répéter les procédures ci-dessus pour$t$
  fin$//si$
fin$//pour$
pour chaque arrête$e$
  fusionner les extrémités de segment stockées dans e qui ont la même location géométrique en 3D
fin$//pour$
pour chaque triangle T
  calculer les intersections entre les arêtes stockées dans T
fin$//pour$

4.2 Extraction

L'algorithme 1 extrait l'agencement de cellules. Chaque triangle stocke une liste de segments, et chaque bord stocke une liste de extrémités de segment. L'algorithme calcule les différents segments définis par les intersections des triangles avec les lignes $(s = 2k\pi)$ et $(t = 2k\pi)$. Toutes les positions 2D et 3D des extrémités des segments sont calculées. L'algorithme fusionne les extrémités de segment le long des bords et intersecte les segments à l'intérieur des triangles ajoutant les intersections en tant que nouvelles extrémité. Tous segments balançants sont enlevés (chaque extrémité de segment de valence 1 est "grignoté" jusqu'à une extrémité de valence plus que 2 est produite). Toutes opérations ci-dessus pour extraire l'agencement de cellules sont mises en application en utilisant notre structure de données complexe cellulaire présentée en sous-section suivante.

Après avoir trouvé les intersections de ces iso-lignes, nous obtenons un agencement de

cellules qui composé principalement de cellules quadrilatérals. Cependant, apparaissent aussi quelques cellules à *N*-côtés correspondant aux singularités de la paramétrisation globale. Nous marquons aussi comme singulières les cellules qui contiennent les sommets singuliers et/ou les triangles singuliers ; puis nous appliquons les étapes suivantes pour traiter ses cellules singulières.

4.2.1 Structure de données : plongement du complexe cellulaire

Un aspect clé de l'algorithme est d'utiliser une structure de données efficace pour représenter les ensembles de lignes plongées dans une surface et trouver leur intersection. Nous avons introduit pour cela une nouvelle structure de données nommée *Plongement du complexe cellulaire* qui va être expliquée dans la section suivante (voir aussi Annexe C).

Définitions : abstrait complexe cellulaire

Cette section donne les définitions et les notations classiques pour l'abstrait complexes cellulaires. Le lecteur est référé à [Mas91] pour plus de détails.

- Soit Γ un ensemble fini.
  Les éléments de Γ sont nommés les *cellules* de Γ ;
- soit $\leq$ un ordre strict sur Γ, c.-à-d. une relation réfléchie, antisymétrique et transitive.
  La relation $\leq$ est référé comme *relation bordante* ;
- considérer la fonction $dim : \gamma \rightarrow \mathbb{N}$ caractérisé par :
  $\forall \gamma, \gamma' \in \Gamma \times \Gamma, \gamma \leq \gamma'$ and $\gamma \neq \gamma' \Rightarrow$
  $dim(\gamma) < dim(\gamma')$
  La fonction dim est nommée *la fonction d'ordre*.
  Une cellule γ tels que $dim(\gamma) = k$ est une *k-cellule*, et k est la *dimension* de γ ;
- la fonction dim rend une partition de Γ en $\Gamma^0 \ldots \Gamma^n$ définis par : $\forall \gamma \in \Gamma^k, dim(\gamma) = k$.
  La dimension dimension n est la dimension du complexe cellulaire Γ.
- le *bord* $B(\gamma)$ d'une cellule γ est défini par :
  $B(\gamma) = \{\gamma' \in \Gamma | \gamma' \neq \gamma \text{ et } \gamma' \leq \gamma\}$
- la *étoile* γ^* d'une cellule γ est défini par :
  $\gamma^* = \{\gamma' \in \Gamma | \gamma \leq \gamma'\}$
- une *réalisation géométrie* de Γ est un isomorphisme qui met Γ en correspondance avec un ensemble d'ensembles ouverts Γ'. La relation d'ordre $\leq$ est portée à Γ' comme suit :
  $\gamma_1 \leq \gamma_2 \Leftrightarrow \gamma_1' \in \partial \gamma_2'$

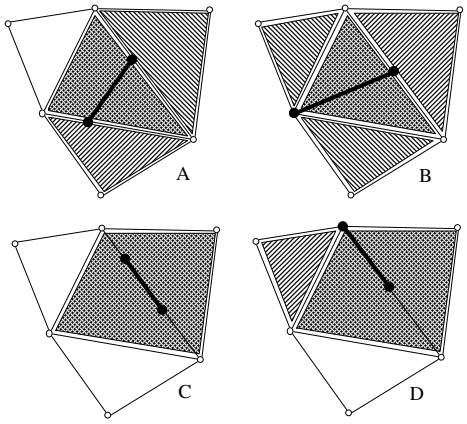


FIG. 4.2: *La fonction em* : de trouver le plongement d'un segment du maillage de contrôle* Γ *(segment gras), nous trouvons d'abord dans quelle cellule ses sommets sont plongés, et calculons les intersections de l'étoile de ces cellules (zones à tiret).*

où $\partial\gamma'_2$ dénote le bord de γ'_2. La réalisation géométrie est dite *conforme* si les cellules géométriques de Γ' sont disjointes.

Plongement de ligne dans un surface

Pour représenter un maillage de contrôle plongé dans une surface, nous utilisons deux abstraits cellulaires complexes et une relation pour les connecter :

- le maillage de contrôle peut être représenté par un complexe cellulaire 1D $\Gamma = (\Gamma^0, \Gamma^1, \leq)$;
- la surface peut être représenté par un complexe cellulaire 2D $\Sigma = (\Sigma^0, \Sigma^1, \Sigma^2, \leq)$;
- les cellules du maillage de contrôle sont connectés avec les cellules de la surface par une relation *inclusion* $\subseteq$ définie dans $\Gamma \times \Sigma$.

Noter que si la réalisation géométrique est conforme, nous avons :

$$\forall \sigma_1, \sigma_2 \neq \sigma_1 \in \Sigma \times \Sigma, \sigma_1 \cap \sigma_2 = \emptyset$$

puis, pour tous les $\gamma \in \Gamma$, nous avons au maximum un $\sigma \in \Sigma$ tels que $\gamma \subseteq \sigma$. Par conséquent, c'est naturel de représenter la relation $\subseteq$ par la fonction $em : \Gamma \to \Sigma \cup \{\emptyset\}$, définie par :

$$\begin{array}{ll} em(\gamma) = \sigma \text{ où } \gamma \subseteq \sigma & \text{si } \sigma \text{ existe} \\ em(\gamma) = \emptyset & \text{autrement} \end{array}$$

La fonction *em* sera référée comme fonction de *plongement* dans ce qui suit. Maintenant, nous supposons que la fonction de plongement *em* est *combinatoirement* représentée aux sommets de Γ. D'un point de vue implantation, nous supposons que chaque sommet de Γ a un

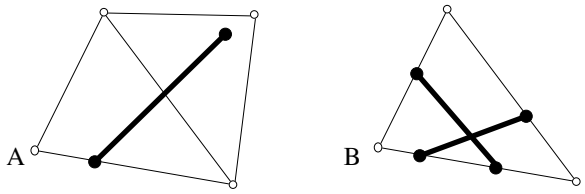


FIG. 4.3: *Les configurations des plongements invalides : A : em^* est vide ; B : maillage de contrôle non-conforme.*

pointeur à une cellule de Σ. Par exemple, si le maillage de contrôle Γ était numérisé manuellement sur la surface Σ, le système stocke dans chaque sommet de Γ quelle cellule deΣ à été choisie. Notre but est de maintenant répondre les questions suivantes :

1. De la définition de *em* sur les sommets de Γ, comment peut-on déduire la définition de *em* sur toutes les autres cellules de Γ ? Autrement dit, peut-on trouver dans quelle cellules de Σ les arêtes de Γ sont plongées ?
2. quelles sont les conditions combinatoires de validité qui rendent que cette extension de *em* possible ?
3. d'une configuration invalide, comment peut-on définir un algorithme qui renforce ces conditions ?

Pour répondre ces questions, nous considérons la fonction $em^* : \Gamma \rightarrow \mathscr{P}(\Sigma)$ définie par :

$$em^*(\gamma) = \bigcap \{em(\gamma')^\star | \gamma' \in B(\gamma) \cap \Gamma^0\}$$

Intuitivement, $em^*(\gamma)$ donne le plongement de tous les sommets de γ, et intersectent les étoiles de tous ces plongements. Trivialement, de la définition de em^*, nous avons :

$$\forall \sigma \in em^*(\gamma), \gamma \subseteq (\sigma \cup B(\sigma))$$

Figure 4.2 montre à quoi elle ressemble la fonction em^* pour certaines cas. Les deux sommets du maillage de contrôle (points noirs) sont plongés dans les cellules de la surface. Les étoiles de tous ces cellules sont affichées avec deux styles différents. l'intersection est montré par une pattern croisée.

Cependant, comme on voit dans les exemples montrés dans Figure 4.2-B et C, $em^*(\gamma)$ peut contenir trop de cellules. Afin de déterminer la prolongation $e\bar{m}(\gamma)$ de *em*, c.-à-d. de trouver la cellules de Σ dans laquelle γ est plongée, nous avons besoin de filtrer les cellules σ de $em^*(\gamma)$ tels que $\gamma \subseteq B(\sigma)$. Cela peut être facilement fait en trouvent la cellule de minimum dimension dans $em^*(\gamma)$:

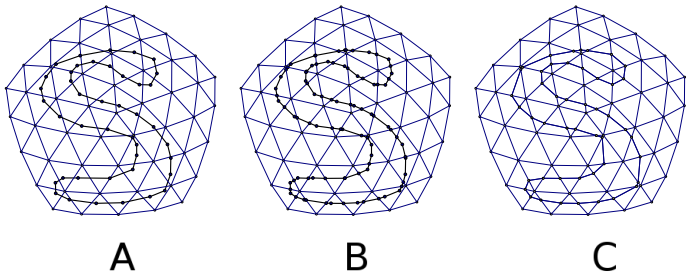


FIG. 4.4: *Classes de relations maillage de contrôle/surface. A : faiblement-plongé maillage de contrôle ; B : fortement-plongé maillage de contrôle ; C : maillage de contrôle réalisé.*

$$
\begin{aligned}
&\bar{em}(\gamma) = em(\gamma) \quad \text{si } \gamma \in \Gamma^0 \\
&\bar{em}(\gamma) = \sigma \in em^*(\gamma) \text{ tels que} \\
&\qquad dim(\sigma) = min\{dim(\tau) | \tau \in em^*(\gamma)\} \\
&\qquad\qquad \text{si } em^*(\gamma) \neq \emptyset, \\
&\bar{em}(\gamma) = \emptyset \quad \text{autrement}
\end{aligned}
$$

Les exemples C et D dans Figure 4.2, $em^*(\gamma)$ contient deux facettes et un segment. En sélectionnant la cellule de la plus basse dimension nous permet de trouver le bon segment. Noter que le $\bar{em}(\gamma)$ ainsi-défini est unique, sinon il contredirait la supposition de la conformité(Σ aurait deux cellules intersectantes). L'extension $\bar{em}$ de la fonction em répond à question 1.

Maintenant, nous voulons répondre à question 2, par exemple identifier les configurations invalides. Comme montrées dans Figure 4.3-A, un segment γ de Γ ne peut être plongé dans une cellule σ de Σ si ses extrémités sont plongées dans deux cellules σ_1 et σ_2 qui sont "trop loin", c.-à-d. si nous ne pouvons pas trouver une cellule σ dans Σ tels aue $\sigma_1, \sigma_2 \in (B(\sigma) \cup \{\sigma\})^2$. Cette condition correspond aussi à $em^*(\gamma) = \emptyset$.

L'autre configuration invalid c'est quand le maillage de contrôle Γ est non-conforme (Figure 4.3-B). Noter que si on a un maillage de contrôle non-conforme, on a deux segments $\gamma_1 \neq \gamma_2$ et $\gamma_1 \cap \gamma_2 \neq \emptyset$. Si $\bar{em}$ est défini, c'est à dire qu'on sait que les deux segments sont plongés dans la même cellule σ of Σ, $\bar{em}(\gamma_1) = \bar{em}(\gamma_2) = \sigma$, et que l'intersection $\gamma_1 \cap \gamma_2$ est plongé dans σ. Cela sera utilisé après pour faciliter la computation de $\gamma_1 \cap \gamma_2$.

Le domaine de définition de la fonction de plongement $\bar{em}$ permet de distinguer trois différentes classes de relations maillage de contrôle/surface. Nous appelons que le maillage de contrôle est :

– **faiblement-plongé** (Figure C.5-A) si la fonction $\bar{em}$ n'est définie que sur les sommets ;

– **fortement-plongé** (Figure C.5-B) si la fonction $e\bar{m}$ est définie sur les sommets et les segments, c'est à dire $\forall \gamma \in \Gamma, e\bar{m}(\gamma) \neq \emptyset$;
– **réalisé** (Figure C.5-C) si les cellules de γ sont plongés dans les cellules de la même dimension dans Σ, c'est à dire $\forall \gamma \in \Gamma, dim(e\bar{m}(\gamma)) = dim(\gamma)$.

Algorithme

Avec les notions ci-dessus, il est possible de répondre à la question 3 posée dans la section précédente en désignant un algorithme qui renforce la condition *réalisé* d'un *faible plongement* et éventuellement un maillage de contrôle *non-conforme* :

1. **assurer un fort plongement**
   Pour tous les $\gamma \in \Gamma^1$ tels que $e\bar{m}(\gamma) = \emptyset$, replacer γ avec le geodesique tracé sur Σ qui lie les deux extrémités de γ
2. **assurer la conformité du maillage de contrôle**
   • Pour tous les σ dans Σ^0, fusionner tous les sommets de Γ plongés dans σ.
   • Pour tous les σ dans Σ^1, ordonner les sommets de Γ plongés dans σ le long σ.
   • Pour tous les σ dans Σ^2, intersecter toutes les arêtes de Γ plongées dans σ (en utilisant un algorithme de "ligne balayante").
3. **réaliser le maillage de contrôle dans la surface**
   Insérer tous les sommets et arêtes manquants dans Σ.

Noter que l'algorithme utilise la plus information combinatoire possible : les calcules d'intersection sont systématiquement limités à la plus petite possible de Σ en appliquant la fonction $e\bar{m}$. Cela améliore à la fois robustesse et efficacité par rapport à une solution purement géométrique. Toutes les intersections sont soit ordonnées le long d'une arête (1D) soit dans une facette de Σ (2D). Les seuls calcules géométriques sont pour tracer les geodesiques (dans l'étape 1), pour ordonner les points le long des arêtes (dans l'étape 2) et pour trouver les intersection des segments dans des facettes (dans l'étape 2).

Complexité

Soit $n = |\Gamma|$ le nombre des cellules du maillage de contrôle et $m = |\Sigma|$ le nombre des cellules de la surface.

Dans l'algorithme, l'étape la plus couteau est le calcule des geodesique. La méthode que nous utilisons est celle de Chen et Han qui a une complexité de $O(m^2)$. Avec un heuristique que

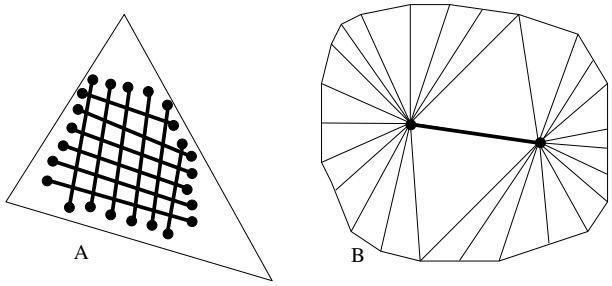


FIG. 4.5: *Configurations de pire-cas complexité.*

nous employons, la complexité est réduite à $O(mlog(m))$ (aux dépens de renvoyer pas toujours le géodésique le plus court, qui n'est pas problématique pour notre application).

1. **pire-cas complexité**

   Afin d'assurer la condition de fort plongement, il y a deux configurations dégénérées : si le maillage de contrôle est complètement plongé dans une simple facette, l'algorithme réduit à une construction d'un map planaire du maillage de contrôle, avec une pire-cas complexité de $O(n^2)$ (voir Figure 4.5-A). Si la surface a deux sommets avec une valence et le maillage de contrôle lie ces sommets, la partie la plus coûteuse est de calculer l'intersection étoile (intersection des deux ensembles), qui coût $O(m\log(m))$ (voir Figure 4.5-B).

2. **complexité en moyen**

   Pratiquement, le maillage de contrôle et la surface ont de résolutions semblable. Cela veut dire que chaque facette contient au maximum une intersection du maillage de contrôle, et la complexité est réduite à $O(n)$. Dans la pratique, tous les exemples montrés dans cette thèse sont effectués en moins qu'une seconde.

4.3 Édition manuelle et utilisation des géodésiques (optionnels)

Puisque l'agencement de cellules extrait automatiquement est sauvé sous la forme de *Plongement du complexe cellulaire*, si nécessaire, il est possible facilement et efficacement d'améliorer manuellement l'extraction automatique de l'agencement de cellules comme dans 2D "Map-Sketching" [BG89, GHPT89] (Section C.5). Notre structure de données permet à l'utilisateur

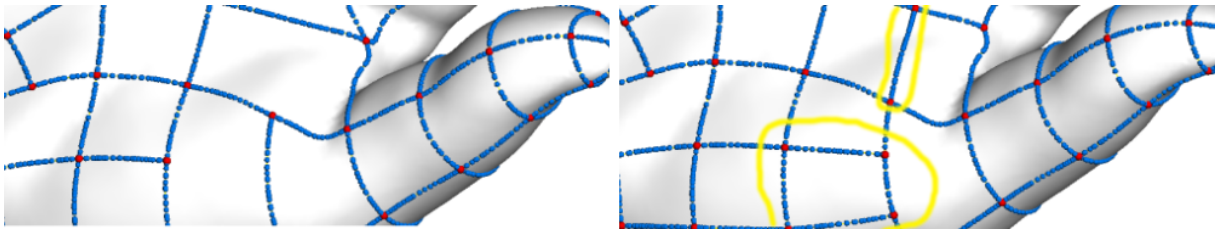


FIG. 4.6: *Édition manuelle de l'agencement de cellules en ajoutant un geodesique (cerclé). Noter que toutes intersections sont tenues à jour.*

d'utiliser les trois opérations suivantes (Figure 4.6) :

- renforcement d'arête : une arête de maillage de contrôle est remplacée par une géodésique.
- insertion d'arête : une nouvelle arête est créée (une géodésique) entre deux points entre deux points choisis de la surface. Toutes les intersections sont maintenues à jour,
- suppression d'arête.

4.4 Conclusion

En ce chapitre, nous avons expliqué comment extraire un agencement de cellules à partir de la paramétrisation calculée dans l'étape C. Pour ce faire efficacement, nous avons proposé une nouvelle structure de données de plongement du complexe cellulaire. Grâce à cette nouvelle structure de données, on peut également améliorer manuellement l'extraction automatique de l'agencement de cellules.

Cette partie de l'algorithme est expliquée en détail dans les annexes F.3.3 et C. Le lecteur pourra aussi consulter nos article publiés à *ACM Transactions on Graphics* [RLL$^+$] et à *Proceedings of IEEE International Conference on Shape Modeling and Applications '05* [LLP05].

Chapitre 5

Étapes E, F, G : Ajustement

5.1 Introduction

L'agencement des cellules créé par les étapes précédentes correspond à un maillage de contrôle. Nous attachons alors ce maillage de contrôle à une surface spline. Dans cette thèse, comme exemple, nous avons choisi d'utiliser la représentation en T-spline proposée par [SZBN03]. Parmi toutes les représentations possibles, nous avons choisi celle-ci car elle est compatible avec les splines 2D classiques utilisés en CAGD/CAM.

5.1.1 Surface T-spline

Un maillage de contrôle d'une surface T-spline est un maillage quadrilatéral pour laquelle des jonctions en T sont permises (Figure 5.1). En d'autres mots, les T-splines sont une généralisation des surfaces NURBS avec des T-jonctions. Elles correspondent à une représentation *sous forme de points*, où le domaine d'influence D_i d'une fonction de base B_i (à la différence des surfaces NURBS) n'est pas défini pas les vecteurs-noeuds globaux $\mathbf{s}$ et $\mathbf{t}$ des deux courbes NURBS (en supposant une grille régulière). Cette fois, l'influence d'une fonction de base est définie par sa propre paire de vecteurs-noeuds $\mathbf{s_i} = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$ et $\mathbf{t_i} = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$. Cette paire de vecteurs-noeuds est déduite du maillage de contrôle de la T-spline, et est appelée T-maillage. Dans un T-Maillage, chaque arête est associé à un intervalle-noeud, qui doit satisfaire deux conditions de validité :

Règle 1 : La somme des intervalles noeuds sur les arêtes opposées de chaque face doit être égale,

Règle 2 : Si deux T-jonctions sur les arêtes opposées d'une face peuvent être connectées, en respectant la règle précédente, alors cette arête doit être incluse dans le T-maillage.

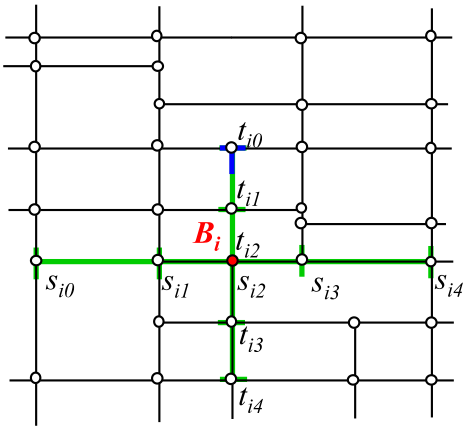


FIG. 5.1: *La pré-image d'un T-Maillage. La croix verte indique le domaine d'influence de la fonction de base $B_i(s,t)$. Le T bleuté montre une T-jonction sur le T-Maillage.*

5.2 Étape E : Satisfaction des contraintes du maillage de contrôle T-spline

En conséquence, l'agencement de cellules obtenu dans l'étape précédente (étape D) doit être retravaillé, pour satisfaire les règles 1 et 2. En particulier, nous devons transformés les cellules à N-côtés de l'agencement de cellules.

5.2.1 Conversion d'une cellule à N-côtés en quadrilatères

Nous subdivisons chaque cellule à N-côtés en cellules quadrilatères. Trois différents cas peuvent apparaître (Figure 5.2) :

- A : Normalement, quand N est impaire, il est possible d'insérer un nouveau sommet et de le connecter à l'aide de jonctions en T à chaque arête du N-gone. Cela crée N nouveaux quadrilatères, N jonctions en T et un sommet extraordinaire,
- B : Dans ce cas, où des sommets de valence 3 apparaissent dans le N-gone original, plusieurs sommets extraordinaires peuvent apparaître,
- C : Lorsque N est pair, pour éviter de créer des jonctions en T inutiles, un nouveau sommet est inséré et connecté à chaque paire de sommets de la cellule. Cela crée N/2 nouveaux quadrilatères et un sommet extraordinaire.

Après ce traitement, l'agencement de cellules définit un maillage de contrôle qui satisfait les conditions de validité d'une T-spline (i.e un T-maillage valide). Dans la suite du texte, les sommets du maillage de contrôle seront notés P_i.

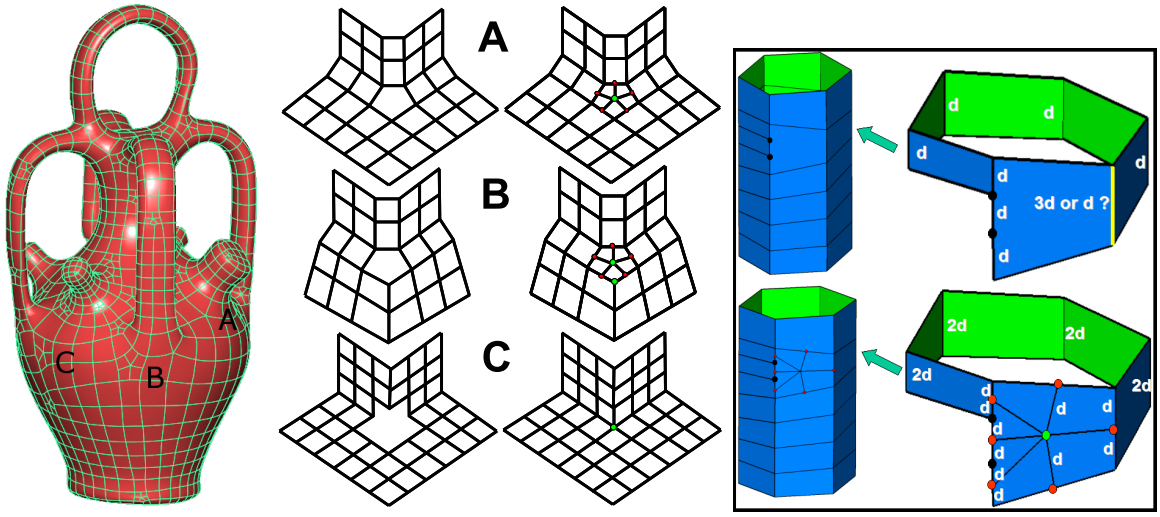


FIG. 5.2: *Gauche : le polygone original à N-côtés avec des sommets extraordinaires. Droite : un intervalle noeud consistant formé en créant de nouveaux sommets extraordinaires.*

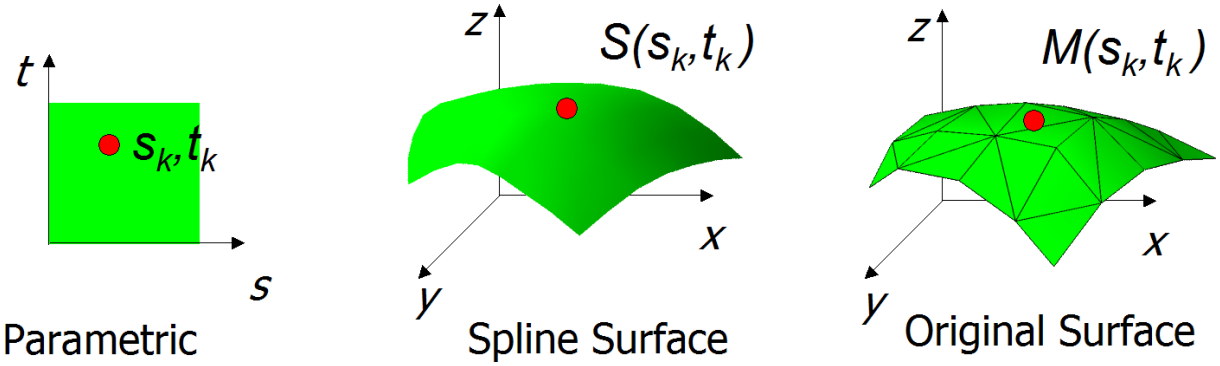


FIG. 5.3: *Les points sur la spline et les surfaces originales identifiés à travers la paramétrisation.*

5.3 Étape F : Réajustement de la T-spline surface en fonction de la triangulation originale

Nous souhaitons maintenant fixer les degrés de liberté (les positions des points de contrôles P_i) de sorte à ce que l'erreur commise entre la triangulation originale et la surface T-spline soit minimisée.

Rappelons tout d'abord la définition mathématique d'une T-spline avant de voir comment procéder à cette réajustement.

Étant donné un T-maillage valide, une surface T-spline est définie comme suit :

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

où,

- $D = D_1 \cap D_2 \cap \cdots \cap D_n$,
- P_i est le $i^{\text{ème}}$ point de contrôle (x_i, y_i, z_i),
- la fonction de combinaison $B_i(s,t) = B[\mathbf{s}_i](s)B[\mathbf{t}_i](t)$,
- $B[\mathbf{s}_i](s)$ est la fonction de base B-spline cubique associée avec le vecteur-noeud $\mathbf{s}_i$
- et $B[\mathbf{t}_i](t)$ la fonction de base B-spline cubique associée avec le vecteur-noeud $\mathbf{t}_i$.

Les deux vecteurs-noeuds $\mathbf{s}_i$ et $\mathbf{t}_i$ de la fonction de base $B_i(s,t)$ sont déduits de l'information du noeud du T-maillage comme suit (voir figure 5.1) : soient (s_{i2}, t_{i2}) les coordonnées de P_i. Considérons une droite dans l'espace paramètre $R(\alpha) = (s_{i2} + \alpha, t_{i2})$. Alors s_{i3} et s_{i4} sont les s cordonnées des deux premières s-arêtes intersectés par la droite lorsque α est croissant. Une s-arrête est une ligne verticale correspond à une valeur s constante. Les valeurs s_{i0}, s_{i1}, t_i sont trouvées de la même façon.

Malheureusement, la définition précédente n'est pas applicables lorsque des sommets extraordinaires sont présents sur le maillage de contrôle T-spline. Les sommets extraordinaires sont les sommets du T-maillage qui ne sont pas incidents à quatre arêtes. Ils créent des trous à N-côtés sur la surface spline. Pour résoudre ce problème, nous avons adopté la solution suggérée dans le papier original sur les T-splines [SZBN03] qui consiste à remplir les trous à N-côtés en utilisant des T-NURCCs. L'idée de basse des T-NURCCs est de successivement diviser le maillage de contrôle T-spline autour du sommet extraordinaire, et ainsi de réduire la taille du trou à N-côté.

Lorsque la surface T-spline $\mathbf{S}(s,t)$ est défini partout, nous pouvons la réajuster vers la triangulation originale. Pour mesurer et définir une métrique d'erreur, il est nécessaire d'utiliser une fonction de correspondance entre les points des deux surfaces. Nous obtenons une telle fonction en retrouvant des paramétrisations par morceaux à partir de la paramétrisation globale périodique.

5.3.1 Retrouver les paramétrisations par morceaux

L'étape suivante de l'algorithme (ajustement) a besoin d'une paramétrisation sur la surface initiale. Les paramétrisations par morceaux sont retrouvées une par une de deux façons différentes en fonction de la singularité ou non d'un morceau :

– Pour un morceau ne contenant pas de primitives singulières, la paramétrisation peut être retrouvant en assemblant les triangles dans un espace 2D en utilisant un algorithme glouton (voir par exemple [SdS01]).
– Un morceau contenant des primitives singulières ou un morceau modifié (soit par l'utilisateur, ou par une décomposition d'une cellule à N-côté lors de l'étape E), est reparamétrisé, en utilisant la méthode des coordonnées moyennes [Flo03]. La continuité C^1 sur le bord est assuré en appliquant une méthode de relaxation locale comme dans [KLS03, SPPH04].

5.3.2 Ajustement global L^2

Maintenant que nous avons à la fois la paramétrisation de la surface T-spline $\mathbf{S}(s,t)$ et la paramétrisation de surface triangulée $\mathbf{M}(s,t)$, nous minimisons l'énergie fonctionnelle suivante, en utilisant une méthode classique d'ajustement régulier (comme par exemple dans [Gre94]) :

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

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

Le terme d'ajustement E_{fit} est utilisé pour minimiser l'erreur entre la triangulation originale et la surface T-spline. Il est calculé en identifiant les points des deux surfaces à l'aide de deux paramétrisations $\mathbf{S}(s,t)$ et $\mathbf{M}(s,t)$ (figure 5.3). L'énergie de plaques minces "thin-plate energy" est utilisée pour éviter les oscillations sur la surface spline, une méthode souvent utilisée en modélisation géométrique.

Il faut noter que par construction, le maillage de contrôle et le vecteur-noeud associé définissent un T-spline standard (ou semi-standard)[SZBN03]. Les dénominateurs apparaissant donc identiquement égaux à 1, et nous pouvons nous limiter aux numérateurs :

$$\mathbf{S}(s,t) = \sum_{i=1}^{n} \mathbf{P}_i \mathbf{B}_i(s,t)$$

Chaque coordonnée x, y, et z peut être ajustée indépendamment. Pour la coordonnée x, le terme d'ajustement est donné par :

$$E_{fit}^x = \sum_{k=1}^{m} \left(\sum_{i=1}^{n} X_i \mathbf{B}_i(s_k,t_k) - x_k \right)^2$$

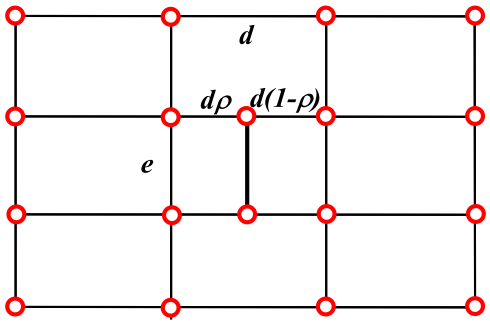


FIG. 5.4: *Une opération élémentaire de raffinement locale de T-spline.*

où X_i (resp. Y_i, Z_i) indique la coordonnée du point de contrôle $\mathbf{P}_i$. La valeur X_i qui minimise l'erreur ci-dessus est aussi solution du système linéaire $A^tAX = A^tb$, où les coefficients de la matrice A (matrice $m \times n$) sont données par $a_{k,i} = B_i(s_k,t_k)$ et la partie droite de l'égalité par $b_k = x_k$. Le vecteur inconnu X correspond à toutes les coordonnées x des points de contrôles. En ajoutant le terme approprié E_{fair}, le système linéaire devient :

$$\left(A^tA + \sigma(A^t_{ss}A_{ss} + 2A^t_{st}A_{st} + A^t_{tt}A_{tt})\right)X = A^tb$$

où les coefficients $(.{}_{k,i})$ des matrices $(m \times n)$ A_{ss}, A_{st}, A_{tt} sont les dérivées du second ordre $B_{iss}(s_k,t_k)$, $B_{ist}(s_k,t_k)$ et $B_{itt}(s_k,t_k)$ des fonctions de bases respectives B_i.

Cette partie de l'algorithme est expliquée plus en détail dans les annexes F.3.4 et G.3. Elle a l'objet d'un article publié à *Proceedings of EG/ACM Symposium on Geometrie Processing '06* [LRL06].

5.4 Étape G : Ajustement L^∞ en utilisant une méthode d'ajustement adaptative locale

L'ajustement global L^2 opère sur une nombre fixe de points de contrôles. Un nombre fixé de degrés de liberté n'est donc parfois pas suffisant pour reconstruire une approximation fidèle de la surface originale ; d'autres degrés de liberté doivent donc être ajoutés. Généralement, cette opération est effectué en raffinant globalement le maillage de contrôle, ce qui rajoute des points de contrôle supplémentaires dans des régions ayant déjà une bonne approximation. Dans notre cas assez spécifique où des jonctions en T sont autorisées, des nouveaux points de contrôle peuvent être ajoutés dans les régions où l'erreur est la plus élevée. De plus, grâce au support local des T-splines, il n'est pas nécessaire de refaire un réajustement global L^2 chaque fois qu'un nouveau point de contrôle est ajouté ; seul un petit système linéaire doit alors être

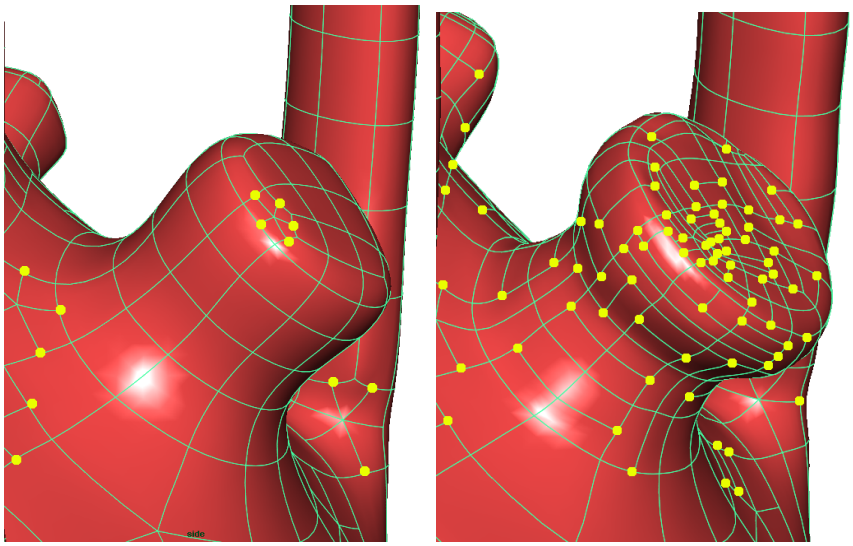


FIG. 5.5: *Le raffinement adaptatif local subdivise quelques faces pour mieux capturer une géométrie complexe.*

résolu, système prenant seulement en compte les éléments du maillage de contrôle affectés par l'opération de raffinement locale. Une métrique L^∞ est définie comme suit :

$$L^\infty(\mathbf{S},\mathbf{M}) = \max_S \|(\mathbf{S}(s,t) - \mathbf{M}(s,t))\|^2$$

où $\mathbf{M}(s,t)$ correspond à une paramétrisation de la surface originale. Cette métrique est évaluée sur une grille régulière de l'espace paramétrique placée sur chaque face.

Nous appliquons itérativement la procédure de raffinement (figure 5.4) décrite ci-dessous à la face où l'erreur L^∞ commise est la plus forte, jusqu'à ce que l'erreur devienne inférieure à un seuil fixé par l'utilisateur (confère figure 5.5).

La possibilité de raffiner localement d'une T-spline est l'une de ses propriétés intéressantes. Elle est aussi appelée insertion d'un noeud local (voir [SCF+04] pour une explication plus détaillée). De nouveaux points de contrôle sont insérés dans le T- maillage sans changer la géométrie de la surface T-spline originelle. L'algorithme conserve la validité des contraintes T-spline en insérant successivement de nouveaux points comme suit :

1. Insertion de nouveau(x) point(s) de contrôle dans un T-maillage,
2. Si un noeud manque pour une quelconque fonction de base pour satisfaire la Règle 1 dans le T-maillage actuel, insertion des noeuds nécessaires pour mettre à jour cette fonction de base.
3. Si une quelconque fonction de base possède un noeud qui ne correspond pas à la Règle 1, ajouter un point de contrôle approprié dans le T-Maillage.

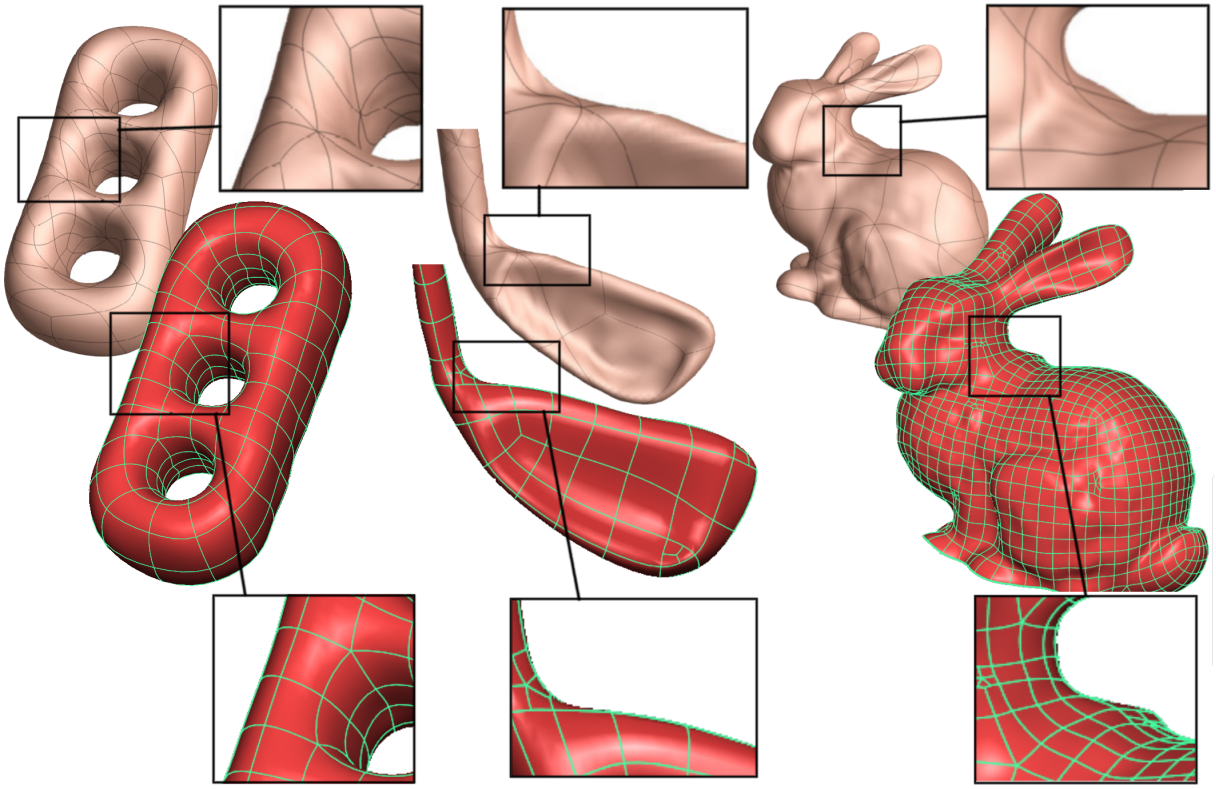


FIG. 5.6: *Comparaison entre les resultats de [EH96] et les nôtres : noter que la symétrie et l'anisotropie sont respectes par notre approche.*

4. Répéter les étapes 2 et 3 tant qu'une insertion apparaît.

La face avec la plus grande erreur L^∞ est subdivisée en deux rectangles en adaptant les intervalles noeuds (i.e. mis à 0.5 pour les arêtes subdivisées).

L'algorithme de raffinement local préserve la géométrie de la surface T-spline originale. Cependant, nous souhaitons utiliser ces nouveaux degrés de liberté pour mieux approximer la surface maillée originale. Un ajustement local est donc effectué après le raffinement local. Mais comme les fonctions T-splines ont une influence locale, cela nécessite seulement la résolution d'un petit système linéaire.

Cette partie de l'algorithme est expliquée plus en détail dans l'annexe G.4 et a fait partie d'un article publié dans *Proceedings of EG/ACM Symposium on Geometry Processing '06* [LRL06].

5.5 Resultats

Figure 5.6 compare des résultats d'Eck et Hoppe avec le nôtre (noter que les images d'Eck et Hoppe ont reproduite ici montrer seulement les bords des morceaux rectangulaires, chaque

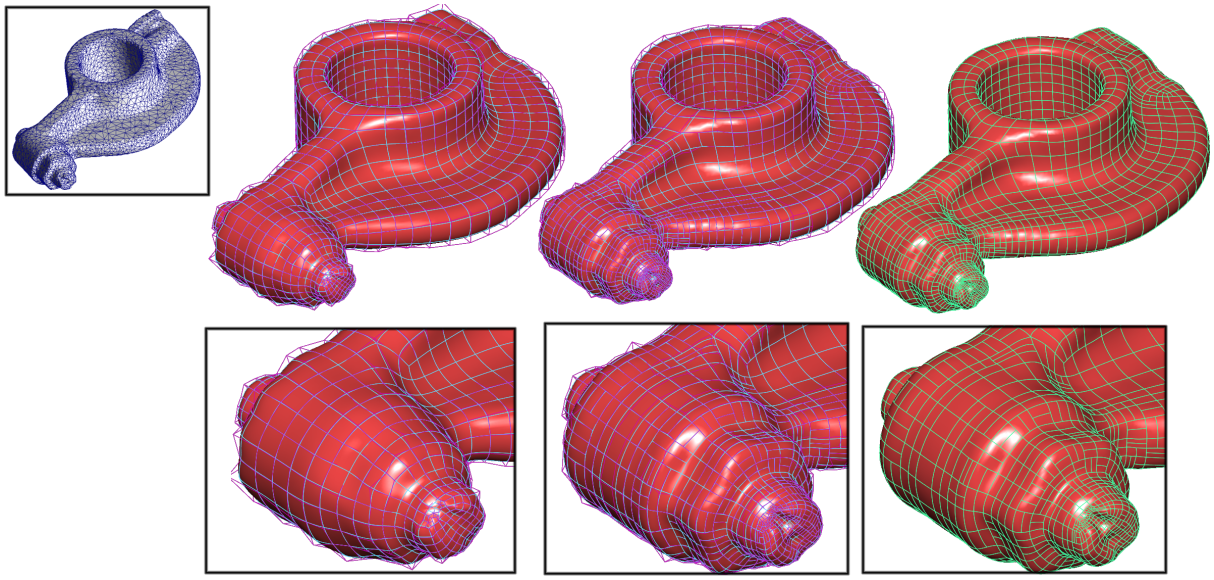


FIG. 5.7: *Convertir un maillage scanné en une surface de T-spline. De gauche à droite : L^2 ajustement, L^∞ ajustement avec refinement local (avec et sans le maillage de contrôle).*

morceau a une rangée de points de contrôle 4x4, donc les maillages de contrôle sont comparables). Comme peut être vu, notre méthode respecte mieux les symétries (voir par exemple le tore de trois-trous) et l'anisotropie des objets. Par conséquent, les surfaces résultantes n'ont pas de rides (voir les plans rapprochés). Nous montrons dans les figures 5.8 et 5.7 notre méthode appliquée aux surfaces de divers topologies et geometries. Pour tous ces exemples, moins que 15 bords ont été ajoutés par l'utilisateur. Noter que l'exemple de Rocker (Figure 5.7) est topologiquement équivalent à un tore. Par conséquent, il serait possible de créer un maillage de contrôle sans aucune singularité. Cependant, nous pensons que le maillage de contrôle construit par notre méthode est plus normal, car il prend en compte la géométrie de l'objet.

5.6 Conclusion

En ce chapitre, nous avons présenté comment ajuster une surface de T-spline à la surface triangulée originale. Des résultats, on peut voir que grâce aux maillages de contrôle anisotropes et les paramétrisations de basse déformation des surfaces originales, les surfaces de spline se sont produites après l'ajustement ont beaucoup moins de rides par rapport aux méthodes de reconstruction de surface en spline qui emploient des paramétrisations locales et des maillages de contrôle non-anisotropie-adaptées. Pour réaliser l'ajustement de précision de L^∞, nous avons également expliqué un procédé local d'amélioration adaptative automatique de T-Maillage pour

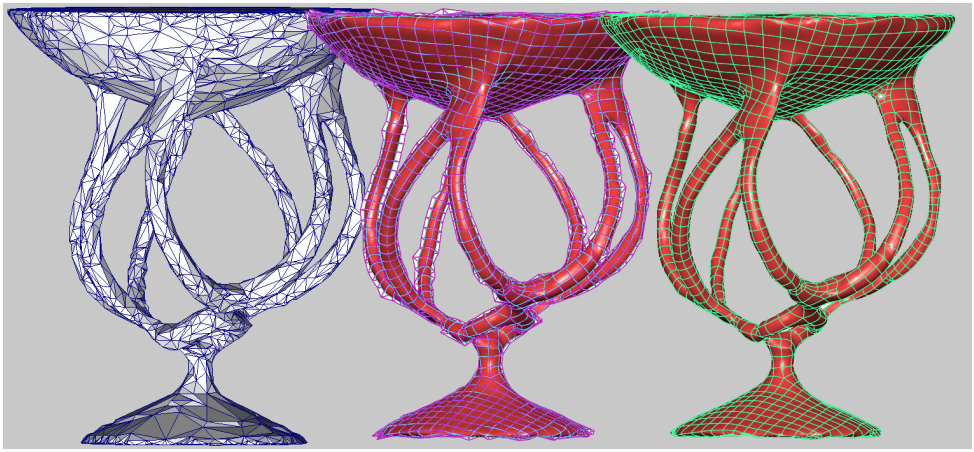


FIG. 5.8: *Notre méthode appliquée à un objet de genre élevé. De gauche à droite : maillage initiale, maillage de contrôle ajusté et surface. Cet exemple montre aussi la robustesse de notre méthode aux maillages de mauvaises qualités.*

ajouter les points de contrôle supplémentaires dans la T-Maillage pour augmenter les degrés de liberté. Utilisant ces outils automatiques et manuels, un modèle complexe peut être converti en moins de 15 minutes et être chargé dans le logiciel industriel. C'est plus rapide de manière significative que les solutions entièrement manuelles existant dans les logiciels commerciaux.

Chapitre 6

Conclusion

Les représentations paramétriques des surfaces et plus précisément les surfaces splines (*tensor-product splines*), sont les représentations les plus utilisées dans les domaines de la CAGD (*Computer-Aided Geometric Design*) et CAM (*Computer-Aided Manufacturing*). Dans ces représentations, la géométrie du maillage est définie par un maillage de contrôle quadrilatéral.

En CAGD/CAM, pour obtenir une représentation spline d'un objet réel, il faut commencer par définir un bon maillage de contrôle. Dans cette thèse, nous avons proposé deux critères pour évaluer la qualité des bons maillages de contrôle :

– les arêtes doivent se croiser perpendiculairement, et
– les arêtes doivent être alignées avec les directions de courbure principales de la surface.

Cependant, obtenir des maillages de contrôle satisfaisant ces critères est un problème *non-trivial*. Nous avons donc proposé un nouvel algorithme automatique, appelé *p*aramétrisation globale périodique (voir section F.2, résultat aussi publié dans [RLL$^+$]), pour convertir les données brutes extraites par scanner 3D (maillages triangulaires) en un bon maillage de contrôle. Les maillages de contrôle ainsi obtenus non seulement respectent les deux critères ci-dessus mais permettent aussi d'obtenir de meilleurs résultats de remaillage que les méthodes précédentes.

L'idée principale de notre approche est de trouver une paramétrisation qui ait "une signification géométrique" en utilisant une paire de champs de directions qui suivent l'anisotropie de la surface. Puis, en extrayant des lignes d'iso-valeurs de cette paramétrisation, un maillage de contrôle initial est obtenu. Cette extraction se fait en utilisant notre structure de donnée de

plongement du complexe cellulaire (voir chapitre C, résultat aussi publié dans [LLP05]).

Afin d'obtenir des champs de directions anisotropes et orthogonaux, nous avons introduit deux nouvelles méthodes :

- relaxation globale d'une estimation des directions principales de courbure (voir section D.4, résultat aussi publié dans [RLL$^+$]) ;
- construction des champs de direction avec explicite contrôle des singularités (voir chapitre E, résultat aussi publié dans [RVLL06, LV$^+$06]).

Avec le maillage de contrôle, nous avons expliqué comment construire une surface de T-spline en approximant la surface initiale. Nous employons l'ajustement L^∞ en utilisant une méthode adaptative locale (voir chapitre G, résultat aussi publié dans [LRL06]). Parmi toutes les représentations paramétriques, nous avons choisi la représentation en T-spline puisqu'elle permet un raffinement local et qu'elle est compatible avec les *tensor-product splines* standards utilisées en CAGD/CAM.

Nous avons montré quelques résultats de la conversion des maillages triangulaires en surfaces T-spline (voir chapitre G). Ils permettent de voir que grâce à des maillages bien adaptés à l'anisotropie et à des paramétrisations qui transforment avec peu de déformation les surfaces originales, les surfaces splines finales obtenues après l'ajustement ont beaucoup moins d'oscillations que celles résultantes des méthodes qui emploient des paramétrisations locales et des maillages de contrôle non-adaptés à l'anisotropie de la surface originale.

Dans cette thèse, nous avons proposé une méthode pour la conversion automatique et interactive de maillage en surface T-spline. Notre algorithme propose une première solution, celle-ci peut être manuellement raffinée par l'utilisateur. En utilisant ces outils automatiques et manuels, un modèle complexe peut être converti en moins de 15 minutes et est chargé dans un logiciel industriel ; ce qui est beaucoup plus rapide que les solutions entièrement manuelles existant dans les logiciels actuels de commerce.

Notre méthode d'extraction de maillage de contrôle est un grand pas en avant vers une approche qui imite la manière avec laquelle un créateur dessine un maillage de contrôle en tenant compte de la géométrie de l'objet. Quelques directions de recherche restent à explorer. Ainsi, nous pensons qu'une meilleure caractérisation mathématique des singularités dans la paramétrisation globale périodique permettrait d'éviter l'apparition de cellules à N- côtés dans

l'agencement de cellules, et par conséquent mener à une solution plus rapide et totalement automatique. Ensuite, nous pensons que pour certaines applications spécifiques (par exemple, le "reverse engineering" [Rap]), l'amélioration de notre méthode pour pouvoir créer un maillage de contrôle qui s'adapte bien aux arêtes vives du maillage triangulaire originale (par exemple, une pièce de moteur de voiture) serait très utile.

Annexe A

Mesh Representation

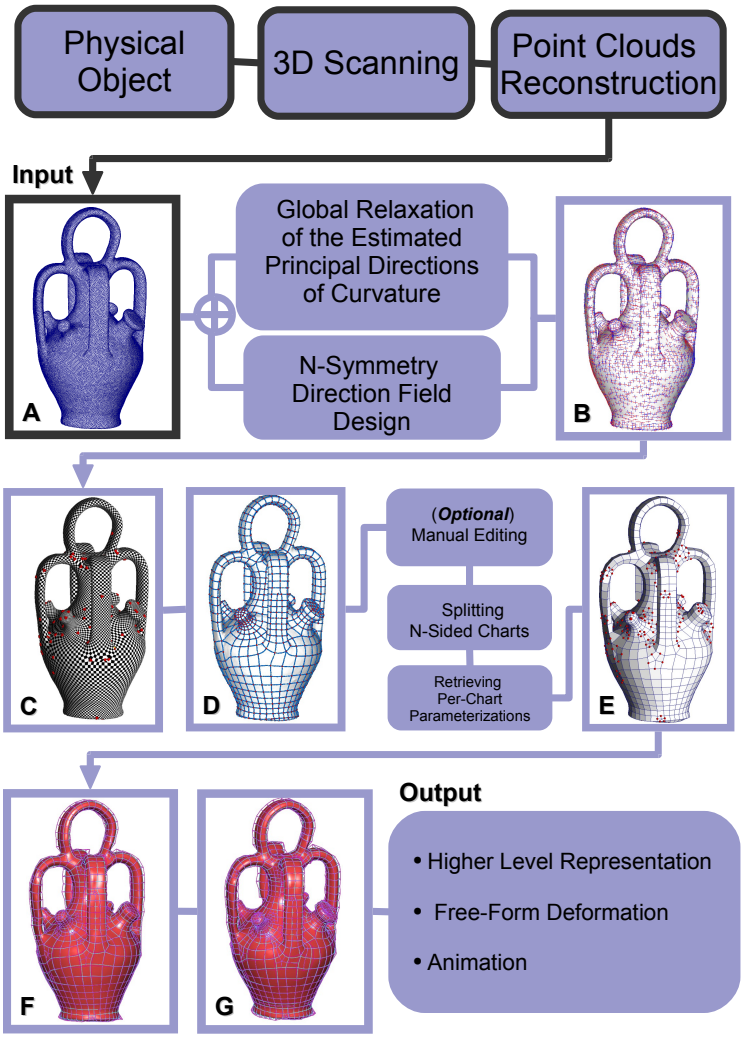


FIG. A.1: *Mesh representation.*

A.1 Introduction

Meshes are the classical representation of surfaces in $\mathbb{R}^3$ in computer graphics. Intuitively, a mesh is a surface represented by polygonal facets. It is in fact a 2D graph drawn in 3D, consisting of combinatorial entities : vertices, edges, and faces, and geometric information : the 3D position of the vertices. In the context of Computer Graphics, one often stores as well attributes, such as, normal, color and texture coordinates on the vertices (in Section A.1.1, we will revisit these ideas but in a more formal way).

Formally speaking, a polygonal mesh is a *cell complex*. When all the facets of the mesh are triangles, i.e. a triangular mesh, we have a *simplicial complex*. Since all the input meshes that we manipulate in this thesis are triangular, in the next section, we first restrict ourselves only to the definition of simplicial complexes. For a definition of cell complex, please refer to Section C.3.1 in Chapter C, where is used for our line-mesh embedding data structure for control mesh drawing on meshed surfaces. (The cells formed by intersecting lines on the surface are not necessarily triangles.)

A.1.1 Definition of Triangular Meshes

A nD mesh M is defined as a *simplicial complex* K, which is a topological space built up of points, line segments, triangles, tetrahedral, and their n-dimensional counterparts called k-simplices. These constituting simplices are called the faces of the complex. A k-simplex is the convex hull of $(k+1)$ 0-simplices. (See [DKT05, PH97] for an elementary introduction. For more mathematical background, see also *algebraic topology* [Mas91, Hat01].)

For instance, in a 2D simplicial complex, i.e. surface, the building elements are either a point (0-simplex), a line segment (1-simplex), or a triangle (2-simplex). We always assume that local orientations are given for each simplex. Namely, each element of the mesh has been given a particular orientation. The boundary operator ∂ of a k-simplex is the chain of all its $(k-1)$-faces. A simplicial complex must satisfy the following two rules (see Figure A.2) :

1. every face of each simplex in K is in K ;
2. the intersection of any two simplices in K is either empty, or a entire common face.

Topological realization

The topological realization of the simplicial complex $|K|$ is defined as follows. One first identifies the m 0-simplices with the standard basis vectors $\{\vec{e_1}, \cdots, \vec{e_m}\}$ with $\vec{e_i} \in \mathbb{R}^m$. For a

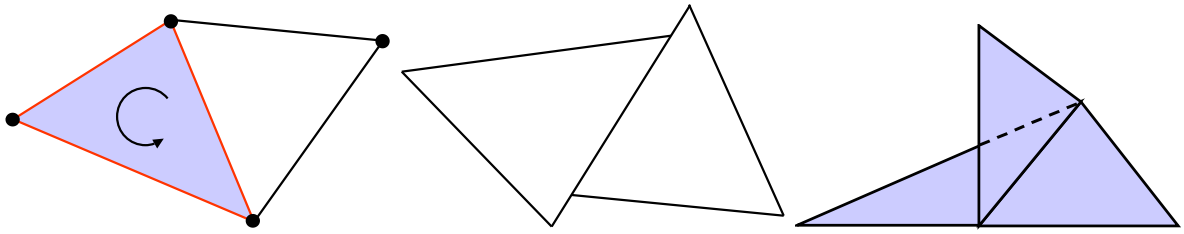


FIG. A.2: *Left : the dots are 0-simplices, the lines are 1-simplices, and the triangles are 2-simplices. The arrow inside the blue triangle indicates the orientation of the simplices. The three red lines shows the boundary of the blue triangle ; Middle : not a simplicial complex ; Right : not a 2-manifold.*

k-simplex, σ, let us denote $|\sigma|$ as the non-degenerated convex hull of its $(k+1)$ distinct 0-simplices in $\mathbb{R}^m$. The topological realization $|K|$ is defined as : $|K| = \cup_{\sigma \in K} |\sigma|$.

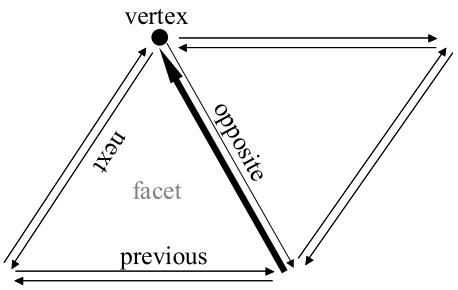
Geometric realization

For a nD simplicial complex embedded in $\mathbb{R}^q$, where $q \geq n$, we define a linear map $\pi : \mathbb{R}^m \rightarrow \mathbb{R}^q$, which sends the ith standard basis vector $\vec{e}_i \in \mathbb{R}^m$ to $\vec{g}_i \in G$, where $G = \{\vec{g_1}, \cdots, \vec{g_m}\}$, $\vec{g}_i \in \mathbb{R}^q$. The geometric realization is the image $\pi_G(|K|)$. To be an embedding, The map π must be one-to-one, i.e. there is no self-intersecting in the image. If π is an embedding, any point $p \in \pi_G(|K|)$ can be parameterized by finding its unique pre-image on $|K|$. The vector $\vec{b} \in |K|$ with $p = \pi_G(\vec{b})$ is called the barycentric coordinate vector of p (with respect to the simplicial complex K). Note that barycentric coordinate vectors are convex combinations of standard basis vectors $\vec{e}_i \in \mathbb{R}^m$ corresponding to the 0-simplices of a face of K. (Notice that the barycentric coordinates can also be used to interpolate other attributes associated to the 0-simplices, e.g. normal, color and texture coordinates.)

Therefore, for a nD simplicial complex embedded in $\mathbb{R}^q$, it is described by the pair (K, G). Note that in this thesis, we are only interested in simplicial complexes with a topological realization of 2-manifold (see Figure A.2-Right).

We use half-edge structure [Wei85] to represent simplicial complexes in our implementation. In the next section, we will describe the data structure and discuss its advantages and limitation.

A.2 Half-Edge Data Structure



In practice, when manipulating meshes, it often requires adjacency relationships between components of the mesh to be discovered. Some typical adjacency queries on a mesh are :

- which faces use this vertex ?
- which edges use this vertex ?
- which faces border this edge ?
- which edges border this face ?
- which faces are adjacent to this face ?

In order to perform such adjacency queries efficiently, we choose the half-edge or (DCEL : Doubly Connected Edge List) representation [Wei85]. A concise description of this data structure can be found in a document on the internet [McG], from where we have taken some of the materials in this section.

The half-edge data structure is a B-rep (boundary representation) which allows the adjacency queries to be performed in constant time (compared to winged-edge data structure [Bau75], where not all the queries are constant time). In addition, even though we are including adjacency information in the faces, vertices and edges, their size remains fixed (no dynamic arrays are used) as well as reasonably compact. In this data structure, instead of storing the edges of the mesh, we store half-edges (hence the name). As the name implies, a half-edge is a half of an edge and is constructed by splitting an edge down its length. The two half-edges that make up an edge are called a pair. Half-edges are directed and the two edges of a pair have opposite directions. The half-edges that border a face form a circular linked list (conventionally oriented counter-clockwise). The data structure is summarized in the following and illustrated in the above figure on the right.

- What a half-edge stores ?
  - vertex at the end of the half-edge
  - oppositely oriented adjacent half-edge
  - facet the half-edge borders
  - next half-edge around the face
  - previous half-edge around the face
- What a vertex stores ?
  - the 3D position
  - one of the half-edges emanating from the vertex

- What a facet stores ?
  - one of the half-edges bordering the face

Limitation

Half-edge data structure is only limited to representing orientable 2-manifolds. This is not a problem for us since we are only interested in oriented 2-manifolds. More generalized data structures, such as, Quad-Edge data structure [GS85] can be used for non-orientable 2-manifolds.

A.3 Pros and Cons

Although a facet can be polygonal, triangular meshes are the mostly used type in Geometry Processing and Computer Graphics.

- Triangular meshes are ubiquitous due to two reasons :
  - It is a straightforward representation. Triangles are the most basic unity that graphics cards recognize. Therefore, parametric surfaces and CSG representations are usually converted into triangular meshes for rendering.
  - Visually effective rendering algorithms exist to achieve smooth lighting on triangulated surfaces. One of the classical techniques is the Gouraud shading [Gou71a].
- Despite the simplicity, there are two drawbacks associated to meshes :
  - It is hard to strike a balance between the accuracy and efficiency. Specifically, when a low-resolution mesh is used, the error between the model and the object is high but the rendering is much faster. On the other hand, when a high-resolution mesh is used, more geometric details can be captured but it is inefficient in term of memory and rendering. In the literature, many works [Hop96, GH97] have been dedicated to help find the best resolution by decimating from a high-resolution mesh.
  - When being used in the modeling of objects, ensuring continuity (smoothness) when modifying the curved regions of the surface means many points have to be positioned accurately, which is often tedious.

A.4 Conclusion

In this chapter, we have explained what mesh representation is both in an intuitively and formally way. We have also discussed the Half-Edge data structure, and why we have chosen

it in our mesh implementation. These introductions give the reader a general idea of the input objects that we manipulate, i.e. triangular meshes are a discrete description of surfaces with samples G, and a connectivity K. The (K,G) pair gives a C^0 approximation of the original surface.

Annexe B

Parametric Representations

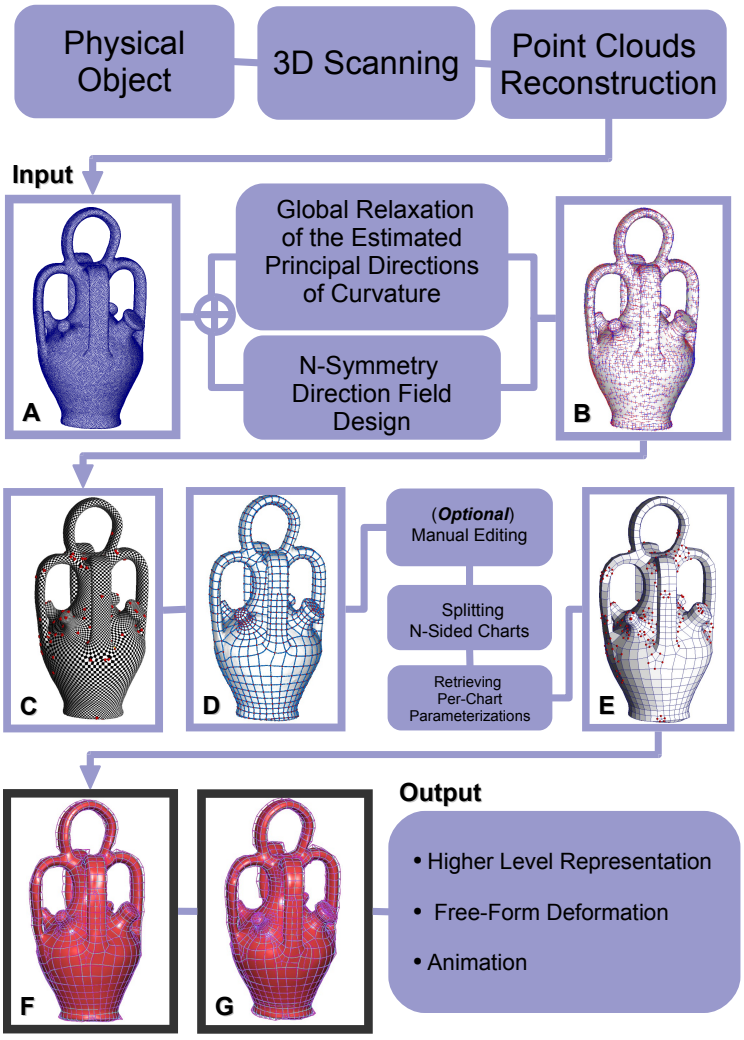


FIG. B.1: *T-spline surface representation.*

B.1 Introduction

In the previous chapter, we have introduced the mesh representation. Objects from 3D scanning are often output as meshes. Although this representation of 3D scanned objects is straightforward, the large amount of sampled point of a scanned physical object makes it very inefficient in terms of memory and disk space for interactive editing. Furthermore, ensuring continuity (smoothness) when modifying the curved regions of the surface means many points have to be positioned accurately, which is tedious. These two problems can be overcome by representing the 3D object in a parametric representation. Parametric representations of surfaces, and more specifically tensor-product splines, [Far02] are widely used in CAGD (Computer-Aided Geometric Design) and CAM (Computer-Aided Manufacturing). In these representations, the geometry of a surface is usually defined by a quadrilateral control mesh. In the context of free-form deformation, through the manipulation of control points of the control mesh, one can smoothly modify the shape of the underlying spline surface, which is generally C^2. Moreover, in the context of FEM (Finite Element Method) and automotive design, with the analytic form, one can precisely calculate the normal and all the derivatives of the surface at arbitrary parameter values. For instance, in the context of automotive design, since that it has a dramatic impact on the highlights on the car body, higher order continuity is of great importance (e.g. Class-A surface design [ICE] where the surfaces should have a higher-order continuity to achieve an aesthetically pleasing effect).

In this thesis, we have chosen T-spline as the target spline representation. T-spline surfaces are a generalization of NURBS surfaces. In addition, T-junctions are allowed in the control mesh. There are several ideas in this new representation that the reader may find unfamiliar. For instances, point-based spline and knot vector of a basis function (not of a curve). In the remainder of this chapter, through the exposition starting from parametric curves, then to parametric surfaces, and finally to T-spline surfaces, we hope that the several basic ideas of T-spline surfaces could become clear to the reader.

Some materials of the exposition are taken from Alan Watt's book [Wat93], where a very solid elementary introduction to parametric curves and surfaces is given. Interested readers can also refer to [Far02] for more advanced subjects in parametric curves and surfaces. Note that in the following, if not explicit, the curves and surfaces that we describe are all assumed to be of degree-3, i.e., cubic.

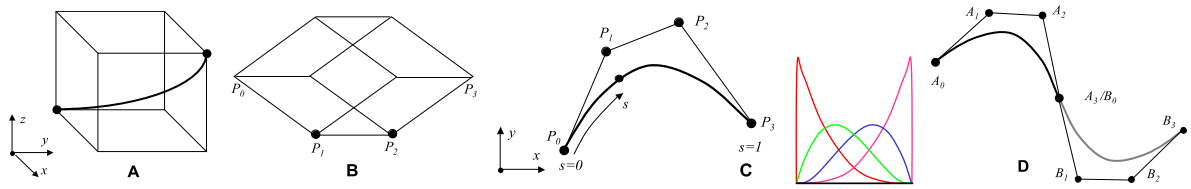


FIG. B.2: *Bezier curves.*

B.2 Parametric Curves

B.2.1 Bézier Curves

We will first introduce the notions of Bézier curves due to their simplicity and intuitive underlying mathematics. Cubic Bézier curves can be understood as a space curve being contained in a cube, which when distorted into a parallelepiped distorted the curve (Figure B.2). The curve is attached to the parallelepiped using the following three conditions :

- The start and end points of the curve are located at opposite vertices of the parallelepiped.
- As its start point the curve is tangential to $0x$.
- As its end point the curve is tangential to $0z$.

In this way, the parallelepiped, and thus the curve, can be completely defined by four control points P_0, P_1, P_2 and P_3, which are vertices of the parallelepiped as shown in Figure B.2-B. Figure B.2-C shows the curve projected into the 2D space of the diagram, which is the common way to depict a Bézier curve and its control points when drawing in 2D.

In the following, one will see how the above three conditions can be written mathematically. The parametric domain of a Bézier curve is often defined as $0 \leq s \leq 1$, where s is the parameter. Each point on the curve $Q(s)$ is obtained by blending the control points P_i. Namely, the weight of each control point is determined by the corresponding Bernstein cubic polynomial (basis or blending function) $B_i(s)$ (Figure B.2-C). Now, the curve is given by :

$$Q(s) = \sum_{i=0}^{3} P_i B_i(s) \tag{B.1}$$

$B_0(s) = (1-s)^3$,
$B_1(s) = 3s(1-s)^2$,
$B_2(s) = 3s^2(1-s)$, and
$B_3(s) = (s)^3$.

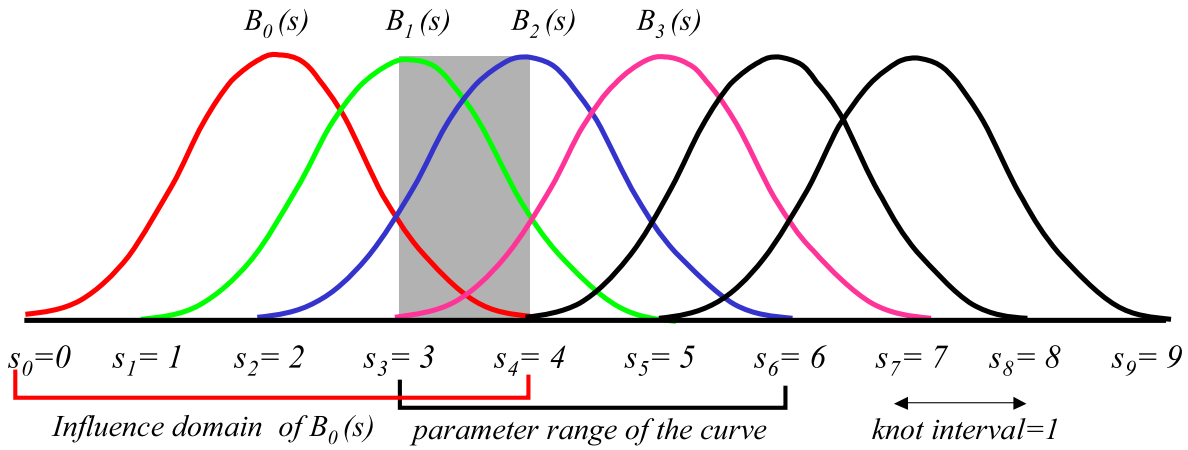


FIG. B.3: *A uniform knot vector [0,1,2,3,4,5,6,7,8] and its corresponding B-spline basis functions.*

One can verify that the above expression of Bézier curves satisfies the three conditions that the curve is attached to the parallelepiped. First,

$$Q(0) = P_0 \qquad \text{and} \qquad Q(1) = P_3$$

since $s = 0$ (*resp.* $s = 1$), only B_0 (*resp.* B_3) is non-zero. Hence, the first condition.

Then, it can also be derived that

$$Q_s(0) = 3(P_1 - P_0) \qquad \text{and} \qquad Q_s(1) = 3(P_2 - P_3)$$

where Q_s is the tangent vector (or first derivative) to the curve. Hence, the second and third conditions. Due to these tangent vector conditions at the ends of curves, special attention must be paid to maintain continuity between two Bézier curves A and B. The conditions are $A_0 = B_3$ and $\overrightarrow{A_2A_3} = \lambda\overrightarrow{B_0B_1}$, where $\lambda \in \mathbb{R}\backslash 0$ (see Figures B.2-D).

B.2.2 Uniform B-spline Curves

To represent N points using Bézier form, one has to use multiple curve segments (or by using a single curve with a higher degree). Hence, one needs to take care of the inter-curve continuity constraints. By using B-spline curves, this trouble is overcome since control points can be moved in any way. The reader will see how this is achieved soon in the following.

A cubic B-spline is a complete piecewise cubic polynomial consisting of any number of curve segments. Each curve segment is defined by four control points and each control point influences four and only four curve segments. We need to first define several technical terms.

A knot vector $[s_0, \cdots, s_k]$ is a sequence of points $\in \mathbb{R}$ (the parametric space of the curve). An element of the knot vector is called knot, which indicates where a B-spline basis function is defined. The difference between two successive knot values is called knot interval (see Figure B.3). As we will see later, the knot vector is very important in determining the shape of the basis functions of a B-spline curve. Note that, scaling and translating the knot vector does not change the shape of the B-spline curve.

A B-spline curve is called uniform when its knot vector is uniform, i.e. the knot intervals are identical. Blending functions of a uniform B-spline are symmetric. Moreover, they are translates of each other from knot to knot. Therefore, the same four basis functions are used for all the curve segments of the spline (see the gray area in Figure B.3). For a cubic B-spline curve, the number of control points $n \geq 4$ and the knot vector has $n+4$ knots. The curve is made up of $n-3$ curve segments. For the ith segment (starting from 3), $[s_i, s_{i+1}]$, by renormalizing the segment to $[0,1]$, one can express it in a similar formulation as in the Bézier case as follows :

$$Q_i(s) = \sum_{m=0}^{3} P_{i-3+m} B_{i-3+m}(s) \tag{B.2}$$

where,
$0 \leq s \leq 1$,
$B_i(s) \quad = \frac{1}{6}s^3$,
$B_{i-1}(s) = \frac{1}{6}(-3s^3 + 3s^2 + 3s + 1)$,
$B_{i-2}(s) = \frac{1}{6}(3s^3 - 6s^2 + 4)$, and
$B_{i-3}(s) = \frac{1}{6}(1-s)^3$.

Similar to a Bézier curve segment, a B-spline curve segment is influenced by four C^2 basis functions. Recall that in a Bézier curve, the four basis functions influence only the same curve segment (see Figure B.2-C). However, in a B-spline each basis function influences exactly four curve segments (see Figure B.3). Consequently, B-spline curve segments are themselves C^2 (linear combinations of C^2 functions are still C^2) and continuity across segments is also C^2 (a basis function spans four curve segments). Hence, any control point can be moved freely while the continuity is still kept.

B.2.3 Non-Uniform B-spline Curves

When values of the knot intervals of the knot vector of a B-spline curve are not identical, the B-spline is called non-uniform. The most common form of a non-uniform B-spline is multiple knots, where some of the knot intervals are reduced to zero. This is usually used to interpolate

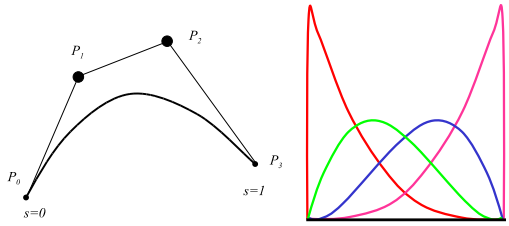


FIG. B.4: *A non-uniform B-spline curve with knot vector [0,0,0,0,1,1,1,1] and its corresponding B-spline basis functions. Note that the end points are interpolated.*

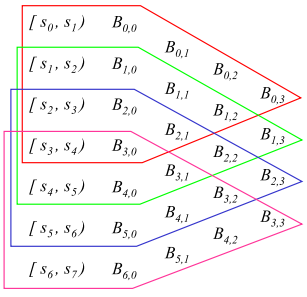
control points and inserting new control points (to represent discontinuities, e.g. a sharp turn on the curve). Figure B.4 shows such a curve that interpolates end control points by using quadruple knots.

With the non-uniform knot intervals, shapes of the basis functions of the B-spline is no longer translates of one another as in the uniform case. One can no longer reuse the same basis functions for all the curve segments. Despite being more computational expensive, B-spline basis functions can be efficiently computed using the Cox-de Boor recursive formula [Cox72, Boo72] as follows :

$$
\begin{aligned}
B_{i,0}(s) &= \begin{cases} 1, & \text{if } s_i \leq s \leq s_{i+1} \\ 0, & \text{otherwise} \end{cases} \\
B_{i,k}(s) &= \frac{s - s_i}{s_{i+k} - s_i} B_{i,k-1}(s) + \frac{s_{i+k+1} - s}{s_{i+k+1} - s_{i+1}} B_{i+1,k-1}(s)
\end{aligned}
\tag{B.3}
$$

where, k is the degree of the basis function.

Knot vector of a univariate B-spline basis function



The figure on the right clearly shows schematically how the Cox-de Boor recursive formula works and reveals clearly the fact that the cubic basis function associated to a knot s_i is defined solely by itself, two precedent knots and two following knots, namely, $[s_{i-2}, s_{i-1}, s_i, s_{i+1}, s_{i+2}]$, which is called the knot vector of the basis function. Knot vectors of basis functions of a B-spline curve are defined implicitly when being given the *global* knot vector of the curve. One can also think the other way round that the knot vector of a basis function is the *influence domain D* of the basis function in the parametric space. Using the same example of the uniform B-spline shown in Figure B.3, $B_0(s)$ associated to the knot s_2 and is defined by the knot vector $[0,1,2,3,4]$.

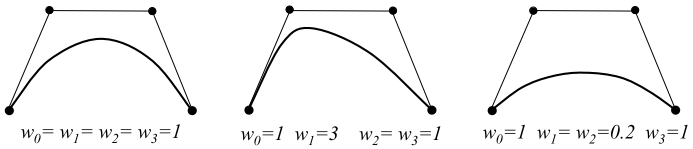


FIG. B.5: *The shape of a NURBS curve (with a knot vector [0,0,0,0,1,1,1,1]) varies using the same control points but different weights associated to control points.*

Similarly, $B_1(s)$ is defined by the knot vector $[1,2,3,4,5]$ *etc.*. Note that understanding that a basis function can be treated independently and defined by its own pair of knot vectors is very important for understanding T-spline surfaces, which are a point-based representation.

B.2.4 Non-Uniform Rational B-spline (NURBS) curves

NURBS is probably the most popular representation in CAGD/CAM. A NURBS curve is similar to a non-uniform B-spline curve except that the control points are defined in homogeneous coordinates rather than Cartesian coordinates. Consider a control point (x_i, y_i, z_i) in 3D, its coordinates in homogeneous space can be simply be written as $(x_i w_i, y_i w_i, z_i w_i, w_i)$. A NURBS curve in 3D is defined as :

$$R(s) = \frac{\sum_{i=0}^{n} w_i B_i(s) P_i}{\sum_{i=0}^{n} w_i B_i(s)} \tag{B.4}$$

where, $B_i(s)$ is the basis function computed using the Cox-de Boor formula defined in Equation B.4, w_i is called the weight of the control point since increasing the value of it gives the corresponding control point more influence. The way in which a weight affects the curve is subtly different from the movement of a control point (see Figure B.5). The term rational in the name NURBS is due to the denominator, which is basically the projection of the point from 4D back to 3D (where, $w = 1$). Figure B.6 shows a the idea of a similar projection from 3D to 2D. Being rational, one of the most important applications of NURBS is to represent conic sections.

B.3 From Curves to Surfaces

Now that we know how a parametric curve (1D) works, to generalize it to bi-parametric (or simply called parametric) surface patches (2D), one can consider the parametric surface as the Cartesian product of two parametric curves. One for the s-coordinate and the other for the t coordinates. Doing so, the geometry of the surface is mathematically described by a parameterization. Namely, being given a point (s,t) in the parametric space, through the parameterization $Q(s,t) = \{X(s,t)\ Y(s,t)\ Z(s,t)\}$, one obtains a corresponding point in 3D.

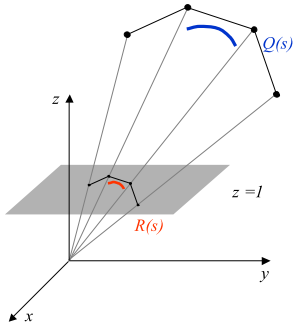


FIG. B.6: *$Q(s)$ is a B-spline curve defined in homogeneous space (3D). When the curve is projected onto the plane $z = 1$, it gives a 2D curve $R(s)$.*

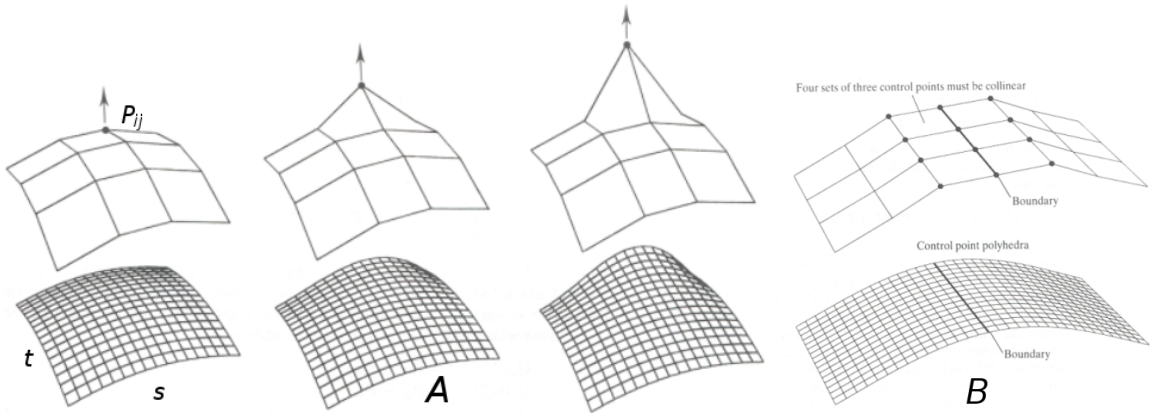


FIG. B.7: *A : Editing the surface by moving the control point. The Bézier patch surface is displayed using iso-parametric lines ; B : $4 \times 3 = 12$ control points must be constrained to ensure tangential continuity.*

B.3.1 Bézier Surface Patch

One of the simplest representations is Bézier patch, which is defined by a matrix of 4×4 control points, which forms a control mesh. The parameterization $Q(s,t)$ is completely defined by the geometry of the control mesh. Figure B.7-A shows the modification of the parametric surface through the manipulation of one of the control points. Mathematically, a Bézier patch is defined as :

$$S(s,t) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij} B_i(s) B_j(t) \tag{B.5}$$

Similar to the 1D case, maintaining tangential continuity between two Bézier patches imposes constraints on control points along the boundary between the two patches. For example, Figure B.7-B shows that boundary curves must be aligned and four sets of three control points

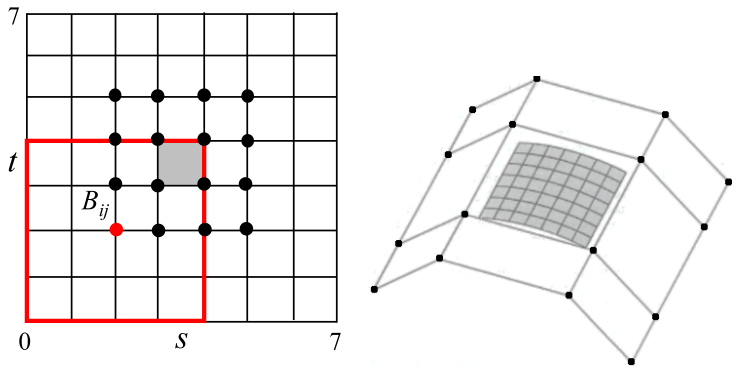


FIG. B.8: *A B-spline surface patch. The red square indicates the influence domain of the control point* B_{ij}.

must be collinear to obtain positional and tangential continuity between two adjacent patches.

B.3.2 B-spline Surfaces

Since uniform B-spline is just a subset of non-uniform B-spline, in the following, we will simply use the general term B-spline. A B-spline surface is a Cartesian product of two B-spline curves with knot vector $[s_0 \cdots s_{n_s+3}]$ and $[t_0 \cdots t_{n_t+3}]$ respectively, where n_s and n_t are the number of control points in the s and t direction respectively. Analogous to B-spline curves, a C^2 B-spline surface is made up of piecewise C^2 surface patches. Each patch is defined by 4×4 bivariate basis functions as follows :

$$S(s,t) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij} B_{ij}(s,t) \tag{B.6}$$

where, $B_{ij}(s,t) = B_i(s)B_j(t)$, $B_i(s)$ and $B_j(t)$ are the previously defined univariant B-spline basis functions.

Knot vectors of a bivariate B-spline basis function

Just as with B-spline curves, a basis function B_{ij} of a B-spline surface patch is defined solely by its two knot vectors $\mathbf{s}_{ij} = [s_{ij0}, s_{ij1}, s_{ij2}, s_{ij3}, s_{ij4}]$ and $\mathbf{t}_{ij} = [t_{ij0}, t_{ij1}, t_{ij2}, t_{ij3}, t_{ij4}]$. The two knot vectors form the influence domain $Dij = [\mathbf{s}_{ij} \times \mathbf{t}_{ij}]$ of the basis function (see Figure B.8).

B.3.3 NURBS Surfaces

Similar to B-spline curves, if the control points of a B-spline surface are defined in homogeneous space $(x_i w_i, y_i w_i, z_i w_i, w_i)$, one obtains a NURBS surface defined in 3D as follows :

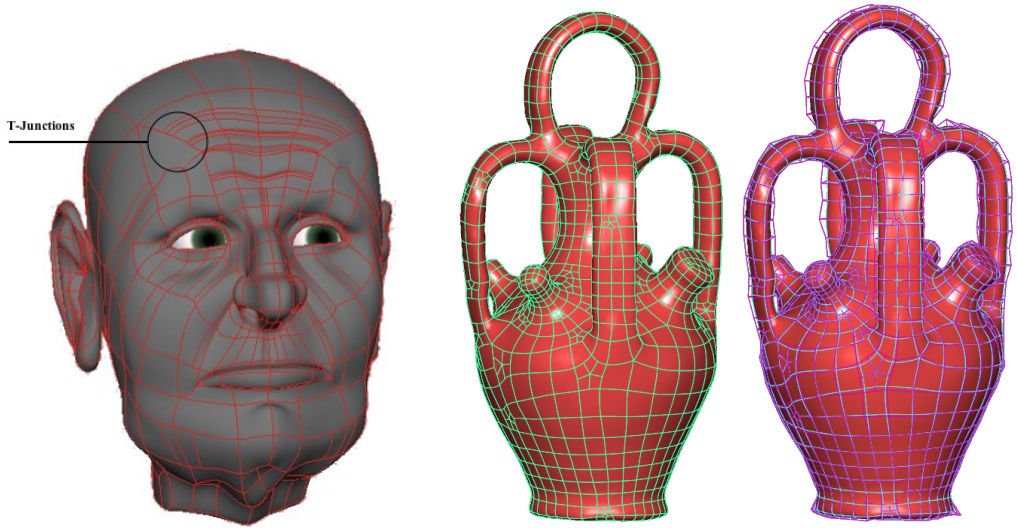


FIG. B.9: *T-spline surfaces. Left : T-junctions created by local refinement are used to add detail geometry ; Middle : N-gons are split into quadrilaterals by creating T-junctions and extraordinary vertices ; Right : T-spline surface overlaid with the T-Mesh.*

$$S(s,t) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij}B_{ij}(s,t)P_{ij}}{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij}B_{ij}(s,t)} \tag{B.7}$$

B.4 T-spline Surfaces

T-spline surfaces [SZBN03] are a generalization of NURBS surface with T-junctions. They are a *point-based* representation, where the influence domain D_i of a basis function B_i (unlike NURBS surfaces) is not implied by being given the global $\mathbf{s}$ and $\mathbf{t}$ knot vectors of the two NURBS curves (by assuming a regular grid). Instead, the influence domain of a basis function is defined by its own pair of $\mathbf{s_i} = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$ and $\mathbf{t_i} = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$ knot vectors. This pair of knot vectors are inferred from a T-spline control mesh called the T-Mesh.

B.4.1 T-Mesh Validity Conditions

A T-Mesh is a quadrilateral control mesh but do not need to be aligned on a regular grid (unlike NURBS). More specifically, a row (resp. column) of control points is allowed to terminate in a T-Mesh. The final control point in a partial row (resp. column) is called a T-junction (Figure B.10). However, since in a T-Mesh, each edge has an associated knot interval, the following two connectivity requirements need to be enforced in order to maintain know interval

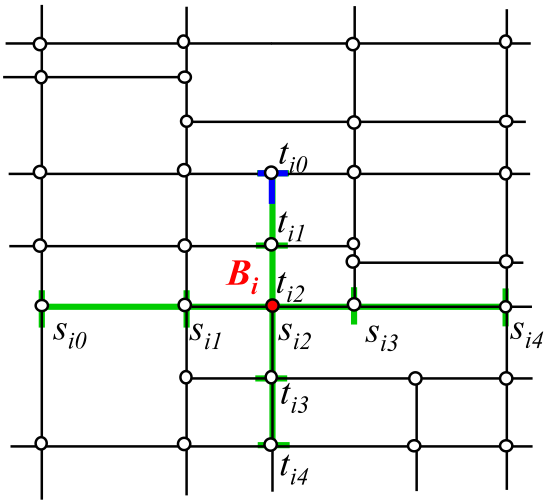


FIG. B.10: *The pre-image of a T-Mesh. The green cross indicates the influence domain of the basis function $B_i(s,t)$. The blue T shows one of the T-junctions of the T-Mesh.*

consistent over the whole T-Mesh.

Rule 1 : The sum of the knot intervals on opposing edges of any face must be equal.
Rule 2 : If two T-junctions on opposing edges of a face can be connected without violating the previous rule, that edge must be included in the T-Mesh.

B.4.2 Knot Vectors of a Bivariate T-spline Basis Function

Basis functions are defined on vertices of a T-Mesh. The two knot vectors $\mathbf{s}_i$ and $\mathbf{t}_i$ of a basis function $B_i(s,t)$ are inferred from the knot information of the T-Mesh as follows (see Figure B.10) : Let (s_{i2},t_{i2}) are the knot coordinates of P_i. Consider a ray in parameter space $R(\alpha) = (s_{i2}+\alpha,t_{i2})$. Then s_{i3} and s_{i4} are the s coordinates of the first two s-edges intersected by the ray in the positive α direction. A s-edge is a vertical line segment of constant s. The s_{i0}, s_{i1} and t_i are found similarly. If a T-Mesh is simply a rectangular grid with no T-junctions, one obtains the same set of basis functions as in the NURBS surface case.

B.4.3 T-spline Surface Equation

A T-spline surface is defined as follows :

$$\mathbf{S}(s,t) = \frac{\sum_{i=1}^{n} w_i P_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}, \quad s,t \in D \tag{B.8}$$

where,

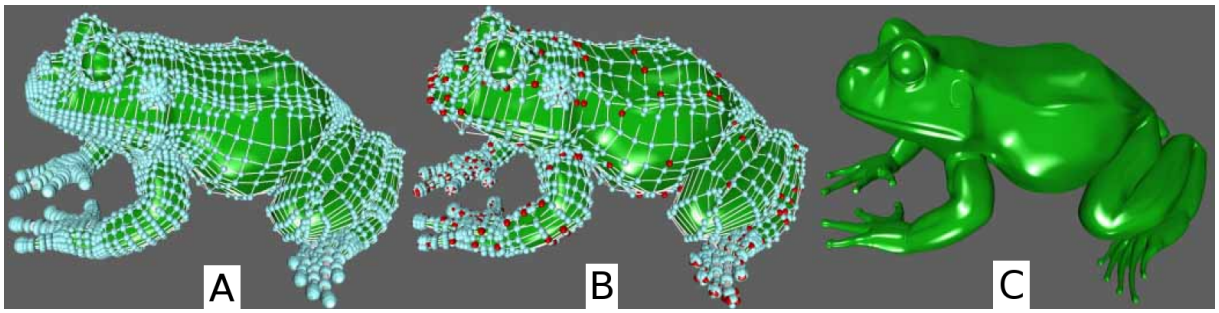


FIG. B.11: *A : a NURBS model of a frog modeled by using NURBS ; B : the NURBS model converted into T-spline representation by removing all the superfluous control points through creating T-junctions ; C : both the NURBS and T-spline model give the same geometry.*

- $D = D_1 \cap D_2 \cap \cdots \cap D_n$,
- P_i is the ith control point (x_i, y_i, z_i),
- The blending function $B_i(s,t) = B[\mathbf{s}_i](s)B[\mathbf{t}_i](t)$,
- $B[\mathbf{s}_i](s)$ is the cubic B-spline basis function associated with the knot vector $\mathbf{s}_i$, and
- $B[\mathbf{t}_i](t)$ is associated with the knot vector $\mathbf{t}_i$.

B.4.4 Local Refinement

A serious drawback associated to NURBS surfaces is that NURBS control points must lie topologically in a rectangular grid (recall that a NURBS surface is defined by two NURBS curves). Inserting extra control points is not possible without propagating an entire row or column of control points. This implies that when objects with much geometric detail are being modeled, a large number of NURBS control points are superfluous, i.e. serve no purpose other than to satisfy topological constraints (see Figure B.11).

Thanks to the underlying point-based concept, T-junctions are allowed in T-splines (see Figure B.10). Allowing partial rows gives T-splines the invaluable local refinement property. When modeling with T-splines, extra control points are only added to the T-Mesh where the designer needs to add more geometry variation. Similarly, during surface fitting process, one can adaptively add extra control points to regions of the T-Mesh where the approximation errors to the original surface are high.

B.4.5 Converting a Quad-Dominant Mesh into a T-Mesh

Quad-dominant meshes consist of mostly quadrilaterals and a small number of N-gons. Recall from Section B.4.1 that every edge in a T-Mesh is being associated with a knot interval, and the knot interval assignment must be satisfying the two rules. For a quad-dominant mesh, some extra edges may be needed to be added to the mesh to ensure the validity. The most convenient way to do is to perform a step of Catmull-Clark subdivision to the whole quad-dominant mesh. This way, every cell is quadrilateral in the subdivided mesh, and it is guaranteed to be a valid T-Mesh, i.e. the knot intervals can be assigned consistently over the whole mesh. However, the drawback of this simple solution is that it generates many superfluous control points, and hence unwanted wrinkles on the spline surface.

Since T-junctions are allowed in a T-Mesh, we propose another solution which locally subdivides each N-gon by generating extraordinary vertices and T-junctions. The rules of subdivision are listed as follows and illustrated in Figure B.12-Left.

- A : Generally, in each odd N-gon, one can insert a new vertex and connected to a new T-junctions in each edge of the cell. This creates N additional quads, N T-junctions and one extraordinary vertex ;
- B : In the case, where there are vertices of valence-3 in the original N-gon, more than one extraordinary vertices may result
- C : In the case, where N is even, to avoid creating unnecessary T-junctions, a new vertex is inserted and connected to every two vertex of the cell. This creates N/2 additional quads and one extraordinary vertex

Each time an edge is subdivided, its associated knot interval is divided by two. These three operations guarantee that knot intervals can remain coherent (see Figure B.12-Right).

B.5 Conclusion

In this chapter, we have introduced a recent parametric curves and surfaces representation call T-spline that is the representation chosen to be used in this thesis. T-splines are is a generalization of NURBS. This new scheme is a point-based representation in which there are some new ideas introduced, for instance, we have revisited the concept of a knot vector of a control points instead of the knot vector of the whole parametric curve. In order to show how the new notations are related to the old ones, we have revisited some classical parametric curves and surfaces representations : Bézier, B-spline, and NURBS.

Then, we have shown how the point-based concept in T-spline representation gives rise to

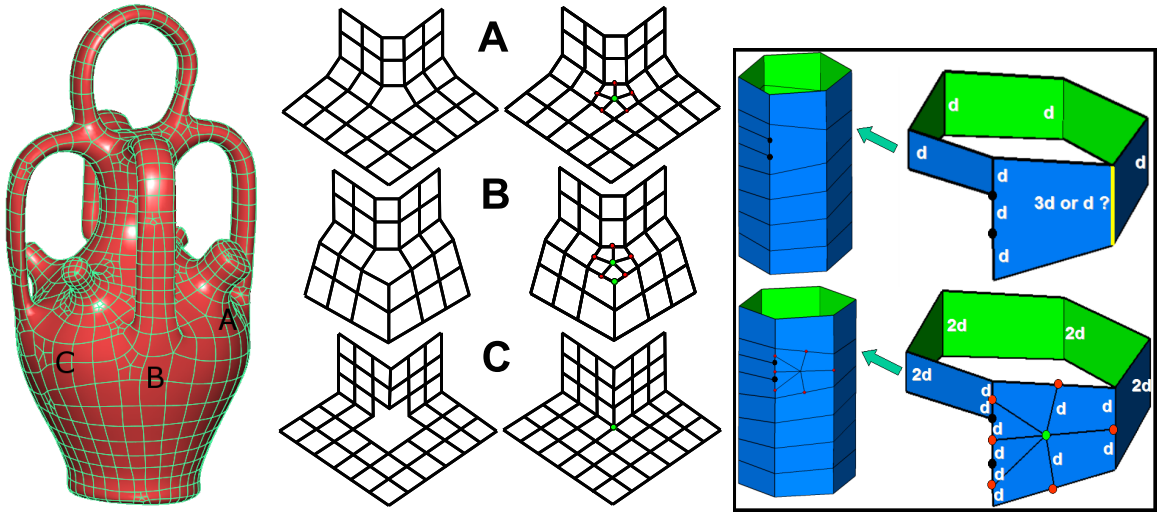


FIG. B.12: *Left : from N-sided polygon to extraordinary vertices ; Right : consistent knot interval assigning through creating of extraordinary vertices.*

the permission of T-junctions in a T-spline control mesh and hence an invaluable property : local refinement. This property solves the major drawback associated to NURBS surfaces, i.e. inserting extra control points is not possible without propagating an entire row or column of control points in the regular control mesh.

We also demonstrated that benefiting from this local refinement property, how to convert a quad-dominant mesh into a valid T-spline control mesh through creating T-junctions and extraordinary vertices, instead of globally subdividing the whole control mesh.

Annexe C

Control Mesh Drawing Data Structure

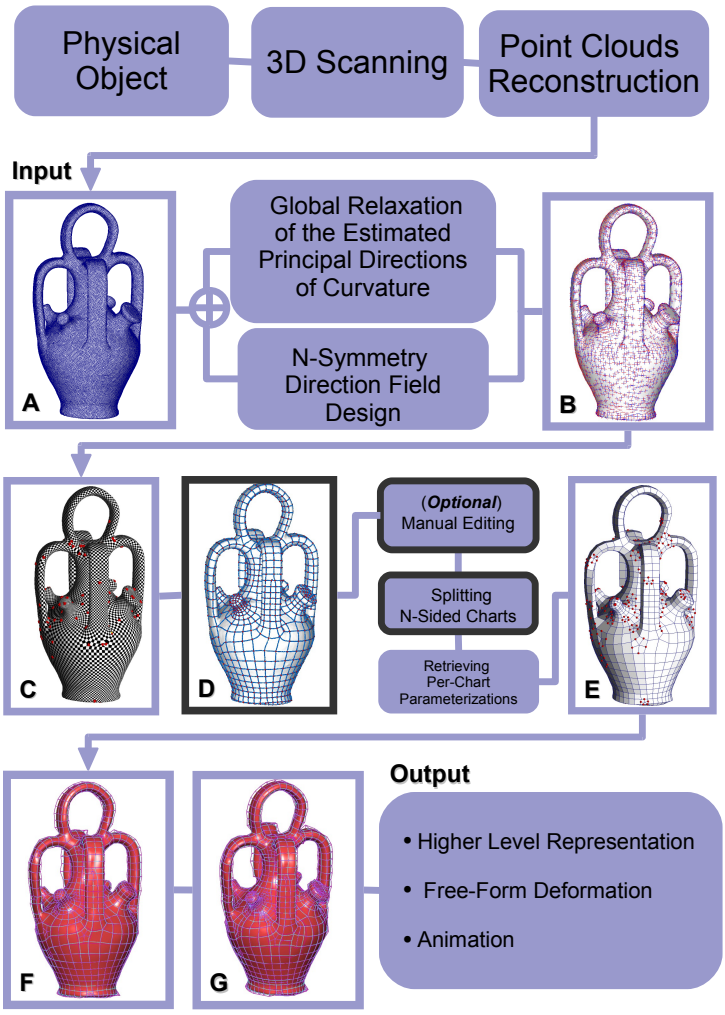


FIG. C.1: *Steps in the conversion pipeline that use the data structure.*

C.1 Introduction

In this chapter, we introduce a framework that facilitates control meshes to be efficiently drawn on meshes automatically. More specifically, we propose a new topological data structure for representing a set of polygonal edges, which we call "control mesh", embedded in a meshed surface (see Figure C.8). In this embedding, the vertices of the edges do not need to correspond to the vertices of the underlying mesh. For instance, for an edge between two vertices that is not embedded, we replace it with a *geodesic path* on the surface. The intersections of the edges are found automatically. The control mesh stores the combinatorial information of the network of the edges, i.e. the order of the incident edges of each vertex of the control mesh. This data structure is robustness and efficiency since the combinatorial form of information is systematically preferred to geometrical information. This property is especially useful to kept up-to-date the intersections of the edges. There are two main building blocks in this control mesh drawing framework :

1. how to define the geodesic path ("straight line") between two points on the surface, and
2. the data structure and algorithms.

Sometimes, one may also wish to manually edit the automatic result in an interactive way. In order to do so, we present a set of operations developed using our data structure in the spirit of 2D "Map-Sketching" [BG89, GHPT89].

The remainder of this chapter is organized as follows. In Section C.2, we will first discuss the geodesic and shortest path problem on surfaces. Then, we will introduce our solution to this problem. In Section C.3, we will introduce our *embedded cellular complex* data structure for control mesh drawing and explain how geodesic path is used to ensure strong embedding of the control mesh. In Section C.5, we will revisit the concept of Map-Sketching, and present the set of operations that helps to manually editing of the control mesh. Finally, in Section C.6, we will discuss some results on control mesh drawing using our framework.

C.2 Geodesic and Shortest Path Problem

A geodesic is a locally length-minimizing curve. In the plane, the geodesics are straight lines. On the sphere, the geodesics are arcs of the great circles. Here, we are only interested in finding a geodesic between two points on a triangulated mesh. Hofer and Pottmann[HP04] proposed a method which finds the geodesic between points using energy-minimized splines on a triangulated mesh.

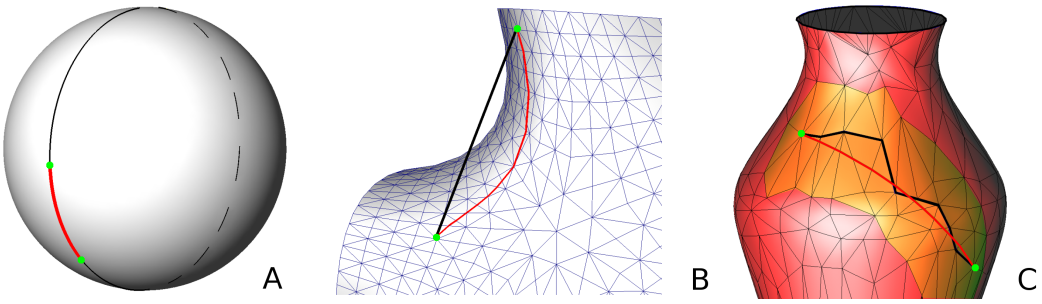


FIG. C.2: *The two green points are curve vertices. The red curves are the shortest path between the two vertices in each of the three figures. The black curve in each figure is another geodesic in A, a straight line in B, the Dijkstra's shortest path in C respectively.*

Sometimes, one needs not only a geodesic between two points but also needs it to be the shortest geodesic (which will be called shortest path henceforth, see Figure C.2-A)), for instance, to do texture synthesis on surface[NC99].

C.2.1 State of the Art

Many algorithms are proposed to do so which can be categorized into exact methods and approximate methods.

Exact solution

Mitchel *et al.* [MMP87] and Kapoor [Kap99] used wavefront propagation method to compute shortest path with a complexity of $O(n^2 logn)$ and $O(nlog^2 n)$ respectively, where n is the number of vertices. These methods fall into the Dijkstra paradigm ; Chen and Han[CH90] proposed non-Dijkstra method which finds shortest path on a polyhedral surface convex or non-convex, with or without border. This algorithm works by unfolding the facets of the mesh. Although it has a $O(n^2)$ complexity, it is considered to be the only feasible exact method due to its simple concept.

Approximate methods

Kanai and Suzuki [KS01] proposed an algorithm that computes the shortest path of a discrete weighted graph simplified from the original mesh. Then, this path is refined within a certain neighborhood. The shortest path found depends a lot on the first approximated path ; Kimmel and Sethian [KS98] proposed a method which runs in $O(nlogn)$. However, it can be

quite inaccurate even in planar surface. Kirsanov *et al.* [SSK+05] proposed an approximate method which runs also in the same time complexity as Kimmel and Sethian[KS98] but guarantees exact solution on a planar mesh.

C.2.2 Our Approach

The exact methods are generally with quadratic complexity or even worse, which means computationally expensive . This prevents them from being used in interactive applications especially when the meshed model is large. The approximate methods, in spite of the improved computational time, do not always compute a satisfactory accurate solution. On the other hand, exact methods execute in reasonable time provided that the number of vertices is small enough even with their quadratic complexity. Therefore, we introduce a heuristic that uses exact methods and significantly reduces the computational time by confining the search region for the shortest path to a subset of triangles.

Our heuristic works as follows : Provided that the starting and destination points on the mesh are not too far away one from each other, we can find an exact shortest path between these two points by restricting the search in a small local region. The method consists of two steps, we first find the Dijkstra's shortest path, i.e. the shortest path along the edges of the mesh (as shown by the black curve in Figure C.2-C), between these two points. Then, with this path, we find a ribbon-like neighborhood (as shown by the orange region in Figure C.2-C) along this path. This ribbon is simply obtained by finding the N-ring triangles along the Dijsktra's path, where N is the thickness of the ribbon. Within this neighborhood, we apply an exact method to find the shortest path between these two points. We use the implementation of the Chen and Han [CH90] provided by Kaneva and O'Rourke [KO00]. In this way, by limiting the number of vertices to the number of vertices in this band of neighborhood, our method reduces the computation time enough to be used for interactive mesh editing operations even on large meshed models.

C.3 Embedded Cellular Complex Data Structure

C.3.1 Definitions : Abstract Cellular Complex

This section gives the classic definitions and notations for abstract cellular complexes. The reader is referred to [Mas91] for more details.

– let Γ be a finite set.

The elements of Γ are called the *cells* of Γ ;

- let $\leq$ be a strictly partial order on Γ, i.e. a reflexive, anti-symmetric and transitive relation. The relation $\leq$ is referred to as the *bounding relation* ;
- consider the function $dim : \gamma \rightarrow \mathbb{N}$ characterized by :
  $\forall \gamma, \gamma' \in \Gamma \times \Gamma, \gamma \leq \gamma'$ and $\gamma \neq \gamma' \Rightarrow$
  $dim(\gamma) < dim(\gamma')$
  The function dim is called the *order function*.
  A cell γ such that $dim(\gamma) = k$ is called a *k-cell*, and k is called the *dimension* of γ ;
- the function dim yields a partition of Γ into $\Gamma^0 \ldots \Gamma^n$ defined by : $\forall \gamma \in \Gamma^k, dim(\gamma) = k$.
  The maximum dimension n is called the dimension of the cellular complex Γ.
- the *boundary* $B(\gamma)$ of a cell γ is defined by :
  $B(\gamma) = \{\gamma' \in \Gamma | \gamma' \neq \gamma$ and $\gamma' \leq \gamma\}$
- the *star* γ^* of a cell γ is defined by :
  $\gamma^* = \{\gamma' \in \Gamma | \gamma \leq \gamma'\}$
- a *geometric realization* of Γ is an isomorphism putting Γ in correspondence with a set of open sets Γ'. The order relation $\leq$ is translated to Γ' as follows :
  $\gamma_1 \leq \gamma_2 \Leftrightarrow \gamma'_1 \in \partial \gamma'_2$
  where $\partial \gamma'_2$ denotes the border of γ'_2. The geometric realization is said to be *conform* if the geometric cells of Γ' are disjoint.

C.3.2 Line Embedded in a Surface

To represent a control mesh embedded in a surface, we use two abstract cellular complexes and one relation connecting them :

- the control mesh can be represented by a 1D cellular complex $\Gamma = (\Gamma^0, \Gamma^1, \leq)$;
- the surface can be represented by a 2D cellular complex $\Sigma = (\Sigma^0, \Sigma^1, \Sigma^2, \leq)$;
- the cells of the control mesh are connected with the cells of the surface by an *inclusion* relation $\subseteq$ defined on $\Gamma \times \Sigma$.

Note that if the geometric realization is conform, we have :

$$\forall \sigma_1, \sigma_2 \neq \sigma_1 \in \Sigma \times \Sigma, \sigma_1 \cap \sigma_2 = \emptyset$$

then, for all $\gamma \in \Gamma$, we have at most one $\sigma \in \Sigma$ such that $\gamma \subseteq \sigma$. As a consequence, it is natural to represent the relation $\subseteq$ by the function $em : \Gamma \rightarrow \Sigma \cup \{\emptyset\}$, defined by :

$$\begin{array}{ll} em(\gamma) = \sigma \text{ where } \gamma \subseteq \sigma & \text{if } \sigma \text{ exists} \\ em(\gamma) = \emptyset & \text{otherwise} \end{array}$$

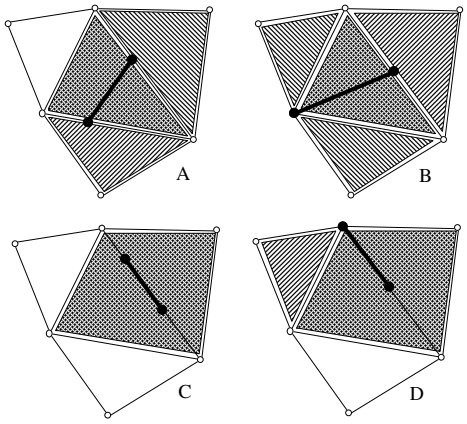


FIG. C.3: *The em^* function : to find the embedding of a segment of the control mesh Γ (bold segment), we first find in which cells its vertices are embedded, and compute the intersection of the stars of those cells (dashed zones).*

The function *em* will be referred to as the *embedding* function in what follows. We now suppose that the embedding function *em* is *combinatorially* represented at the vertices of Γ. From an implementation point of view, we suppose that each vertex of Γ has a pointer to a cell of Σ. For instance, if the control mesh Γ was manually digitalized on the surface Σ, the system stores in each vertex of Γ which cell of Σ was picked. Our goal is now to answer the following questions :

1. from the definition of *em* over the vertices of Γ, how can we deduce the definition of *em* over all the other cells of Γ ? In other words, can we find in which cells of Σ the edges of Γ are embedded ?
2. what are the combinatorial conditions of validity which make this extension of *em* possible ?
3. from an invalid configuration, how can we define an algorithm that enforces these conditions ?

To answer these questions, we consider the function $em^* : \Gamma \rightarrow \mathscr{P}(\Sigma)$ defined by :

$$em^*(\gamma) = \bigcap \{em(\gamma')^\star | \gamma' \in B(\gamma) \cap \Gamma^0\}$$

Intuitively, $em^*(\gamma)$ computes the embeddings of all the vertices of γ, and intersects the stars of all those embeddings. Trivially, from the definition of em^*, we have :

$$\forall \sigma \in em^*(\gamma), \gamma \subseteq (\sigma \cup B(\sigma))$$

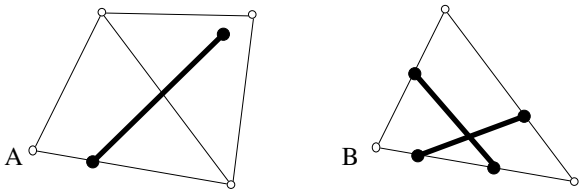


FIG. C.4: *Invalid embedding configuration : A : empty em^* ; B : non-conform control mesh.*

Figure C.3 shows what the em^* function looks like for certain cases. The two vertices of the control mesh (black dots) are embedded in cells of the surface. The stars of those two cells are displayed using two different hashing styles. The intersection is shown using crossed hashes.

However, as can be seen in the examples shown in Figure C.3-B and C, $em^*(\gamma)$ may contain too many cells. To determine the prolongation $\bar{em}(\gamma)$ of em, i.e. to find the cell of Σ in which γ is embedded, we need to filter-out the cells σ of $em^*(\gamma)$ such that $\gamma \subseteq B(\sigma)$. This can be easily done by finding the cell of minimum dimension in $em^*(\gamma)$:

$$
\begin{aligned}
&\bar{em}(\gamma) = em(\gamma) \quad \text{if } \gamma \in \Gamma^0 \\
&\bar{em}(\gamma) = \sigma \in em^*(\gamma) \text{ such that} \\
&\qquad dim(\sigma) = min\{dim(\tau) | \tau \in em^*(\gamma)\} \\
&\qquad\qquad \text{if } em^*(\gamma) \neq \emptyset, \\
&\bar{em}(\gamma) = \emptyset \quad \text{otherwise}
\end{aligned}
$$

In examples C and D in Figure C.3, $em^*(\gamma)$ contains two facets and one segments. Selecting the cell of lowest dimension enables the segment to be retrieved. Note that the so-defined $\bar{em}(\gamma)$ is unique, else this would contradict the conformity assumption (Σ would have two intersecting cells). This extension $\bar{em}$ of the em function answers question 1.

Now we want to answer question 2, e.g. identify the invalid configurations. As shown in Figure C.4-A, a segment γ of Γ cannot be embedded in a cell σ of Σ if its extremities are embedded in two cells σ_1 and σ_2 that are "too far away", i.e. if we cannot find a cell σ in Σ such that $\sigma_1, \sigma_2 \in (B(\sigma) \cup \{\sigma\})^2$. This condition also corresponds to $em^*(\gamma) = \emptyset$.

The other invalid configuration occurs when the control mesh Γ is non-conform (Figure C.4-B). Note that if we got a non-conform control mesh we have two segments $\gamma_1 \neq \gamma_2$ and $\gamma_1 \cap \gamma_2 \neq \emptyset$. If $\bar{em}$ is defined, we know that the two segments are embedded in the same cell σ of Σ, i.e. $\bar{em}(\gamma_1) = \bar{em}(\gamma_2) = \sigma$, and that the intersection $\gamma_1 \cap \gamma_2$ is embedded in σ. This will be used later to facilitate the computation of $\gamma_1 \cap \gamma_2$.

The domain of definition of the embedding function $\bar{em}$ makes it possible to distinguish three different classes of control mesh/surface relations. We say that the control mesh is :

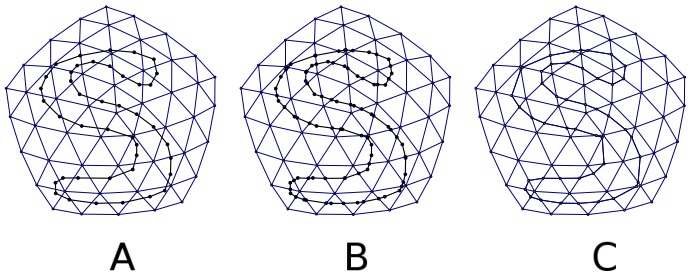


FIG. C.5: *Classes of control mesh/surface relations. A : weakly-embedded control mesh ; B : strongly-embedded control mesh ; C : realized control mesh.*

- **weakly embedded** (Figure C.5-A) if the function $\bar{em}$ is defined at the vertices only ;
- **strongly embedded** (Figure C.5-B) if the function $\bar{em}$ is defined at the vertices and the segments, i.e. $\forall \gamma \in \Gamma, \bar{em}(\gamma) \neq \emptyset$;
- **realized** (Figure C.5-C) if the cells of γ are embedded in cells of the same dimension in Σ, i.e. $\forall \gamma \in \Gamma, dim(\bar{em}(\gamma)) = dim(\gamma)$.

C.4 Algorithm

Using the above notions, it is possible to answer question 3 in the previous section, by designing an algorithm that enforces the *realized* condition from a *weakly embedded* and possibly *non-conform* control mesh :

1. **ensure strong embedding**
   For all $\gamma \in \Gamma^1$ such that $\bar{em}(\gamma) = \emptyset$, replace γ with the geodesic traced on Σ that links the two extremities of γ
2. **ensure conformity of control mesh**
   • For all σ in Σ^0, merge all the vertices of Γ embedded in σ.
   • For all σ in Σ^1, sort the vertices of Γ embedded in σ along σ.
   • For all σ in Σ^2, intersect all the edges of Γ embedded in σ (using a line-sweeping algorithm).
3. **realize the control mesh in the surface**
   Insert all the missing vertices and edges in Σ.

Note that the algorithm uses as much combinatorial information as possible : intersection computations are systematically restricted to the smallest possible cell of Σ, by using the $\bar{em}$

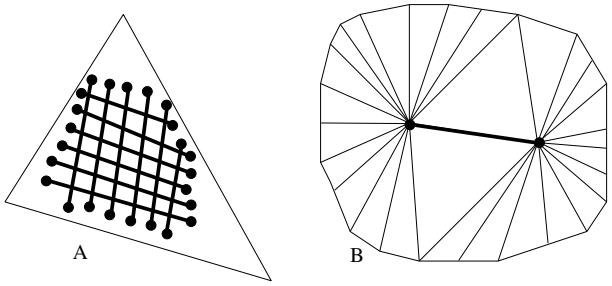


FIG. C.6: *Configurations of worst case complexity.*

function. This improves both robustness and efficiency as compared to a pure geometrical solution. All the intersections are either sorted along edges (1D) or within facets of Σ (2D). The only required geometric computations are the geodesic tracing algorithm (in step 1), sorting points along edges (in step 2.2) and segments intersections within facets (in step 2.3).

Figure C.7-Bottom shows initially non-conform control meshes made conform by applying step 2 after each edge design.

C.4.1 Implementation Details

From a practical point of view, each cell σ of Σ stores the list of cells $em^{-1}(\sigma) = \{\gamma | e\bar{m}(\gamma) = \sigma\}$ (in our implementation, each cell of Σ has `std : :vector` of pointers). To represent the control mesh Γ, our implementation uses a classical graph data structure. The surface Σ is represented by a half edge data structure (see e.g. [Bau75, Edm60, Lie94, Ket98, CGAL]). We use our implementation[Gra], that has the possibility of dynamically attaching information to the cells of the representation (the vertices of Γ store em and the cells of Σ store $e\bar{m}^{-1}$).

C.4.2 Complexity

Let $n = |\Gamma|$ denote the number of cells of the control mesh and $m = |\Sigma|$ the number of cells of the surface.

In our algorithm, the most costly step is the geodesic computation algorithm. The method we use (Chen and Han's method) has $O(m^2)$ complexity. With the heuristic we use, this reduces to $O(mlog(m))$ (at the expense of not always returning the shortest geodesic, which is not problematic for our application).

Worst case complexity

For enforcing the strong embedding condition, there are two degenerate configurations : if the control mesh is completely embedded in a single facet, the algorithm reduces to constructing a planar map of the control mesh, with a worst-case complexity of $O(n^2)$ (see Figure C.6-A). If the surface has two vertices with a high valence and the control mesh connects these two vertices, the bottleneck is the stars intersection computation (intersection of two sets), which costs $O(m\log(m))$ (see Figure C.6-B).

Average complexity

In real-life cases, the control mesh and the surface have similar resolutions, which means that each facet contains at most one intersection of the control mesh, and the complexity reduces to $O(n)$. In practice, all the examples shown in this paper run in less than one second.

C.5 Interactive Manual Editing by Map-Sketching

Sometimes, one may need to manually edit the automatic result of the control mesh drawing. Interactive manual editing of the control mesh often involves iterations of drawing and erasing. Moreover, new intersections between the new edges and the exiting control mesh should be dynamically found and updated. Therefore, the manual editing can be thought of as a *Map-Sketching* [BG89, GHPT89] (a concept of interactive 2D graphics shapes editing, see Figure C.7-Top) on meshes. With our embedded cellular complex data structure, we implement the manual editing by providing the following three basic operations :

- edge straightening : an edge of the control mesh is replaced with a geodesic ;
- edge insertion : a new edge is created (a geodesic) between two points picked on the surface. All the intersections are kept up-to-date ;
- edge deletion.

The manual editing is best illustrated with the Map-Sketching example shown in Figure C.7-Bottom, in which we design a complex shaped control mesh on a triangular mesh. The shape of the door is designed iteratively using the above three operations. To show that the control mesh is strongly embedded to the triangulated surface, and the intersections of the edges are well kept up-to-date, we deliberately disconnect the underlying mesh along the control mesh.

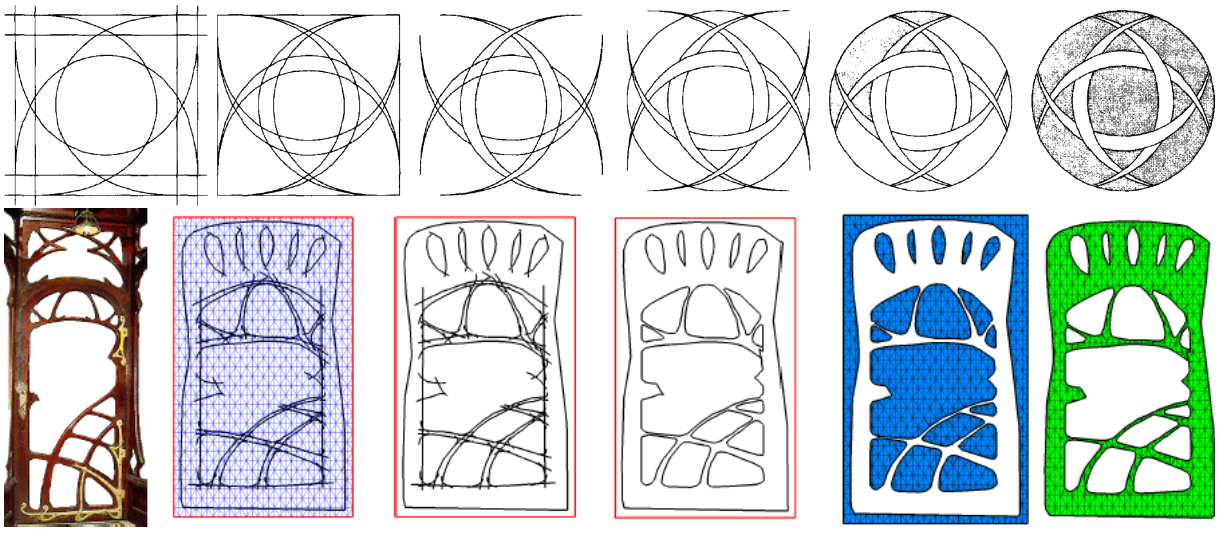


FIG. C.7: *Top : illustration of map-sketching using incremental computation of planar maps ; Bottom : sketching a complex shape on a mesh : The design is first sketched with edges. Then, the dangling edges segments in the control mesh re easily erased thanks to the topological data structure. Finally, the mesh is disconnected along the control mesh. (Note : The flat mesh used is for the sake of clarity only.)*

C.6 Results and Conclusion

In this chapter, we have introduced an embedded cellular complex data structure for control mesh drawing on meshed surfaces. In this data structure, the relations of a network of polygonal edges called control mesh and the underlying mesh to be maintained robustly and efficiently. Since the data structure is topological, the geometric information is only used during the computation of the intersections of the lines. Moreover, weakly-embedded paths between two extremities are replaced by geodesic paths on the surface. With this data structure, one can efficiently draw a control mesh of a surface using automatic methods and manual tools.

Figure C.8 shows the result of drawing a control mesh on the “Botijo” surface. One can see the embedding of the control mesh and the intersections nicely found inside the triangles. However, a limitation of our approach is that it requires a conform surface, i.e. the surface cannot self-intersect. Overcoming this limitation requires more complex data structures, since the embedding relation is not injective anymore.

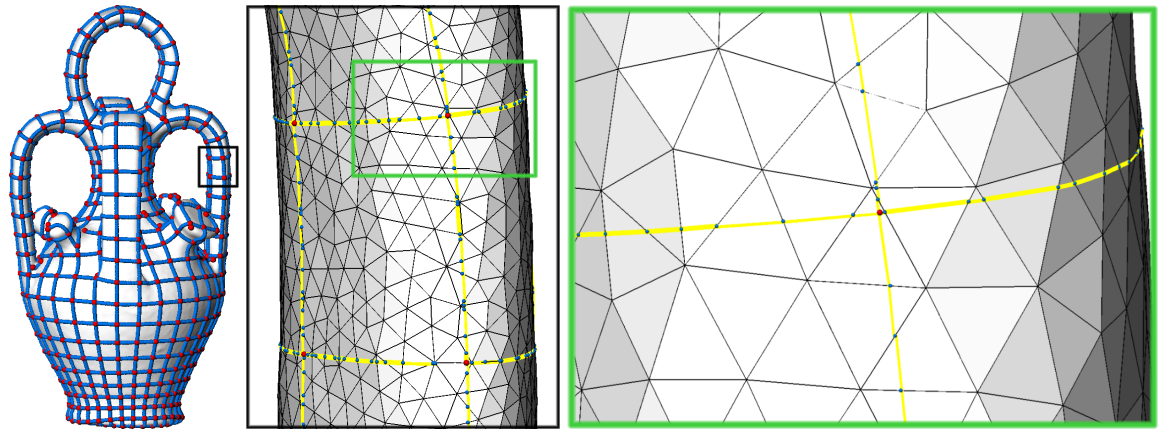


FIG. C.8: *A control mesh drawn on a meshed surface. Blue dots are control mesh vertices. Red dots are control mesh intersections.*

Annexe D

Principal Direction Field Smoothing

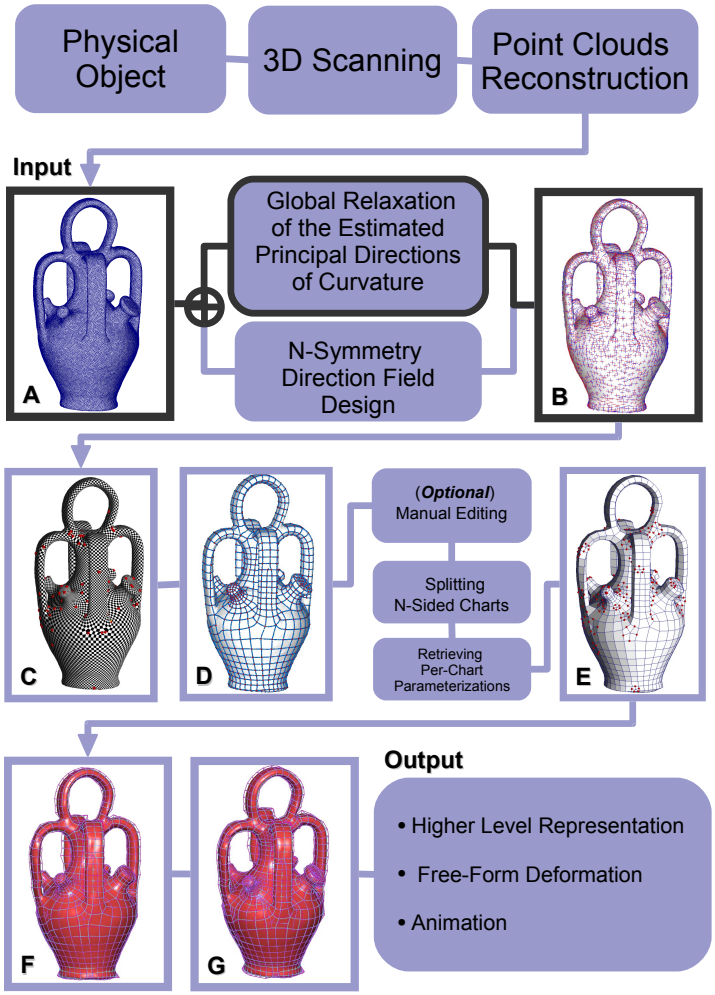


FIG. D.1: *Producing guidance direction fields by smoothing principal direction fields.*

D.1 Introduction

Spline surfaces require a quadrilateral control mesh of the object. As explained in [d'A00] by d'Azevedo, the edges of an optimal quadrilateral mesh of an object should be orthogonal and aligned with the principal directions of curvature of the surface (anisotropy-adapted). A anisotropy-adapted control mesh helps to reduce unwanted oscillations in the final spline surface.

In this thesis, to create such control meshes, we first compute *guidance direction field*s that will steer the placement of the edges. In other words, this direction field captures the anisotropy of the surface.

A direction field is a unit tangent vector fields ($\vec{u} \cdot \vec{u} = 1$, $\vec{u} \cdot \vec{n} = 0$), where $\vec{n}$ is the surface normal. The direction fields that we manipulate in this step have the property of N-symmetry which is mathematically characterized as follows (a more rigorous definition is given in Chapter E). For a point in a N-symmetry direction fields, we define N-symmetry directions, which are sets of directions invariant by rotation of $2k\pi/N$:

$$\vec{d} = \{\vec{u}_k = R_{\vec{n}}(\vec{u}_0, 2k\pi/N)\}$$

where $R_{\vec{n}}(., \theta)$ denotes the rotation by an angle of θ of the vector $\vec{u}_0$ around the normal $\vec{n}$ of the surface.

In this chapter, we propose a method to produce anisotropy-adapted guidance direction fields that is outlined as follows :

1. obtaining two direction fields that correspond to the principal directions of curvature of the surface from the estimated curvature tensor of the surface (using the method proposed in [CSM03]) ;
2. smoothing the direction fields through our new global relaxation approach.

The remainder of this chapter is organized as follows. In Section D.2, we will revisit the definition of the curvature tensor, and review the methods that estimate this quantity on triangular meshes. Then, in Section D.3, we will explain the existing and our methods to visualize principal direction fields. In Section D.4, we will present our direction field smoothing method, and discuss some results. Finally, we will conclude this chapter in Section D.6.

D.2 The Tensor of Curvature on the Surface

We have taken a concise exposition of the curvature tensor on surface S, namely, a 2-manifold embedded in Euclidean $\mathbb{R}^3$, from [Tau95] as follows (for detailed derivation of the

curvature tensor from classical differential geometry, see [Car76]). The tensor of curvature of the surface : $p \mapsto \kappa_p$, is the map of a point p on the surface to a function $\kappa_p(\vec{t})$, which measures the *directional curvature* of the unit tangent vector $\vec{t}$. The directional curvature function $\kappa_p(\vec{t})$ can be written in the quadratic form

$$\kappa_p(\vec{t}) = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}^t \begin{pmatrix} \kappa_p^{11} & \kappa_p^{12} \\ \kappa_p^{21} & \kappa_p^{22} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \tag{D.1}$$

where, $\vec{t} = t_1\vec{t}_1 + t_2\vec{t}_2$, $\{\vec{t}_1, \vec{t}_2\}$ is an orthonormal basis of the tangent space at p, the symmetric 2×2 matrix is called curvature tensor T_p^{2D}, where, $\kappa_p^{11} = \kappa_p(\vec{t}_1)$, $\kappa_p^{22} = \kappa_p(\vec{t}_2)$, and $\kappa_p^{12} = \kappa_p^{21}$. When $\kappa_p^{12} = \kappa_p^{21} = 0$, the vectors $\{\vec{t}_1, \vec{t}_2\}$ are called *principal directions* of S at p. The corresponding directional curvatures are the *principal curvatures*. For the sake of conciseness, we denote κ_p^{11}, κ_p^{12} and κ_p^{22} as a, b and c respectively. The principal directions and curvatures can be easily retrieved by doing the eigendecomposition of the curvature tensor.

$$T_p^{2D} = \begin{pmatrix} \vec{e}_1 \vec{e}_2 \end{pmatrix}^t \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} \vec{e}_1 \vec{e}_2 \end{pmatrix} \tag{D.2}$$

where, $\vec{e}_1$ and $\vec{e}_2$ are the two eigenvectors of T_p^{2D} corresponding to the eigenvalues λ_1 and λ_2 respectively, which are calculated in closed form as follows.

$$\lambda_{1,2} = \frac{(c+a) \pm \sqrt{(a-c)^2 + 4b^2}}{2} \tag{D.3}$$

and the associated eigenvectors $\vec{e}_{1,2} = (e_1^1, e_{1,2}^2)^t$, where,

$$e_1^1 = 2b, \text{ and } e_{1,2}^2 = (c-a) \pm \sqrt{(a-c)^2 + 4b^2} \tag{D.4}$$

We can extend the definitions of the directional curvature to non-tangent directions by using an orthonormal basis $\{\vec{n}, \vec{t}_1, \vec{t}_2\}$ of three-dimensional space.

$$\kappa_p(\vec{t}) = \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix}^t \begin{pmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix} \tag{D.5}$$

where, $\vec{t} = n\vec{n} + t_1\vec{t}_1 + t_2\vec{t}_2$, where, $\vec{n}$ is the normal of the surface at p and $\{\vec{t}_1, \vec{t}_2\}$ are the principal directions.

If one writes the vector $\vec{t}$ in another three-dimensional orthonormal basis $\{\vec{u}_1, \vec{u}_2, \vec{u}_3\}$ as $\vec{t} = u_1\vec{u}_1 + u_2\vec{u}_2 + u_3\vec{u}_3$, one can express the directional curvature as

$$\kappa_p(\vec{t}) = u^t \; T_p^{3D} \; u \tag{D.6}$$

where $u = (u_1, u_2, u_3)^t$, and T_p^{3D} is a 3×3 symmetric matrix, from which, one can recover the principal curvatures and directions of S at p by restricting K_p to the tangent plane and then compute the eigenvalues and eigenvectors of the 2×2 matrix.

Degenerate points of the curvature tensor field

Definition : [Tri02] A degenerate point of a two-dimensional second-order, symmetric tensor field is a point, p on the surface where the field is isotropic, namely, every non-zero vector is an eigenvector. At this point, the tensor is independent of the coordinate system and is proportional to the identity matrix, $\mathbf{I}_2$. The tensor at this point can be written as

$$T_p^{2D} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = \lambda \mathbf{I}_2 \tag{D.7}$$

D.2.1 Estimation of the Curvature Tensor from a Mesh

State of the art

Finding a good estimation of the curvature tensor for a triangular mesh has always been an active research area Geometry Processing and Computer Graphics due to its variety of applications. For instances, the principal directions of the surface serve as a very good hint for quality surface control mesh extraction/quad-remeshing [ACSD+03] and illustrating smooth surfaces [HZ00].

Many methods use the approach of fitting an analytic surface to the immediate neighborhood of a vertex and then extract the curvature information from it (see e.g. [Pet01],[MK04a]). For instances, in [WW94] and [GI04], they used a second order Taylor polynomial (1-ring) and a cubic approximation scheme (using a 2-ring) respectively.

Some methods make use of the discrete normal curvature along edges of the one-ring. In [Tau95], a symmetric matrix is found whose eigenvalues are related to the principal curvatures by linear relations and whose eigenvectors are the principal directions. In [MDSB02], they proposed discrete representations for the mean and Gaussian curvatures for triangulated surfaces using only the 1-ring of the vertex. The two principal curvatures are then expressed in term of the mean and Gaussian curvatures. To determine the two principal directions, normal curvatures

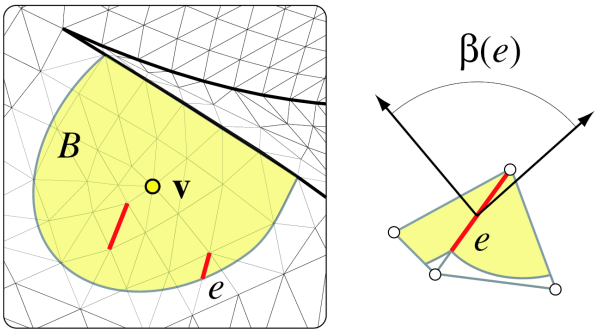


FIG. D.2: *Estimating the curvature tensor at vertex v using Cohen-Steiner and Morvan's method [CSM03].*

along each edge emanating from the vertex is computed. Then, a 2×2 matrix is found by least-square fitting the normal curvatures of the edges. The eigenvectors of the matrix then give the principal directions of the surface at the vertex.

As pointed out in [CSM03], estimating the curvature tensor at a vertex by using only the 1-ring is not natural since every edge already defines a tensor (maximum curvature across the edge and minimum curvature along the edge). Therefore, they advocated to integrate curvature tensor over an arbitrary region around a vertex using a simple expression given in Equation D.8. This is the formula that we have chosen to estimate the curvature tensor of surfaces in this thesis.

$$T^{3D}(v) = \frac{2}{|B|} \sum_{edge\ e} \beta(e)|e \cap B| \overline{e}\ \overline{e}^t \tag{D.8}$$

where v is a vertex, $|B|$ is the region around v over which the tensor is estimated, $\beta(e)$ is the signed angle between the normals to the two oriented triangles sharing e, $|e \cap B|$ is the length of $e \cap B$, and $\overline{e}$ is a unit vector in the same direction as e.

While all the above methods estimate discrete curvature tensors at vertices, in [TRZS04], a method is proposed to give continuous estimation of the curvature tensor over each triangle. The basic idea is quite simple. From classical differential geometry, it is well known that for an embedded surface in $\mathbb{R}^3$, the curvature tensor at a point is completely defined by its partials and the partials of the unit normals. Therefore, in their method, for each triangle, it is parameterized using a local s,t coordinates. Similarly, the normal over the triangle is defined using the three normals at vertices. Then, the curvature tensor is computed using the first and second fundamental forms. Despite the simplicity, this method suffers from drawbacks, such as, non-continuous tensor across triangles and ambiguity at vertices.

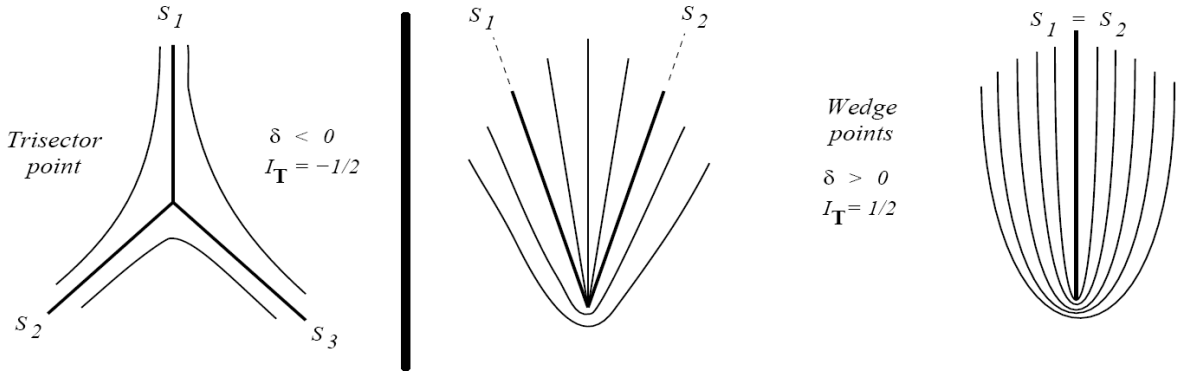


FIG. D.3: *Left : Trisector (index I=$-\frac{1}{2}$) ; right : wedge (index I=$\frac{1}{2}$)*

Linear interpolation of the tensor in a triangle

Now that we have a 3D curvature tensor T^{3D} estimated on each vertex of the mesh, in order to define the tensor over the whole surface, we linearly interpolate the curvature tensor defined on the vertices of the mesh in the facets using barycentric coordinates. Since a triangular facet is flat, namely, with constant normal, it is only meaningful to interpolate the 2D version of the tensor on the tangent plane, namely, T^{2D} over the facet. One convenient way to do so is to compute the two principal directions, which are the two eigenvectors corresponding to the two largest eigenvalues of T^{3D}, on each vertex and project them onto a 2D local basis in the tangent plane defined by the three vertices of the facet. Thus, we obtain the $\vec{e}_1$ and $\vec{e}_2 \in \mathbb{R}^2$. Then, by using Equation D.2, one can easily reassemble the T^{2D}, which can be represented as a triplet $\{a,b,c\}$ due to the symmetry of the matrix . Finally, the triplet can be interpolated linearly in the facet using barycentric coordinates.

Locating and classifying degenerate points of a curvature tensor field

As pointed out in [Tri02], using linear interpolation of the tensor in the triangle, there can only have only one degenerate point in a triangle and it is either a *wedge* or *trisector* (see Figure D.3), which has index of $+\frac{1}{2}$ and $-\frac{1}{2}$ respectively. Using Equation D.7, inside a triangle, the degenerate point p_o can be located by solving the following set of equations.

$$\begin{cases} a(p_o) - c(p_o) &= 0 \\ b(p_o) &= 0 \end{cases} \tag{D.9}$$

Then, the type of the degenerate point, p_o can be classified using the quantity δ defined in Equation D.10 [DH94]. If $\delta > 0$, the degenerate point is a wedge, otherwise it is a trisector.

FIG. D.4: *Left : Curvature tensor visualized as ellipsoids ; center (resp. right) : principal direction field corresponds to the minimum (resp. maximum) curvature.*

$$\begin{aligned} \delta &= AD - BC \\ \text{where,} & \\ A &= \tfrac{1}{2}\tfrac{\partial(a-c)}{\partial x} \qquad B = \tfrac{1}{2}\tfrac{\partial(a-c)}{\partial y} \\ C &= \tfrac{\partial b}{\partial x} \qquad D = \tfrac{\partial b}{\partial y} \end{aligned} \tag{D.10}$$

In practice, partial derivative A, B, C and D can easily be found by computing the gradient in the triangle using the following formula.

$$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix} = \frac{1}{2 \cdot Area\_of\_Triangle} \begin{pmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \tag{D.11}$$

where, u_1, u_2 and u_3 are the vertex value of the function whose gradient is to be determined in the triangle, and (x_1, y_1), (x_2, y_2) and (x_3, y_3) are the coordinates of the three vertices of the triangle in a local 2D orthonormal basis.

Visualizing curvature tensor fields

The most straight-forward way to visualize a curvature tensor field on a surface is to consider the tensor at each point as a ellipsoid (Equation D.12) with the three eigenvalues of the curvature tensor as the lengths of the three semi-axes and the corresponding three eigenvectors as the three directions of the axes of the ellipsoid.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{D.12}$$

In practice, to visualize the ellipsoid on each vertex of the mesh, one can adopt the technique as introduced in [TL04], that ray-traces the ellipsoid by the GPU at a interactive frame rate for

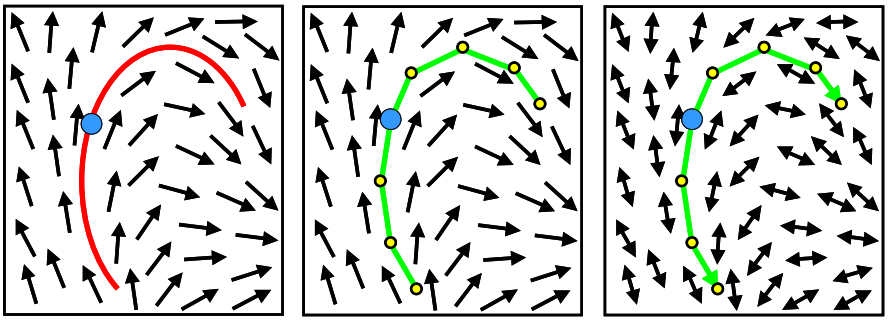


FIG. D.5: *Left : a streamline ; Center : numerical integration of a streamline in a vector field ; Right : numerical integration of a streamline in a vector field with sign-ambiguity.*

visualization (see Figure D.4-left).

Although the above method has the advantage of visualize all the information of the curvature at one time, it is usually uncapable to depict the topology of the tensor field. However, if one isolates the two principal directions of the surface into two separate principal direction fields, the topology of the tensor field becomes much more perceptible (see Figures D.4-Center and Right). The two principal directions are simply the two eigenvectors corresponding to the two largest eigenvalues of the curvature tensor.

D.3 Visualizing Principal Direction Fields

A principal direction field is a unit vector field with a sign-ambiguity, i.e. 2-symmetry direction field. The ambiguity arises from the fact that the two principal directions at each point of the surface are the eigenvectors of the curvature tensor. Despite this difference from classical vector fields, with some suitable adjustment, techniques for the visualization of vector fields can be easily adapted to visualizing principal direction fields. One of the most popular ways to visualize vector fields is by sampling streamlines which characterize the local behavior of the field.

Definition of a streamline

A streamline is a curve $\gamma(s) : \mathbb{R} \rightarrow \mathbb{R}^n$, everywhere tangent to the vector field $\vec{v}$, such that it satisfies the ordinary differential equation (see Figure D.5-Left) :

$$\gamma'(s) = \vec{v}(\gamma(s)) \tag{D.13}$$

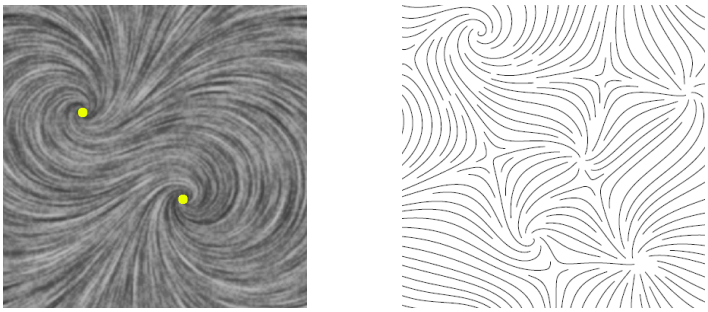


FIG. D.6: *Left : Line Integral Convolution ; Right : placement of streamlines*

Tracing streamlines in vector fields by numerical integration

To solve the differential equation D.13, one can apply the simple Euler's method, described in Equation D.14, that defines a piecewise-linear approximation of the streamline (see Figure D.5-Right). One can also use methods such as the fourth-order Runge-Kutta integration with adaptive step.

$$\begin{aligned} p_i &= p_{i-1} + \vec{v}(p_{i-1})\Delta s \\ p'_i &= p'_{i-1} - \vec{v}(p'_{i-1})\Delta s, \ \ where\ p'_0 = p_0 \end{aligned} \tag{D.14}$$

where, $p = \gamma(s)$, Δs is the step size of the integration, p_0 is the starting point of the streamline and p_i and p'_i denote points in the positive and negative direction respectively.

Tracing streamlines in principal direction fields

There are two families of methods that use streamlines to visualize the vector field. In the following, we are going to review the two families and how we adopt these vector field visualization techniques to curvature direction fields.

D.3.1 First Family : LIC (Line Integral Convolution)

Line Integral Convolution was proposed by Cabral and Leedom [CL93] to perform a texture synthesis for the visualization of the 2D vector fields. The basic idea is that the local behavior of the vector field can be approximated by tracing a streamline starting from a point in its positive and negative directions. By convolving an input white noise texture TEX_{noise} with a low pass filter $\tau(w)$ along this streamline, the pixel intensity is highly correlated along individual streamlines but independent in the perpendicular direction. Denoting a streamline $\gamma(s)$, line

integral convolution gives the intensity for a pixel $x_0 = \gamma(s_0)$

$$Intensity(x_0) = \int_{s_0-L}^{s_0+L} \tau(s-s_0) TEX_{noise}(\gamma(s)) ds \tag{D.15}$$

State of the art

- *2D LIC :*
  Since the introduction of LIC by Cabral and Leedom in 1993, the method has been widely used for the visualization of 2D vector fields. In this work, we have also chosen to use the concept of LIC to visualize our vector fields. After the invention of this method, many works have been done to accelerate the original algorithm. For instance, the FastLIC method presented in [SH95], which employs simple box filter and minimizes the total number of streamlines and hence accelerates the original LIC by an order of magnitude.
- *LIC on surfaces :*
  Later, work have been done to extend the 2D LIC technique to vector fields on surfaces. As said in [LvWJH04], these methods can be classified into mainly three types according to the space in which the LIC operates.
  - *Parametric space :* Forssell and Cohen [For94, FC95] proposed a method that allows the operation of LIC through a parameterization [FH05] of the surface. However, a global parameterization of a general surface may not always be obtained easily. Battke *et al.* [BSH97] and Carr *et al.* [CH02] propose methods that allow the operation of LIC in parametric space by triangle packing algorithms. Triangle packing can be considered as the simplest local parameterization of the surface by triangle. Nevertheless, the drawback of triangle packing is that it is quite sensitive to the quality of the mesh.
  - *Object space :* by immersing the vector field on the surface into a 3D volume, the vector field can be considered as a 3D vector field where streamline tracing can be done [RSHCE99, Int97]. However, the visualization would be limited to the maximum resolution of the 3D texture, thus causing problems with zooming.
  - *Image space :* recently two methods ISA (Image Space Advection) [LSH03] and IBFVS (Image Based Flow Visualization for Curved Surfaces) [vW03] were proposed to enable high-performance visualization of flow on surfaces. A detailed side-by-side comparison of the two methods can be found in [LvWJH04]. Working in this space, no parameterization of the surface is needed. Moreover the LIC process can be accelerated by taking advantage of graphics hardware. Seeing the advantages of image space based methods, in this work, we have chosen to carry out the LIC in image space as well.

Our LIC on surfaces

We present a GPU-accelerated LIC-based approach working in image space, which is inspired by the ISA (Image Space Advection) method proposed in [LSH03]. The algorithm is done in three passes.

1. *Depth value for detecting geometric discontinuities :*
   One of the disadvantages of decoupling the LIC process with the 3D surface geometry (since the method operates in image space) is that undesired visual continuity across the geometric discontinuities could be resulted due to the projection of a self-occluding surface onto the image plane (as pointed out in [LSH03]). We adopt the criteria to distinguish the geometric discontinuities proposed in [LSH03] as follows.

   $$\|z_{i+1} - z_i\| > \varepsilon \|p_{i+1} - p_i\| \tag{D.16}$$

   where ε is a threshold. The test compares the depth values in the object space z_i and $z_{i+1}$of two consecutive points p_i and p_{i+1} along the integral path in the image plane. A positive result of the test identifies a geometric discontinuity. To allow performing the above test in the LIC pass, the Z-buffer is rendered to the framebuffer and stored as a texture, TEX_{depth}.

2. *Object space to image space projection :*
   For each visible point p of the surface, we calculate its direction in image space as follows. Firstly, we project the directions in the object space of the three vertices of the triangle to which p belongs onto the image plane. Since the direction is sign ambiguous, the direction of p in the image plane cannot be determined directly by linear interpolation of the three direction vectors. Instead we need to interpolate the three 2D curvature tensor matrices. For principal direction fields, one can obtain this 2D curvature tensors easily as described in Section D.2.1. Then, the tensor is linearly interpolated across the triangle. The direction at a point is determined (using Equations D.3&D.4) by finding the eigenvector of the tensor corresponding to the corresponding eigenvalue. Since the solution of the eigenvalues can be computed in a closed form, the interpolation is easily accelerated by using a programmable GPU.

   Let the normalized version of the eigenvector be $\vec{v}_{img}$. Since the LIC is to be done on a GPU, the directions at the points along the forward and backward line integral path need to be known in advance. We achieved this by storing the direction field of the surface in image plane using a texture : the vector $\vec{v}_{img}$ is encoded as color and rendered to the

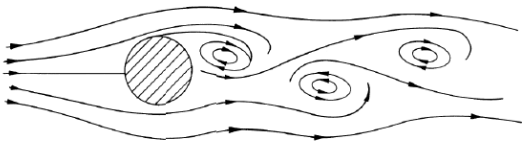
framebuffer, which will be resampled as texture in the third pass. We call this texture $TEX_{direction}$. $\vec{v}_{img}$ is encoded as $\vec{c} = \{r = \vec{v}_{img}.x,\ g = \vec{v}_{img}.y,\ b = 0\}$. Since each component of $\vec{v}_{img}$ is in [-1,1], In order to be store as a texture, the color is recalibrated into the range [0,1] by applying $\vec{c} = \vec{c} \cdot 0.5 + 0.5$.

3. *LIC in image space :*

   Now, we present how we carry out LIC in image space using a programmable GPU as follows. For a visible point on the surface projected onto the image space, starting from this point p_0, the streamline is traced in positive and negative directions by coordinate advection using Equation D.14. The direction at point p is given by $\vec{v}_{img}(p)$, which is the vector in image plane obtained by resampling the vector image $TEX_{direction}$ obtained in the first pass. After the resampling, $\vec{v}_{img}(p)$ is decoded from the color $\vec{v}_{img} = 2 \cdot (\vec{c} - 0.5)$. The time step Δs is defined such that $n \cdot \Delta s = L$, where n is the number of steps and L is the length of the streamline in one direction, in both directions. We use only the direction of the vector as did in [CL93] and we found that the results given are satisfactory. As for the convolution kernel, we use a rather simple low-pass filter which is of value $1/2L$ in $[-L, L]$ and zero elsewhere, namely, the average of the $2n+1$ sample of the TEX_{noise} along the streamline. We have chosen this box filter basically for reasons of performance. To implement more complicated kernels such as Hanning filter as done in [CL93], one needs more precomputation and the extra storage (a one-dimensional texture as discussed in [GL05]).

   The integral in a direction (positive or negative) stops at p_i whenever the position p_i gives a positive result to Equation D.16 by resampling the depth information, TEX_{depth}, stored in the second pass. Random noise value is assigned to unsampled points on the rest of the streamline. This ensures visual discontinuities at geometric discontinuities.

D.3.2 Second Family : Placement of Streamlines



The idea is that, given a specified density, a set of streamlines is traced to cover a given domain where the vector field is defined. This family of approach has been pioneered by the work of Turk and Banks [TB96] to simulate the hand-designed vector field illustrations in textbooks (see figure on the right). Each streamline starts from a seed point and terminates when it is too close to one of the existing streamlines, reaches the vicinity of a singularity of the vector field, reaches the domain boundary or forms a closed loop. The difficulty of this family of method is to find a good seeding strategy such that the streamlines

traced are as continuous as possible, which enhances the perception of the global topology of the vector field being visualized. In the following, we will first review different strategies of the placement of streamlines of vector fields in 2D, surfaces. Then, we will describe our approach of placement of streamlines of vector fields defined on surfaces.

State of the art

– *2D :*

Inspired by the hand-drawn streamline illustrations in physics texts, Turk and Banks [TB96] added the parameter *density* to the visualization of vector fields. In this pioneering work, they proposed a streamline placement method that being given a specified density of streamlines, gives a well-populated placement of the streamlines by using an energy-minimizing approach. The energy function takes a low-pass filtered version of the current output and measure the difference between it and the desired density. Hence, this method is sometimes referred as an image-guided method.

In order to accelerate the pioneering energy-minimizing approach, different greedy algorithms are proposed to the placement of streamlines problem by using different strategies of *seeding* of streamlines. Jobard and Lefer [JL97] proposed to derive all the seed points possible to find from an existing streamline before trying with another existing one. This is done by storing, in a queue, the positions at both sides of the streamline being traced using numerical integration for the seed positions of future streamlines. This method results in nice evenly-spaced streamlines. In this thesis, we adopted a similar seed strategy to perform placement of streamlines of vector fields on surfaces. Verma *et al.* [VKP00] proposed a flow-guided seeding strategy. They first identify the location and type of the critical points in the vector field and use different seeding patterns around the critical points according to their type. Then, they fill the blank region left by tracing streamlines seeded using poisson disk distribution. Since the density of the streamlines is determined mainly by the distance between the seeds, the method results in a less even-spaced placement of streamlines as compared with the results given in Jobard and Lefer's method, in which the desired density is achieved by controlling directly the distance between nearby streamlines. Another drawback of this method is that the seeding using poisson disk distribution around saddle type critical points is not so satisfactory. Recently, Mebarki *et al.* [MAD05] proposed a new greedy algorithm that achieved comparable quality with respect to the pioneer work by Turk and Banks but much faster. The basic idea is to place seeds at the farthest point from from all existing streamlines. By doing so, long

FIG. D.7: *Left : estimated principal directions ; Center : smoothed principal directions visualized using LIC ; Right : the same field visualized using placement of streamlines. Red dots are wedges while green dots are trisectors.*

and evenly spaced streamlines are favored. They also showed results of multiresolution placements.

– *Surface :*
In the context of quad-remeshing, Alliez *et al.* proposed a streamline placement method on surfaces. In order to do numerical integration of the streamline, and applying the previous 2D streamline placement techniques, they used a global parameterization of the surface. However the need of a global parameterization limits the method to single-charted surfaces. Later, Marinov and Kobbelt [MK04b] improved the method by doing the numerical integration directly on the surface without the need of the global parameterization.

In the thesis, we have adopted Marinov and Kobbelt's method to implement our streamlines placement on surfaces. An example of is shown in Figure D.7-Right.

D.4 Principal Direction Field Smoothing

As said before, we have adopted Cohen-Steiner and Morvan's method to estimate the curvature tensor on a surface, which gives the two principal direction fields of the surface. This pair of orthogonal principal direction fields will be used as the guidance direction fields for our periodic global parameterization. However, before this information can be *practically* used as the guidance fields, there are still two problems to be addressed, which are explained as follows.

1. In isotropic regions, the principal directions of the surface are undefined. As a consequence, the estimation is meaningless in regions where the anisotropy $(|\frac{kmax}{kmin}| - 1)$ va-

nishes.

2. Since the control mesh means only to capture the *global geometry* of the surface, it is desirable that the local imperfections (due to the piecewise-linear nature of the input mesh) and the insignificant local topology of the tensor field be removed.

In order to tackle the above two problems, previous work such as [ACSD+03] used a Gaussian filtering *directly* on the curvature tensor in a coefficient-by-coefficient manner. However, despite the simplicity, the Gaussian-filtering approach is slow when the size of the mesh gets bigger and sometimes quite inefficient to get rid of the insignificant local topology of the tensor field.

Since what we are interested in are the principal directions rather than the whole curvature tensor, instead of smoothing the curvature tensor field, we smooth the two principal direction fields, which are given by the two eigenvectors of the curvature tensor that correspond to the two largest eigenvalues. Moreover, since our principal direction fields are 2-symmetry direction fields, we are particularly interested in direction field smoothing methods that takes into account of the symmetry of the direction field. In the following, we will first review two direction field smoothing methods by using local and global relaxation respectively. Then, we will introduce our global relaxation approach that uses *periodic variables* to deal with the symmetry of the principal direction fields, which allow the use of efficient numerical solvers and exhibit faster convergence compared with the previous global approach.

D.4.1 Previous Work

Local approach

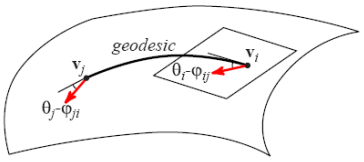
In the context of texture synthesis on arbitrary surfaces using "N-way" symmetric textures, Wei and Levoy [WL01] suggested a local relaxation scheme that iteratively update the direction of the vectors defined on the vertices of the mesh by minimizing the following error function.

$$E = \sum_{v_j \; near \; v_i} \left| \phi_{ji} - round\left(\frac{\phi_{ji}}{\frac{2\pi}{n}}\right) \cdot \frac{2\pi}{n} \right|^2 \tag{D.17}$$

where, n is the symmetry of the direction field, v_i and v_j are vertices and ϕ_{ji} is the angle between the direction at v_i and the projection of the direction at v_j on the local coordinate system of v_i.

Although this method generates direction fields with the required symmetry, the local strategy is generally slow and often keeps unwanted local topology of the vector field, especially when the surface is of high geometric details.

Global approach



In the context of obtaining a "cross field" for guiding the cross-hatching to illustrate smooth surfaces, Hertzmann and Zorin [HZ00] proposed a global relaxation to generate/smooth 4-symmetry direction fields. They suggested a simple way to compare directions on adjacent vertices of a mesh by adopting the concepts of geodesic polar map (or exponential map) and parallel transport, which enables a global energy functional to be defined. The underlying idea is that a common reference can be obtained between two sufficiently close points on a surface by parallel transport a vector in the tangent plane of one point to the other along the geodesics. On a mesh, the tangents to the geodesics between two adjacent vertices v_i and v_j is approximated by the projection of the edge $\vec{e}(v_i, v_j)$ into the tangent planes at the vertices. Then, they arrive at the following energy functional.

$$E = - \sum_{all\ edges(v_i,v_j)} cos4((\theta_i - \varphi_{ij}) - (\theta_j - \varphi_{ji})) \tag{D.18}$$

where, θ_i is the angle in the tangent plane on the vertex v_i defined on a local orthonormal basis and φ_{ij} is the direction of the projection of the edge $\vec{e}(v_i, v_j)$ into the tangent plane of v_i.

The above optimization method could be initialized by the principal directions of the surface where they are marked as reliable.

D.4.2 Our Global Smoothing Approach

The existing two direction field smoothing methods [WL01] and [HZ00] are limited speedwise due to the local relaxation approach and to the non-linearity of the objective function respectively. Therefore, we introduce a new global relaxation approach that use simple quadratic minimization procedure.

The input of this algorithm is a principal direction field defined on the mesh, $\vec{K}$. More specifically, we describe a procedure that smooths the input direction fields and extrapolates them into mesh areas where the associated directions are ill-defined. To obtain direction fields which are well defined everywhere, we introduce a method for "extending" the directions from the anisotropic regions of the surface onto the isotropic ones. In the following, we will first introduce the energy functional. Then, we will explain how to adapt the energy functional to direction fields with higher-symmetry, e.g. principal direction fields (2-symmetry) and cross fields (4-symmetry).

Defining the energy functional for direction fields

We apply a regularized fitting procedure to a set of variables α_i, corresponding to the direction of $\vec{K}_i$. α_i is defined as the angle between the vector $\vec{K}_i$ and a reference direction $\vec{H}_i$ in the tangent plane of the vertex i. To obtain a reference direction we select an edge $\vec{e}$ emanating from i and project it to the plane :

$$\vec{H}_i = \text{normalize}\left(\overrightarrow{e} - (\overrightarrow{e} \cdot \vec{N}_i)\vec{N}_i\right)$$

where $\vec{N}_i$ is the normal at i.
To smoothly fit the curvature directions, we minimize an energy functional providing a balance between fitting and smoothness. The fitting term aims at keeping the new α_i angles close to the original angles α_i^0, computed form the initial values of the direction field $\vec{K}$ at the vertices. The angle differences are approximated by the norm of the difference of the sine/cosine vectors, yielding the following formulation :

$$R = (1-\rho) \underbrace{\sum_{\text{all vertices } v_i} \left|\frac{kmax_i}{kmin_i}\right| \left\| \begin{pmatrix} \cos\alpha_i \\ \sin\alpha_i \end{pmatrix} - \begin{pmatrix} \cos\alpha_i^0 \\ \sin\alpha_i^0 \end{pmatrix} \right\|^2}_{\text{fitting term}} + \rho \underbrace{\sum_{\text{all triangles } T} R_T}_{\text{smoothing term}} \tag{D.19}$$

where $kmax_i$ (resp. $kmin_i$) is the maximum (resp. minimum) curvature at vertex i. The user-defined coefficient ρ corresponds to the desired smoothing intensity (in all our examples, $\rho = 0.8$). The smoothing term R_T on a triangle T minimizes the variations of α over T and is given by :

$$R_T = \sum \lambda_i^T R_{T,i}$$

where the variation $R_{T,i}$ of α along the edge e_i is given by :

$$R_{T,i} = \left\| \begin{pmatrix} \cos\alpha_{i\oplus 2} \\ \sin\alpha_{i\oplus 2} \end{pmatrix} - \begin{pmatrix} \cos\beta_i & \sin\beta_i \\ -\sin\beta_i & \cos\beta_i \end{pmatrix} \begin{pmatrix} \cos\alpha_{i\oplus 1} \\ \sin\alpha_{i\oplus 1} \end{pmatrix} \right\|^2 \tag{D.20}$$

where, β_i denotes the angle between $\vec{H}_{i\oplus 1}$ and $\vec{H}_{i\oplus 2}$. This angle is computed in a similar manner as described in the global approach in Section D.4.1 that one can compare two directions defined at the tangent planes of two adjacent vertices using the geodesics. Specifically, one computes $\cos\beta_i = \vec{H}_{i\oplus 1} \cdot \vec{H}_{i\oplus 2}$ and $\sin\beta_i = (\vec{H}_{i\oplus 1} \times \vec{H}_{i\oplus 2}) \cdot N_T$. The constant terms λ_i^T's used to define

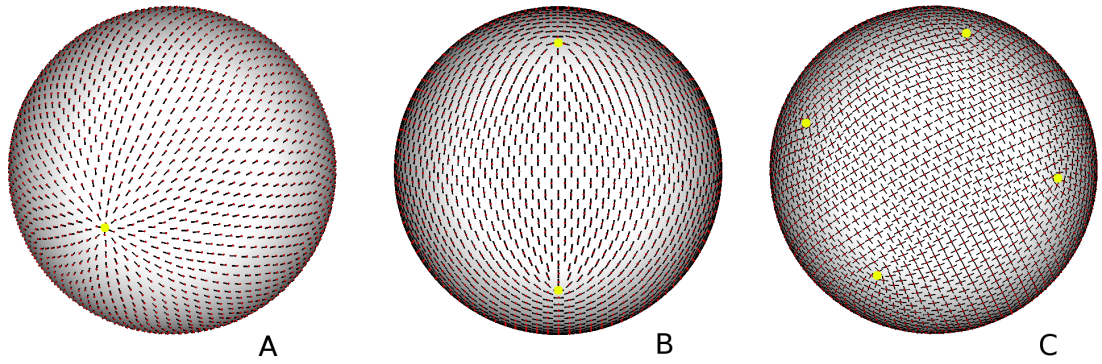


FIG. D.8: *Results of our global smoothing approach on a sphere using equality modulo 2π (A), π (B) and $\pi/2$ (C) respectively. Different first order singularities are observed in the three different cases (yellow dots). ρ is exceptionally set to 0.99 in these examples for demonstrating the smoothing effect. Note that fitting to the principal directions is meaningless in the spherical case since the sphere is isotropic anywhere.*

triangle integrals, solely depend on the triangles geometry, and are explicited in Equation F.14 in Chapter F

We optimize Equation D.19 for the unknowns $(\cos\alpha_i, \sin\alpha_i)$, using the same solution mechanism as in Section F.2.5. We introduce a penalty term that prevents the norm of the unknowns from vanishing. Similar to the penalty function in Section F.2.5, the penalty term evenly distributes the singularities over the surface, as shown in Figure D.8. Using the formulation given in Equations D.19 and D.20, equality between the α_i's is considered modulo 2π.

Given $(\cos\alpha_i, \sin\alpha_i)$ we recompute the direction field $\vec{K}_i = \cos\alpha_i\vec{H}_i + \sin\alpha_i\vec{H}_i \times \vec{N}_i$ and $\vec{K}_i^{\perp} = \vec{N}_i \times \vec{K}_i$.

Extending the functional to principal direction fields (2-symmetry)

To allow minimization of the above energy functional for a principal direction field, which is 2-symmetry direction field, it is possible to enforce equality modulo π by solving for intermediary variables $\tilde{\alpha}_i = 2\alpha_i$. In practice, this simply means dividing all the α_i^0's and the β_i's by 2, minimizing Equation D.19, and multiplying the α_i's by 2.

Extending the functional to "cross fields" (4-symmetry)

One can also smooth the direction fields with an equality modulo of $\pi/2$ to have a "smoother" field thanks to more degrees of symmetry. A 4-symmetry direction field is also called "cross field" (used for illustrating surfaces using cross-hatching [HZ00] and guiding texture

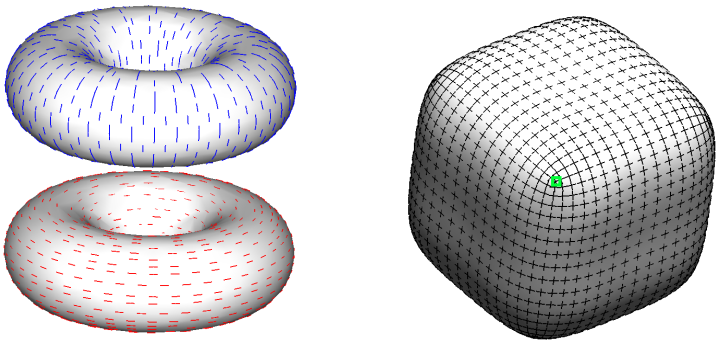


FIG. D.9: *Left : A torus, which is well illustrated by the two principal directions ; right : a cube, which is better illustrated by a cross field, obtained by cross-hatching. Note that, the cross field cannot be separated into two "smooth and continuous" fields. The green point is a singularity of the cross field.*

synthesis [WL01]). For certain surfaces, cross fields are better candidates for the guidance direction fields. For instance, a cube (see Figure D.9).

Similar to the 2-symmetry case, the energy functional is adapted to cross fields by solving for intermediary variables $\tilde{\alpha}_i = 4\alpha_i$. For the surface shown in see Figure D.9-Right, the smoothed cross field shows natural distribution of singular points of the field which is a nice property for being a guidance direction field for our periodic global parameterization.

D.5 Results and Discussions

Our smoothing method has some similarity with the direction field preprocessing described in [WL01]. The main difference is that our formulation with periodic variables allows the use of efficient numerical solvers. In addition, and thanks to our *global* formulation, the numerical solver evenly distributes the singular points over the surface (which is not observed with common *local* relaxation procedures). Due to this evenly distribution, we observed that one can only find first order singularities with the corresponding symmetry of the direction of the direction field. For instance, one finds only singularities with indices ± 1, $\pm 1/2$ and $\pm 1/4$ in a 1,2,4-symmetry direction field respectively (Figure D.8).

In this thesis, we use this method generate high-quality guidance direction fields to steer our periodic global parameterization (Chapter F). We have observed that generally, if a principal direction field is used as the guidance fields, it is more desirable to be smoothed as a single cross field by using equality modulo $\pi/2$. This is quite normal since the quad control mesh can

be thought of as a tiling process of 4-symmetric squares.

Smoothing principal direction fields usually gives reasonably good guidance direction fields with well-placed set of singularities found automatically. It is typically ideal for surfaces with small *local* geometry variations. However, for surfaces with high local geometry variations, the smoothing method may not work quite well since one cannot control the distribution of singularities. Some insignificant singularities due to local topologies cannot be smoothed out due to the high geometry variations. For surfaces like this, one needs more control on the singularities of the field. One of the possible solutions is by designing the *topology* and *geometry* of the direction fields, which is the topic that we are going to discuss in the next chapter.

D.6 Conclusion

Direction fields corresponding to the two principal directions of the surface are good candidates for guidance direction fields for many computer graphics and geometry processing algorithms. The two principal directions of a surface are direction fields corresponding to the eigenvectors of the two largest eigenvalues of the curvature tensor respectively. In this chapter, we have given a concise definition of the curvature tensor on the surface, and reviewed some methods to estimate the curvature tensor on polygonal meshes. The curvature tensor estimated on a surface can be visualized as ellipsoids, or preferably as two separated vector fields corresponding to the two principal directions and curvatures at a point. For the latter type of visualization, we have reviewed two families of methods which are Line Integral Convolution and placement of streamlines, and explained how we visualize principal direction fields in this thesis.

The estimated principal direction fields (2-symmetry) are to be used as guidance fields for our periodic global parameterization. We have explained the necessity of relaxing the estimated direction fields due to isotropic regions and insignificant local topologies. Existing local and global relaxation methods of principal direction fields are reviewed. Then, we have presented our global relaxation method which uses a simpler energy functional than the existing global approach and hence achieves a faster convergence. Moreover, we have discussed the advantages of relaxing a 2-symmetry principal direction field with 4-symmetry in order to achieve even "smoother" guidance fields for the control mesh extraction purpose.

Finally, we have discussed the limitation of our relaxation approach, i.e. it is difficult to smooth out some insignificant singularities on surfaces with high geometric variation, e.g. the hair region of Michelangelo's David. This drawback leads to the investigation of a guidance

field designing approach which is going to be detailed in the next chapter. With this design approach, one has direct control of the topology and geometry of the guidance field through specifying a placement of a set of singularities of the field and a set of geometric constraints respectively.

Annexe E

Designing N-symmetry Direction Fields on Surfaces of Arbitrary Genus

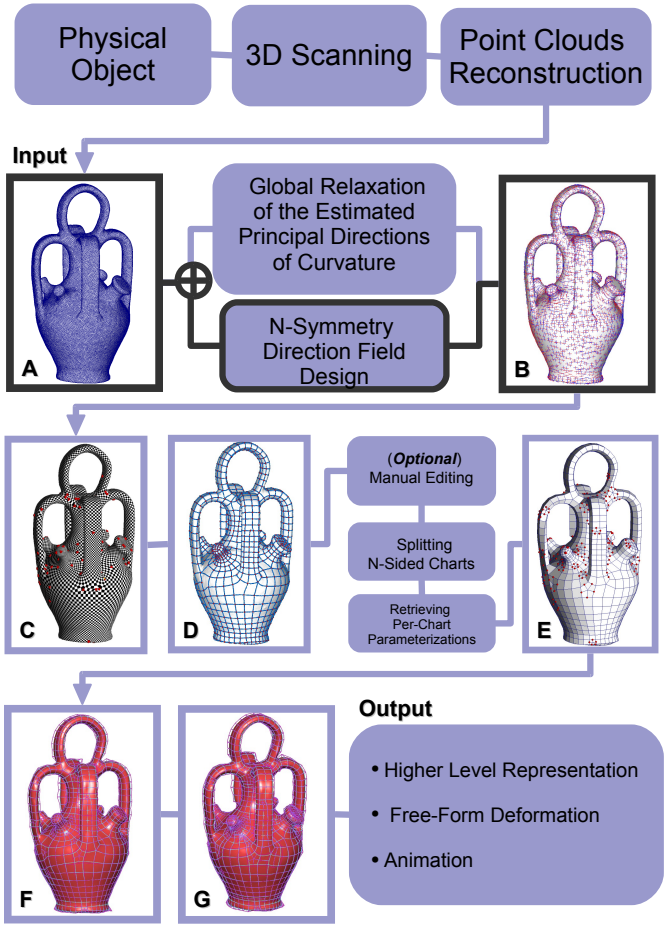


FIG. E.1: *Producing guidance direction fields by designing.*

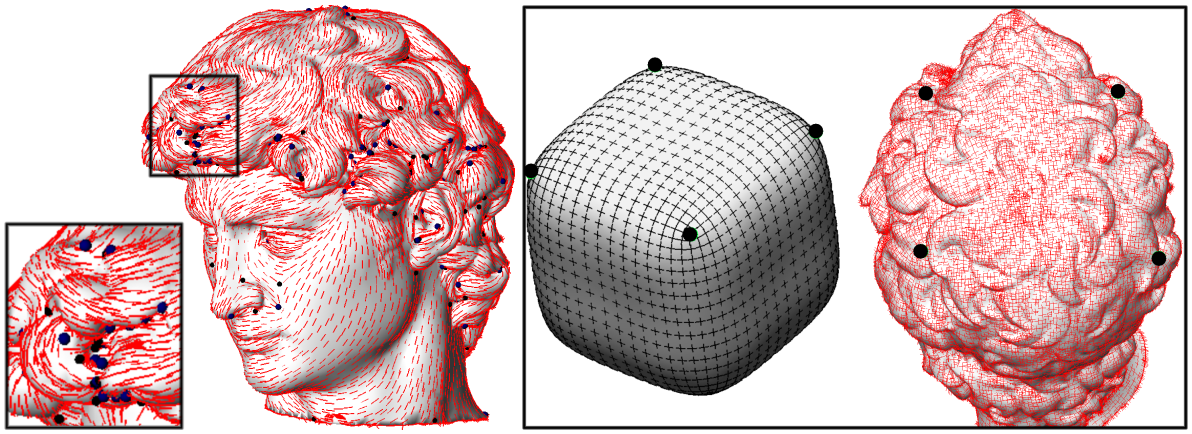


FIG. E.2: *Left : a smoothed principal direction field (2-symmetry) on an input mesh with high geometric variation ; Right : a 4-symmetry direction field with user-defined set of 4 singularities with index 1/4.*

E.1 Introduction

In the previous chapter, we have introduced a method to produce anisotropic guidance direction fields by smoothing principal direction fields of the surface. This method usually gives reasonably good guidance direction fields with well-placed set of singularities. It is typically ideal for surfaces with small *local* geometry variations. Nevertheless, for surfaces with high local geometric variations, the smoothing method may not work well since one cannot control the distribution of singularities. For instance, considering the head of David's statue (from the Standford Digital Michelangelo project [LPC+00]), not the whole set of singularities obtained by smoothing has global significance. Some insignificant singularities due to local topologies cannot be smoothed out due to the high variations, for instance, in the hair region (E.2-Left).

By considering that the global geometry of the head is similar to a unit cube, the set of singularities that one desires may be similar to the one shown in Figure E.2-Right, which is a 4-symmetry direction field with only four singularities of index 1/4 in the hair region. To obtain such desired direction field, one needs a *direct* control on the singularities of the field rather than just smoothing them.

Before proceeding further, we give a brief definition of the singularities of a vector field $\vec{v}$ defined on a surface S. A singularity is a point $p \in S$ such that $\vec{v}(p) = \vec{0}$. Singularities can be of various types, according to how the vector field winds around the singularity. This property is referred to as *index* in vector field topology (see Section E.2.6 and [Tri02]). Figure E.3 shows three singularities of index 1/4, 1/2 and 1 respectively. Note that on a surface of genus g, Poincaré-Hopf Index Theorem states that the indices of all singularities sum to $2-2g$.

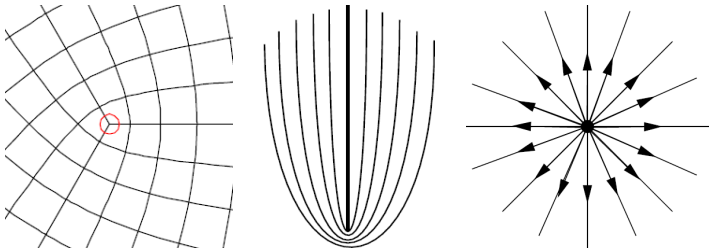


FIG. E.3: *Left : a singularity of index 1/4 ; Middle : a singularity of index 1/2 ; Right : a singuarity of index 1.*

In this chapter, the problem solved by our algorithm can be formalized as follows : given a triangulated surface S of genus g, and a subset of vertices $V_s = \{v_i\} \in V$ with desired indices I_i respectively (such that $\sum I_i = 2 - 2g$), our algorithm generates a vector field such that the index of v_i is equal to I_i and that provably does not contain any other singularity. Note that the so-generated vector field is of unit norm, i.e. a direction field.

Classical direction field design is much studied, e.g. [TSH00, The02, ZMT04]. There are also works, for instance, [ZHT05] that present 2-symmetry direction fields (or tensor fields) design methods. However, the existing methods do not provably control singularities. As for designing 4-symmetry or higher symmetry direction fields on arbitrary surfaces, such method still does not exist in the literature to the best of our knowledge. This is due to the lack of a formal definition of N-symmetry and due to the lack of some mathematical tools that link the topology of the direction field to the indices of singularities and hence the Poincaré-Hopf Index Theorem. The problem of designing 4-symmetry direction fields was first mentioned in the Computer Graphics field by Hertzmann and Zorin in their non-photorealistic paper [HZ00], which uses 4-symmetry direction fields to generate cross-hatching effects on surfaces.

To define such a N-symmetry direction field designing mechanism, we study the underlying topological structure. We capture the topological structure of an N-symmetry direction field $\vec{d}$ by defining the notion of *turning number* (see Sections E.2.5 and E.2.6).

We show that this topological quantity characterizes the singularities in the direction field by computing the turning number of a cycle around a singular point. The turning number is related to the definition of the *index* in vector field topology with a simple equation (Equation E.7).

With this understanding of continuous direction fields topology, we introduce the notion of *period jump* and use it to build a novel discrete direction field representation (see Section E.3). Using this discrete direction field representation, we introduce a new direction field designing mechanism. Our designing mechanism is a two-pass algorithm that produces direction fields

defined over a triangular mesh with a user-specified direction field topology and geometry.

Contributions

- We introduce the notion of *N-symmetry direction field*, that generalizes direction fields. We generalize definitions of *turning number* and *index* to characterize the singularities of a direction field (see Sections E.2.5 and E.2.6) ;
- We provide a simple proof of an analog of the Poincaré-Hopf theorem, implying that the indices of the singularities of a N-symmetry direction field defined on a manifold surface S sum to its Euler characteristic $\chi(S) = 2-2g$, where g is the genus of S (see Appendix H.1.1) ;
- We propose a discrete representation of N-symmetry direction fields for triangulated surfaces. The values of the field are defined on the facets of the surface. In addition, a one-form called the period jump is attached to the edges of the dual represents the variations of direction between two adjacent facets and enables representing singularities of arbitrary indices (see Section E.3) ;
- We introduce an algorithm for N-symmetry direction field design. From a user-defined set of singularities and an optional set of points with fixed directions, our algorithm constructs a smooth direction field. If the indices of the user-defined singularities sum to $2-2g$, the constructed field has no other singularity (see Section E.5.2).

Previous Work

Direction fields on surfaces are ubiquitous in Computer Graphics. The main applications of direction fields defined on surfaces use the so-constructed direction fields to steer the placement and orientation of elements over the surface. Those elements can be of various nature, depending on the application domain :

- **non-photorealistic rendering :** In [HZ00], the constructed direction field is used to place strokes for a non-photorealistic rendering application. The method they use to define and smooth a direction field (or cross-field) shares some common points with ours : the variables used to represent the directions are angles measured relative to a given arbitrary direction field. This paper follows the possibilities of future work and mathematical investigations suggested at the end of their paper.
- **texture synthesis :** In *lapped textures* [PFH00] and *texture particles* [DMLC02] methods, a direction field is used to control the orientations and sizes of texture patches distributed

over the object. In [PFH00], to generate a smooth direction field, they used radial basis functions, with geodesic distances computed over the surface. The method presented in [Tur01] operates at texel level, by using the direction field to steer the anisotropy of a texture synthesizer. The direction smoothing procedure they used is inspired by [GGSC96], which is based on a multi-resolution Laplacian smoother. A similar procedure is described in [OHB01], with the addition of non-linear weights that preserve important direction field discontinuities. In [WL01], 2,4-symmetry direction fields are used to steer synthesizing using 2,4-symmetry texture samples.

– **anisotropic remeshing :** In [ACSD+03], a method is proposed to generate quadrangles aligned with two orthogonal direction fields, obtained by smoothing an estimation of the curvature tensor. The refinement presented in [MK04b] operates without a global parameterization and uses, in a certain sense, an explicit version of Ohtake et al.'s non-linear weights [OHB01] to preserve important features.

In the specific case of a "cross-field" $\vec{v}_1, \vec{v}_2 = R_{\vec{n}}(\vec{v}_1, \pi/2)$, where $\vec{v}_1$ and $\vec{v}_2$ can be swapped, most of the smoothing algorithms mentioned above use a *local* relaxation procedure, updating values $\vec{v}_1(p_i)$ and $\vec{v}_2(p_i)$ *vertex by vertex*. During these computations, $\vec{v}_1(p_i)$ will be influenced either by $\vec{v}_1(p_j)$ or $\vec{v}_2(p_j)$. The algorithm chooses the direction nearest to $\vec{v}_1(p_i)$. As a consequence, singularities may appear without control, and the convergence is slow (see Section D.4). In contrast, based on a combined topological analysis of both the surface and the direction field, we derive a *global* formulation of the problem (quadratic form), yielding a more efficient optimization procedures (conjugate gradient), which computes directly an optimum solution.

Some other applications aim at using the so-constructed direction field to analyze the shape of the surface. For instance, fair Morse functions [NGH04] can be used to extract the topological structure of a surface. This structure is computed from the Morse complex of a smooth harmonic function, with user-controllable number and configuration of singularities. The gradient of the harmonic function is a direction field (with the same singular points as the harmonic functions). It was used in [ZG04] to steer a texture generation method. Similarly, in [GY03] a pair of holomorphic functions are computed. These two approaches share some common points with our approach, in particular, the ability of controlling singularities. The main difference is that in the two methods mentioned above, the direction field is defined to be the gradient of a scalar field (hence it is necessarily curl-free). In contrast, we consider a wider class of direction fields, not necessarily curl-free. Moreover, we can represent a wider class of singularities, with *arbitrary* indices.

Some more recent papers directly address problems related with direction field processing. For instance, in [PP02] and [TLHD03] a procedure for computing the Hodge decomposition of a direction field is described. This decomposition isolates some features of the field, and makes it possible to filter or to enhance them. In [TSH03], a method is presented to simplify the topology of symmetric, second order 2D direction fields. A complete toolkit for interactive direction field design is presented in [ZMT04], and then generalized to tensors in [ZHT05]. The tensor generalization uses a one-to-one mapping between the tensor field and a direction field, permitting to reuse the algorithms (e.g. singular points cancellation).

Our work also shares some similarities with Zhang et al.'s direction and tensor design, since it generates a smooth direction field from a user-defined set of singularities. Our main result is a general formulation (N-symmetry direction fields). The specific case $N = 1$ corresponds to direction field design, $N = 2$ corresponds to tensor field design, and $N = 4$ corresponds to cross-fields. To the best of our knowledge, our work is the first one to give a mathematical characterization (Poincaré-Hopf theorem) for cross-fields.

Finally, [WWT+06] use subdivision schemes to define bases for discrete differential 0- and 2-forms. This allows smooth vector fields to be generated on meshes of arbitrary topology. However, the only types of singularities allowed are source and sink, which are much less general than the singularities we allow in this work.

E.2 Direction Fields on Surfaces with Borders

This section presents the fundamental tools for studying direction fields defined on surfaces with borders, and especially to study their topology. Topology is the study of properties which are invariant by continuous deformations (without cutting or gluing anything), called homotopies. In other words, topology tries to answer the question : under what condition two objects are homotopic (i.e. can be continuously transformed one into another) ? For oriented surfaces with borders, the answer is that they need to have the same genus g (number of handles) and number of borders b. In other terms, if S_1 and S_2 are two surfaces with borders, we have $S_1 \equiv S_2 \Leftrightarrow g(S_1) = g(S_2)$ and $b(S_1) = b(S_2)$, where $\equiv$ denotes the homotopic equivalence. What is even more interesting is the structure of the set of homotopy classes (classes of all objects with same topology). For oriented surfaces with borders again, this set is isomorphic to $\mathbb{N}^2$, since any pair of non negative integers (g,b) can be associated to the class of all surfaces with genus g and b borders. For this reason, the couple (g,b) is referred to as the *topological degrees of freedom* (TDOF) of surfaces. This section addresses the same questions for N-symmetry direction

fields defined on a 2-manifold. In other words, we aim at studying the classes of homotopy of N-symmetry direction fields. More specifically, we want to exhibit the TDOF of these direction fields. To answer these questions, we will introduce the concept of turning numbers. As we will show, the TDOF of a direction field are the turning numbers of a homology basis, such that these turning numbers hold all the field topology. Moreover, turning numbers generalize singularities which alone do not control all of the field topology.

The intuitive idea behind turning numbers is that when following a cycle, a N-symmetry direction might do an arbitrary number of N^{th} of turns before coming back to its original direction. Imagine you are traveling on earth along a cycle with a compass giving you the north direction, then you can count the number of turns of the compass while following the cycle. If you turned around a tree, you will get 1 turn, but if you followed the equator or followed an 8, you will get 0 turn. We can do the same on any surface, and with any direction field defined on it, and we call this quantity the "turning number" of the field along the cycle (it depends on both the cycle and the direction field). We show that turning numbers capture the direction field topology, because two direction fields are homotopic iff they have the same turning numbers along the $2g+b-1$ cycles of a homology basis (a basis for cycles on a 2 manifold). This shows that homotopy classes of direction fields are isomorphic to $\mathbb{Z}^{2g+b-1}$, or in other words that the topology of a direction field is entirely defined by $2g+b-1$ integers. These $2g+b-1$ are the TDOF of direction fields.

In this section, we will first provide the reader with the definitions of the objects that we will be handling throughout this paper : surfaces (Section 2.1), cycles (Section 2.2) and direction fields (Section 2.3). Then we define the curvature for cycles and direction fields. We use these curvatures to define formally turning numbers, and exhibit their fundamental properties. We finally explain how turning numbers relate to field singularities.

E.2.1 Surfaces with Borders

In this paper we call surface (or 2-manifold) S a topological space where each point has a neighborhood homeomorphic to the plane or half plane. hence it can be cut into a finite number of topological disks. The surface S considered here is compact, connected and oriented, so that each point has a unique unit normal vector $\vec{n}$. S is homeomorphic to a sphere with b borders and g handles attached, for some (non-negative) integers b called the *number of borders* and g called the *genus* of S. As they define the topology of S, we call g and b the two *topological degrees of freedom* (TDOF) of S.

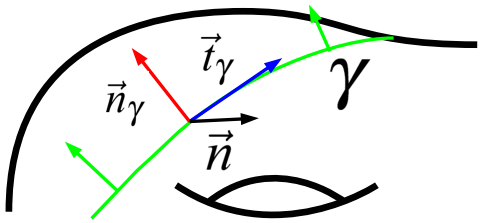


FIG. E.4: *A Darboux frame on a cycle (green) consists of the tangent $\vec{t}_\gamma$ (red), conormal $\vec{n}_\gamma$ (blue) and normal $\vec{n}$ (black)*

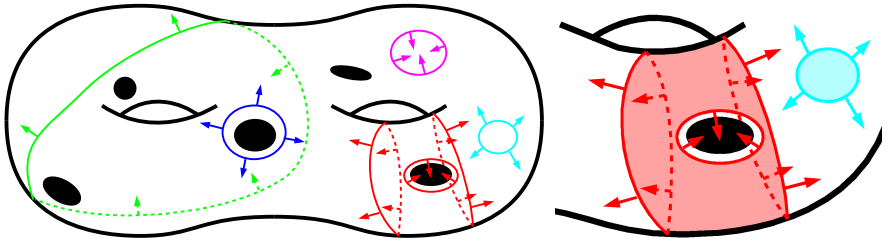


FIG. E.5: ***Left*** *: Cycles on a surface with borders. Red, light blue and purple cycles are boundaries whereas green and blue cycles are not. Only the light blue cycle is contractible.* ***Right*** *: A submanifold of S (in light color) and its boundary with conormal pointing outwards.*

E.2.2 Cycles on S

A *cycle* γ on S is an oriented 1-manifold without borders ($\partial\gamma = 0$) embedded in S ($\gamma \subset S$). We call $\mathscr{C}(S)$ the set of all cycles on S. Because γ is oriented, it has a unique tangent vector $\vec{t}_\gamma$ in each point which is also tangent to S. Using this tangent vector, along with the surface normal $\vec{n}$, we can define a unique unit conormal vector $\vec{n}_\gamma = \vec{n} \times \vec{t}_\gamma$ on the cycle, which ensures that $(\vec{t}_\gamma, \vec{n}_\gamma, \vec{n})$ form a natural local orthonormal basis called the *Darboux frame* (see Figure E.4 right).

Notice that cycles are not necessarily connected, so the term "set of cycles" would be more appropriate (but heavier in the redaction). We define the following notions on cycles (see Figure E.5 left) :

- The *reversal* $-\gamma$ of a cycle γ is the cycle with opposite orientation : $\vec{n}_{-\gamma} = -\vec{n}_\gamma$. We use the notations $\gamma_0 + \gamma_1$ and $\gamma_0 - \gamma_1$ in place of $\gamma_0 \cup \gamma_1$ and $\gamma_0 \cup -\gamma_1$ as it is more practical to handle unions of cycles with various orientations.
- We call ∂ the boundary operator, such that ∂S is the subset of points of S with neighborhood homeomorphic to the half plane. This subset is a cycle for which we can choose an orientation by requiring its conormal to point outwards S.

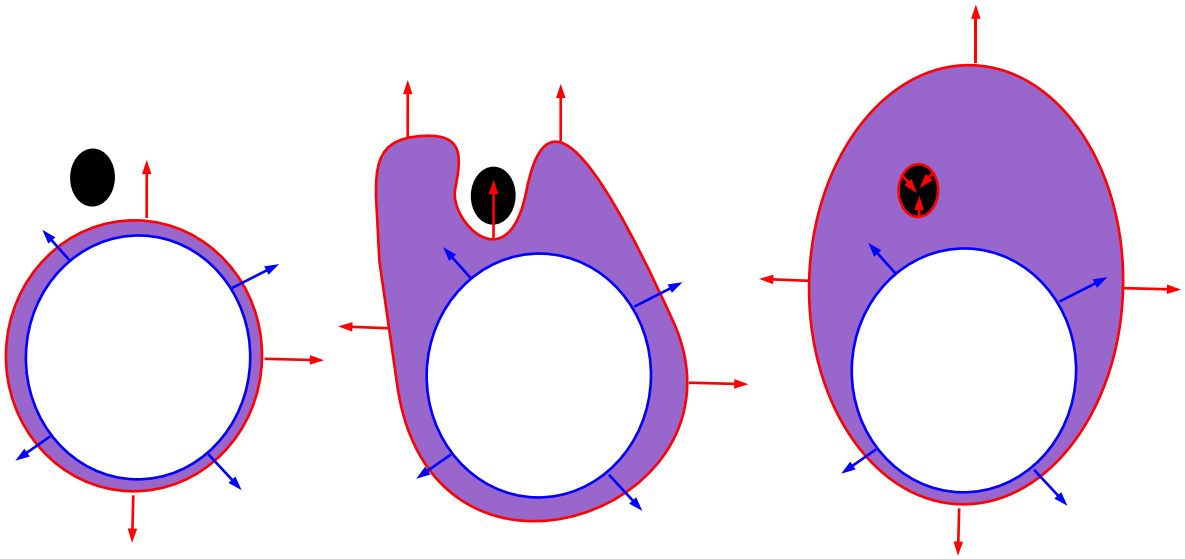


FIG. E.6: *The red cycle is homologic to the blue one because their difference is the border of the purple submanifold. Note that homologic cycles might have a different number of connected components.*

- A cycle γ is a *boundary* if there exist a sub-manifold S of S such that $\gamma = \partial S$. In general, the boundary ∂S is not connected, and we call *borders* of S the connected components of its boundary ∂S. In other words, the boundary of S is the collection of its borders.
- Two cycles are *homotopic* if one can be continuously deformed into another. More formally, $\gamma_0 \equiv \gamma_1 \Leftrightarrow$ there exists a continuous function $\Gamma : [0,1] \rightarrow \mathscr{C}(S)$ such that $\Gamma(0) = \gamma_0$, $\Gamma(1) = \gamma_1$ ($\mathscr{C}(S)$ is the set of all cycles on S).
- If S is a topological disk then $\gamma = \partial S$ is called contractible. The definition of contractibility for loops is that a loop is contractible if it is homotopic to a null loop. Our definition of contractibility for cycles adds a notion of orientation to this definition, as the reversal of a contractible boundary is not necessarily contractible.
- Two cycles γ_0 and γ_1 are *homologic* iff $\gamma_0 - \gamma_1$ is a boundary (see Figure E.6)
- $H(S) = \{\gamma_i^H\}_{i=1..n}$ is called a homology basis on S if they are independent (a basis cycle is not homologic to a sum of other basis cycles), and if any cycle on S is homologic to a formal sum of basis cycles : $\forall \gamma \in \mathscr{C}(S) \exists a \in \mathbb{Z}^n$ such that $\gamma - \sum_{i=1}^{n} a_i \gamma_i^H$ is a boundary.

We use homology for cycles instead of homotopy because it is more flexible : homology allows a cycle to split into two or two cycles to fusion into one (see Figure E.6), whereas homotopy does not. Moreover, homotopy basis become very complex on surfaces with high genus and number of borders, because all base loops need to go through a common basepoint.

E.2.3 Direction Field

A unit tangent vector $\vec{u}$ on S is a vector satisfying $||\vec{u}|| = 1$ and $\vec{u} \cdot \vec{n} = 0$. We call N-symmetry direction on S a set of N unit tangent vectors on S invariant by rotation of $2\pi/N$ around the normal. Hence, based on a unit tangent vector $\vec{u}_0$ we can build a N-symmetry direction $\vec{d} = \{\vec{u}_k = R_{\vec{n}}(\vec{u}_0, 2k\pi/N)\}$ where $R_{\vec{n}}$ is the rotation around $\vec{n}$. We call direction field $\vec{d}$ on S a mapping which associates a N-symmetry direction $\vec{d}(P)$ to each point $P \in S$, and $\mathscr{D}_N(S)$ the set of N-symmetry direction fields on S. In the following, we will omit the term N-symmetry for brevity.

Two direction fields $\vec{d}_0$ and $\vec{d}_1$ are called homotopic if there exists a continuous function $\Gamma : [0,1] \to \mathscr{D}_N(S)$ such that $\Gamma(0) = \vec{d}_0$, $\Gamma(1) = \vec{d}_1$. Homotopy classes of direction fields can be characterized by what we call the *turning numbers* of the field along some cycles. They correspond intuitively to the number of times the direction turns in a local Darboux frame while moving along the cycle. We are now going to define the curvature of both cycles and direction fields, which will be required for a rigorous definition of the turning number.

E.2.4 Curvature

The curvature of a cycle expresses the angular variation of its tangent. If we call s the arclength on a cycle γ, we can define the curvature of γ using the decomposition :

$$\frac{\partial \vec{t}_\gamma}{\partial s} = \kappa_\gamma \vec{n}_\gamma + \kappa_S \vec{n} \qquad \kappa_S = \frac{\partial \vec{t}_\gamma}{\partial s} \cdot \vec{n} \qquad \kappa_\gamma = \frac{\partial \vec{t}_\gamma}{\partial s} \cdot \vec{n}_\gamma \tag{E.1}$$

- κ_S measures the normal curvature of S in direction $\vec{t}_\gamma$.
- κ_γ measures the curvature of γ in the tangent plane of S.
- $\partial \vec{t}_\gamma / \partial s \cdot \vec{t}_\gamma = 0$ by derivation of $\vec{t}_\gamma \cdot \vec{t}_\gamma = 1$ ($\vec{t}_\gamma$ is a unit vector)

We can similarly define the curvature $\kappa_{\vec{d}}(\vec{t}_\gamma)$ for the direction field $\vec{d} = \{\vec{u}_k\}$ in direction $\vec{t}_\gamma$ as :

$$\kappa_{\vec{d}}(\vec{t}_\gamma) = \frac{\partial \vec{u}_k}{\partial s} \cdot \vec{u}_k^\perp \tag{E.2}$$

where $\vec{u}_k^\perp = \vec{n} \times \vec{u}_k$ is a unit vector orthogonal to $\vec{u}_k$ in the tangent plane, such that $(\vec{u}_k, \vec{u}_k^\perp, \vec{n})$ is an orthonormal basis (it is the Darboux frame of the streamlines of $\vec{u}_k$). The curvature $\kappa_{\vec{d}}$ is the same for all $k \in \mathbb{N}$ so it can be called the curvature of $\vec{d}$ in direction $\vec{t}_\gamma$. In what follows $\kappa_{\vec{d}}$ will always refer to the curvature of the field in the direction of integration.

These curvatures now allow us to give the definition of the turning numbers, which will be used to characterize homotopy classes of direction fields.

E.2.5 Turning Number

The turning numbers of a direction field along a cycle corresponds to the number of rotations of the field along this cycle. We will show that the turning numbers are characteristic of homotopy classes of direction fields, hence of their topology.

For a direction field $\vec{d}$ and a cycle γ on S_h, we call the *turning number* of $\vec{d}$ along γ the quantity :

$$T_{\vec{d}}(\gamma) = \frac{1}{2\pi}\oint_{\gamma} d\theta(t) = \frac{1}{2\pi}\oint_{\gamma}(\kappa_{\vec{d}} - \kappa_{\gamma})ds \tag{E.3}$$

where $d\theta$ is the variation of the angle between the direction $\vec{d}$ and the tangent to the cycle $\vec{t}_{\gamma}$. As the angle θ itself is defined modulo $2\pi/N$, the turning number is necessarily a multiple of $1/N$ corresponding to the number of N^{th} turns done by the field along the cycle (see Figure E.7). Note that since the turning number has discrete value, and that its definition makes it continuous with respect to continuous transforms of both the field and the cycle, it is invariant by homotopy.

Turning numbers have two fundamental properties (see Appendix A for a proof) which make them useful for studying direction field topology :

Theorem E.2.1 (Boundary turning number)**.** *Let S be a surface (2-manifold with borders) embedded in* $\mathbb{R}^3$*, then :*

$$T_{\vec{d}}(\partial S) + \chi(S) = 0 \tag{E.4}$$

where $\chi(S) = 2 - 2g(S) - b(S)$ *is the Euler characteristic of S.*

Theorem E.2.1 (Boundary turning number) is equivalent to the Poincaré Hopf theorem with a proper definition for the index of a singularity, which will be developed in next subsection. It links the topology of a direction field with the topology of the manifold on which it is defined. As it is true for any submanifold $S \subset S$, it will give much insight on the relations between turning numbers of cycles of S, especially on homologic cycles as their difference forms a boundary.

Theorem E.2.2 (Topological equivalence)**.** *Two direction fields defined on a surface S are homotopic iff they have the same turning numbers along the cycles of any homology basis* $H(S)$ *of S :*

$$\vec{d}_1 \equiv \vec{d}_2 \Leftrightarrow \forall \gamma \in H(S), T_{\vec{d}_1}(\gamma) = T_{\vec{d}_2}(\gamma)$$

Theorem E.2.2 shows that direction fields, as cycles have homology classes isomorphic to $\mathbb{Z}^{2g+b-1}$. This comes from the fact that the $2g$ so-called generators of the surface and $b-1$ surface borders define a homology basis, and that all basis have the same number of cycles. Hence a direction field has $2g+b-1 = 1-\chi(S)$ topological degrees of freedom (TDOF) on a genus g surface with b borders.

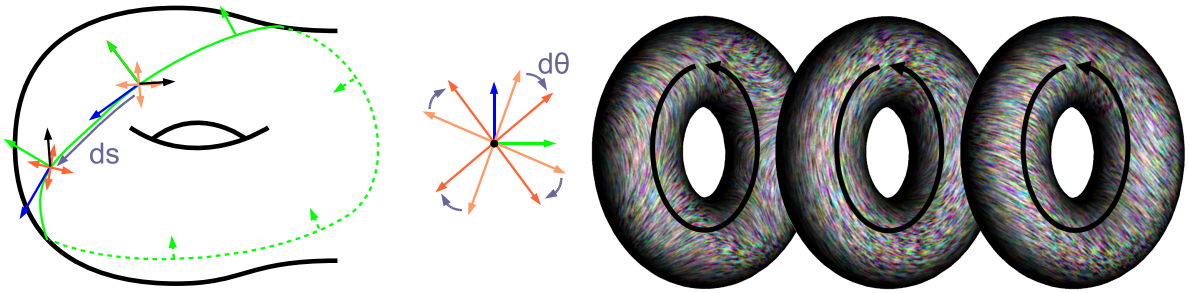


FIG. E.7: ***Left*** *: The turning number of a direction field along a cycle corresponds to the rotation of the field in the local Darboux frame* ***Right*** *: The turning number associated to a generator defines topology that cannot be captured by singularity indices. The difference of topology of theses direction fields without any singular points is defined by the turning numbers of the generator (black cycle) which are respectively -1, 0 and 1*

Both theorems are proved for completeness in Appendix A. With these two theorems, one can exhibit, understand and control all the TDOF of a direction field.

E.2.6 Singularities

Let $\vec{v}:\mathbb{R}^2\rightarrow\mathbb{R}^2$ be a vector field. It is usually assumed that the zero set of $\vec{v}:\{x\in\mathbb{R}^2|\vec{v}(x)=0\}$ consists of a finite number of distinct points P_i which are called the singularities of $\vec{v}$. The singularities can then be classified by their index :

$$I_{\vec{d}}(P_i)=\frac{1}{2\pi}\int_{\partial\Omega(P_i)}d\theta \tag{E.5}$$

where $\Omega(P_i)$ is a small neighborhood of P containing no other singularities (zeros) of the vector field, and θ is the angle formed by the vector field and a reference vector. The singularity index is necessarily an integer which equals 1 for sources, vortices and sinks, -1 for saddles. An index of 0 corresponds to a degenerate singularity, which means the corresponding singularity can be removed without creating another singularity.

We can now transpose the notions of singularity and index to direction fields simply by asking the direction field obtained by normalizing a vector field to have the same singularities and indices. As singularities are zeros, the vector field cannot be normalized there, hence the normalized field is undefined at singularities. Therefore, we identify singularities of a direction field with “holes” in its domain of definition, and singularity index keeps the same definition (E.5) as it depends only on the vector direction.

For N-symmetry vector fields, indices can still be defined by (E.5), but they are now a multiple of $1/N$ because θ is defined modulo $2\pi/N$.

This 2D definition for direction field singularity index cannot be directly extended to surfaces embedded in $\mathbb{R}^3$ because it lacks a reference vector to define θ. However, it can be shown that in $\mathbb{R}^2$:

$$T_{\vec{d}}(\partial\Omega(P)) = I_{\vec{d}}(P) - 1 \tag{E.6}$$

As the turning number can be extended to surfaces in $\mathbb{R}^3$ because the reference vector is the tangent to the cycle, this allows us to extend the definition of the index of a singularity on a surface in $\mathbb{R}^3$ to :

$$I_{\vec{d}}(P) = 1 + T_{\vec{d}}(\partial\Omega(P)) \tag{E.7}$$

With this definition for the index, we can generalize the Poincaré-Hopf theorem to N-symmetry direction fields :

Theorem E.2.3 (Poincaré-Hopf)**.** *The sum of singularity indices on a closed surface S equals its Euler characteristic :*

$$\sum_{i=1}^{b} I_i = \chi(S) = 2 - 2g$$

Proof : Notice first that this theorem is on surfaces without borders, so the number of borders is absent in the Euler characteristic. Let us call P_i the point of index I_i and $S_h = S\backslash\{\Omega(P_i)\}$ (surface with borders in place of singularities), on which the field is continuous because it does not contain the singularities. Applying the boundary relation (E.4) to S_h yields :

$$T(\partial S_h) = T(\partial S) - \sum_{i=1}^{b} T(\partial\Omega(P_i)) = -\sum_{i=1}^{b}(I_i - 1) = -\chi(S_h) = -\chi(S) + b \quad \Rightarrow \quad \sum_{i=1}^{b} I_i = \chi(S)\square$$

The concept of replacing singularities with holes allows singularities to be handled as borders. Moreover borders are topological objects which are very well understood through cycle homology. In what follows, we will use again this idea of replacing singularities with holes, and to characterize the singularity by the behavior of the field along the border of the hole.

Notice that singularities do not hold all the topological degrees of freedom, since a homology basis also contains generators of the surface, which do not enclose any singularity (they are non-disconnecting). This justifies why we need the concept of turning number to capture the direction field topology, and not only the singularity indices.

The next step is to use the concept of turning number to build a representation for direction fields allowing us to have direct control over its topology (and especially singularity indices) through the turning numbers.

E.3 Discrete Direction Field

We now see how to build a discrete representation for a direction field on a mesh which allows direction field topology to be controlled explicitly. The major difficulty of an explicit control is to settle the ambiguity inherent to the interpolation of cyclic variables (e.g. angles). In the following, we will explain this problem and introduce our solution to it.

E.3.1 Discretization and Period Jumps

Let $M =< V,E,F >$ be a connected orientable mesh of genus g with b borders. As we want to allow any vertex $v_i \in V$ to hold a singularity, direction fields that we consider are defined on $M_h = M\backslash\{\Omega(v_i)\}$, which are meshes from which we have conceptually removed small neighborhoods around vertices. The first step of the discretization of a direction field is to sample it at the center of each facet. This is done by choosing a reference direction $\vec{d_0}$ in each triangle (for instance one of the oriented edges of the triangle), and defining its direction $\vec{d}$ as the angle θ measured from from $\vec{d_0}$.

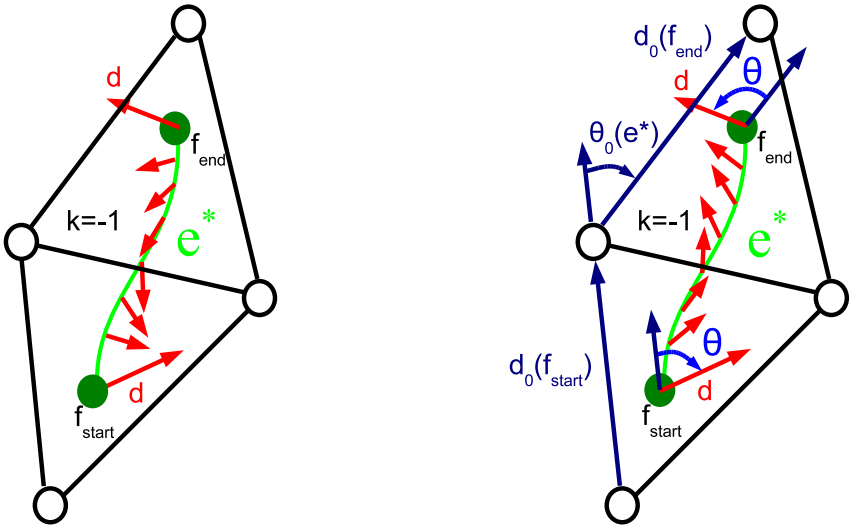


FIG. E.8: *$k(e^*)$ solves the ambiguity for equal angles with the reference direction $\vec{d_0}$ at endpoints of a dual edge e^*.*

This representation however leaves an ambiguity in the behavior of the fields between samples (at facet barycenters). In particular, the direction variation of the field along each dual edge :

$$\Delta\vec{d}(e^*) = \int_{e^*} \kappa_{\vec{d}} ds$$

can take different values :

$$\Delta\vec{d}(e^*) = \theta(f_{end}(e^*)) - \theta(f_{start}(e^*)) + \theta_0(e^*) + \frac{2\pi k(e^*)}{N} \tag{E.8}$$

where f_{start} and f_{end} are the starting and ending points of e^*, $\theta_0(e^*)$ is the angle between $\vec{d}_0(f_{end})$ and $\vec{d}_0(f_{start})$ measured after flattening the pair of triangles along the common edge e. $k(e^*)$ is an integer which we call period jump, and which indicates the way the direction varies between the two given directions along e^* (see Figure E.8). Note that this information is not held by the angle on each facet. To ensure that $\Delta\vec{d}(e^*)$ is invertible when the orientation is changed (by inverting f_{start} and f_{end}), $\theta_0(e^*)$ is defined in $(-\pi, \pi]$ (this avoids the ambiguity in the case when $\vec{d}_0(f_{end})$ and $\vec{d}_0(f_{start})$ are exactly opposite). Note that angle θ is defined in $\mathbb{R}$, which makes it much easier when used in an optimization, as cyclic variables are known to be harder to optimize.

We can now use the direction variations $\Delta\vec{d}(e^*)$ to compute the turning numbers along any cycles of G^*, and we will show that they can be controlled by the period jumps.

E.3.2 Turning Number in Discrete Setting

Using the definition of the turning number (E.3) and of the direction variation (E.8), we derive an expression for the turning number of a cycle γ in the discrete setting. Using the expression above, we obtain :

$$T_{\vec{d}}(\gamma) = \frac{1}{2\pi}\left(\sum_{e^*\in\gamma} \Delta\vec{d}(e^*) - \oint_\gamma \kappa_\gamma ds\right) = T_0(\gamma) + \sum_{e^*\in\gamma} \frac{k(e^*)}{N} \tag{E.9}$$

where $T_0(\gamma)$ is defined as :

$$T_0(\gamma) = \frac{1}{2\pi}\left(\sum_{e^*\in\gamma} \theta_0(e^*) - \oint_\gamma \kappa_\gamma ds\right) \tag{E.10}$$

which is independent of the field because all angles cancel out by summing (E.8) along the cycle.

This allows us to compute the index of the direction field at a vertex v :

$$I_{\vec{d}}(v) = I_0(v) + \sum_{e^*\in\partial v^*} \frac{k(e^*)}{N} \tag{E.11}$$

where $I_0(v)$ is a geometric quantity independent of the field and defined by :

$$I_0(v) = 1 + T_0(\partial v^*) = \frac{1}{2\pi}\left(\sum_{e^*\in\partial v^*} \theta_0(e^*) + A_d(v)\right) \tag{E.12}$$

where $A_d(v)$ is the angle defect at v given by $A_d(v) = 2\pi - \oint_\gamma \kappa_\gamma ds$.

E.4 Visualization of Discrete Direction Fields

Before proceeding further to our direction field design algorithm, let us first explain how to visualize direction fields represented using our discrete representation. Essentially, the discrete representation can be thought of as a *facet-based* "encoding" of direction fields on piecewise linear surfaces. The direction field is described in *polar coordinates* over each facet, with a triangle edge being chosen as the reference to define the angle. A *period jump* is associated to each edge of the triangulation to remove the ambiguity when interpolating the direction of the direction field between two facets that share an edge. At vertices, the directions are undefined. In this way, we completely encode a continuous direction field by using only discrete values.

During the visualization process, our goal is to restore a continuous field from these discrete values. Therefore, we need an interpolation scheme to define the direction field over the whole surface. In this section, we explain how to interpolate a direction field on the mesh M in three steps of increasing dimension. We already know the directions at facet centers (0D). Then we interpolate linearly between two facet centers along the dual edges (1D). Finally, from the dual edges we interpolate the direction to the whole mesh (2D). This interpolation may be thought of as a variant of the "side-vertex" interpolation scheme [Nie79]. As the reader will see in the following, in our case, the value along the side is interpolated linearly while it is constant (identical to the side value) along a side-vertex path.

E.4.1 Step 1 : 0D

From now on, we call $[v_0...v_i]$ the simplex (edge or triangle) based on points $v_0...v_i$ and $G(s)$ the gravity center of a simplex, s. The first step of the interpolation is to define the direction $\vec{d}(f)$ at the gravity center $G(f)$ of each facet $f \in F$. $\vec{d}(f)$ at $G(f)$ is defined by its angle $\theta(f) \in \mathbb{R}$, which is the angle between $\vec{d}(f)$ and the facet reference vector, $\vec{d_0}(f)$. Notice that $\vec{d}(f)$ and $\theta(f)$ are values given at $G(f)$. They will be interpolated, so they are not constant on the whole facet, but we omit the G for the sake of brevity since it is not ambiguous. Moreover, it corresponds to the implementation where a $\theta(f)$ is stored for each facet.

E.4.2 Step 2 : 1D

The second step of the interpolation is to define the direction field on the edges of the barycentric dual of M (see Figure E.9). For an edge $e = [v_1v_2]$ between a pair of adjacent triangles

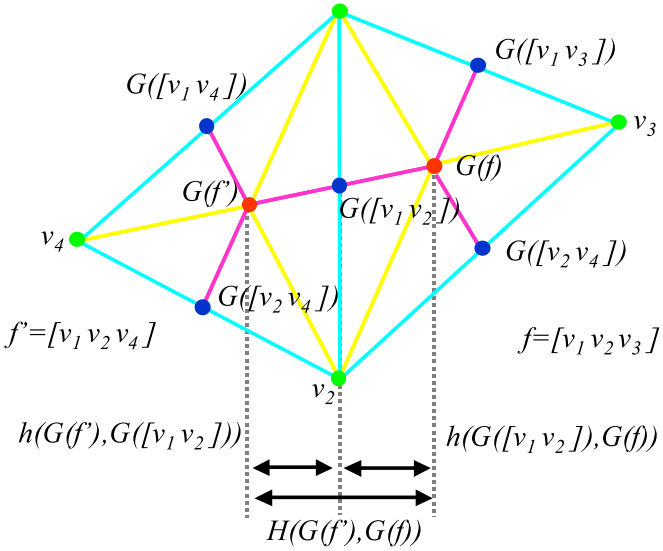


FIG. E.9: *Illustration on two adjacent (primal) triangles of a primal mesh (light blue) and its barycentric dual (magenta). Subdivision simplices are triangles based on a primal vertex (green), a dual vertex (red), and an edge middle (dark blue). Their edges are based on primal and dual edges, and edges between primal and dual vertices (yellow).*

$f = [v_1v_2v_3]$ and $f' = [v_2v_1v_4]$, we can geometrically define the barycentric dual edge :

$$e^* = [G(f)G([v_1v_2])] \cup [G([v_1v_2])G(f')] \tag{E.13}$$

The angular variation along e^* is given by Equation E.8. The main difficulty of this step is to split the angular variation along the two parts of the dual edge. We choose to split according to the height ratio above the common edge (see Figure E.9) :

$$\alpha(G(f), G([v_1v_2])) = \frac{h(G(f), G([v_1v_2]))}{H(G(f'), G(f))} =$$
$$\frac{||\overrightarrow{v_1G(f)} \times \overrightarrow{v_1v_2}||}{||\overrightarrow{G(f')G(f)} \times \overrightarrow{v_1v_2}||} = \frac{||\overrightarrow{v_1v_3} \times \overrightarrow{v_1v_2}||}{||\overrightarrow{v_4v_3} \times \overrightarrow{v_1v_2}||} \tag{E.14}$$

This gives a natural linear interpolation for $P \in [G(f)G([v_1v_2])]$ given in barycentric coordinates $P = (1-t)G(f) + tG([v_1v_2])$:

$$\vec{d}(P) = \theta(f) + \Delta\vec{d}(e^*)\alpha t \tag{E.15}$$

E.4.3 Step 3 : 2D

The third step is to interpolate the field over the whole mesh, by a piecewise defined interpolation on the subdivision simplices $S_{i,j,k}$ of the mesh M. The subdivision simplices are simply

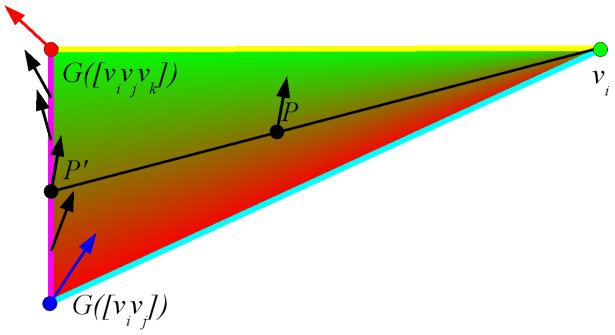


FIG. E.10: *Our "side-vertex" interpolation over a subdivision simplex : we obtain the interpolation value at P as the value at its projection P′ on the (magenta) dual edge.*

defined as the triangles

$$S_{i,j,k} = [v_i G([v_i v_j]) G([v_i v_j v_k])] \quad \forall (i,j,k) | [v_i v_j v_k] \in F \tag{E.16}$$

Notice that because of the possible permutations for (i,j,k), there are six subdivision simplices per facet of the mesh (see Figure E.9). On each subdivision simplex of a facet $f = [v_i v_j v_k]$, the field is known on the edge $[G([v_i v_j]) G(f)]$. As for the remainder of the triangle, it is interpolated such that the vector is constant along each segment between a point of this edge and the primal vertex v_i. Another way of saying that is that we obtain the vector at a point P as the vector at the intersection P' between (v_i, P) and $[G([v_i v_j]) G(f)]$ (see Figure E.10). This interpolation may be thought of as a variant of the "side-vertex" interpolation scheme [Nie79]. In our case, the side value is interpolated linearly while along a side-vertex path, the value is constant, which equals to the side value. This gives a very simple expression in barycentric coordinates. If we write :

$$P = (1-t')v_i + t'((1-t)G(f) + tG([v_i v_j]))$$

then we have :

$$P' = (1-t)G(f) + tG([v_i v_j]) \tag{E.17}$$

and we obtain the direction by the interpolation described in Equations E.15. The simplicity of the above interpolation makes it straightforward to be implemented on a GPU (see the next section).

E.4.4 The GPU-based Visualization Algorithm

Now the direction is defined at every point on the surface (except at vertices). In this section, we present an algorithm to visualize our direction fields. We propose a GPU-accelerated

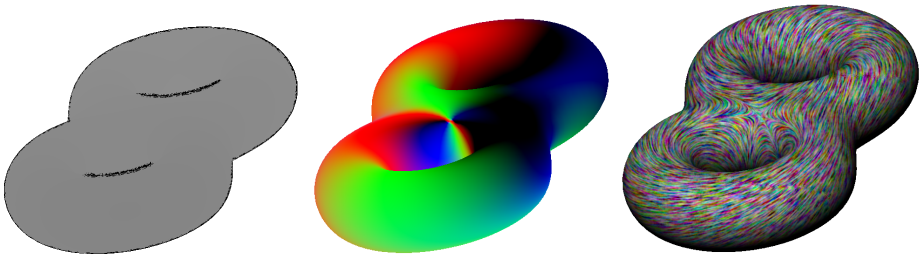


FIG. E.11: *The three passes of the LIC on GPU.Left : the geometric discontinuities identified ; Middle : direction on the surface encoded in colors ; Right : result of the LIC process on a direction field with a singularity of index -2.*

LIC-based approach working in image space, which is inspired by the Image Space Advection method proposed in [LSH03]. The algorithm is done in three passes (see Figure E.11). It is basically the same as the one presented in Section D.3.1 for visualizing principal direction fields except that in the current case, the direction at each point of the surface is obtain using the interpolation scheme introduced in the previous section. More specifically, we modify the Pass 2 of the GPU implementation as follows. We associate (1,0,0), (0,1,0) and (0,0,1) to the vertices of the subdivision simplex $G(f)$, $G([v_iv_j])$ and v_i respectively, where $f = [v_iv_jv_k]$, as their barycentric coordinates $(1-\lambda_1-\lambda_2, \lambda_1, \lambda_2)$. The direction of a point in the subdivision simplex (except v_i, where it is undefined) is given by the angle of rotation, θ measured with respect to the base direction $\vec{d_0}(f)$. Using Equations E.17 and E.15, the θ value is given by $\theta(f) + \Delta\theta\alpha\lambda_1/(1-\lambda_2)$. Then, the obtained vector in object space is projected onto the image plane, which is then normalized.

E.5 N-Symmetry Direction Field Design

Now that we have introduced our discrete direction field representation, we will explain how to use it to our designing purpose. The basic idea of our designing mechanism is to find the right values of the k's and θ's over the whole triangular mesh that correspond to the desired direction field topology and geometry :

The design procedure consists of the following two steps :

1. *Topologic step*

   In this step, the user specifies a set of singular vertices with the desired indices of values of integer multiples 1/N. The sum of indices of the singular vertices should be equal to the Euler Characteristic of the surface (due to Poincaré-Hopf Index Theorem). Using

Equation **??**, we obtain the correct values of period jumps k's, and hence the topology of the direction field is fixed.

2. *Geometric step*
   Once we have fixed the topology of the direction field, the user specifies the geometry of the direction field by means of directional constraints at a subset of facets. In this way, the direction field can be made adapted to the anisotropy of the triangular mesh.

   The values θ's are found such that the so-created direction field has the "smoothest" geometry with the fixed topology and the specified directional constraints. This is done through a simple quadratic minimization procedure with the energy functional (Equation E.23).

E.5.1 Topologic Step : Constraining Singularity Indices

In this step, we choose the period jumps $k(e^*)$ on each edge, which will set the topology of the represented direction field. The period jumps should be computed such that such that the singularities and their indices are exactly the ones defined by a user.

Given a triangulated surface S of genus g, the user specifies a subset of vertices $V_s = \{v_i\} \in V$ with desired indices I_i respectively (such that $\sum I_i = 2-2g$, to avoid unwanted singularities). Therefore, every vertex in the triangulation has a constrained index I_c, i.e. $I_c = I_i$ for user specified vertices, and $I_c = 0$ otherwise.

However, the period jumps control all TDOF, including those not corresponding to singularities, such as turning numbers of generators (see Figure E.7-Right). We choose to let these TDOF free and find a better solution in the geometric step. As a result the algorithm will not compute explicitly all period jumps, but find some free period jumps and express all the other period jumps as a combination of the free ones.

Problem Setting

The problem solved in this step can be formalized as follows :

Given constrained singularity indices $I_c(v_i)$ at each interior vertex v_i of the mesh, find the period jumps k on each dual edge such that :

$$I_{\vec{d}}(v_i) = I_c(v_i) \tag{E.18}$$

Using (E.11), the above expression can be rewritten as :

$$\sum_{e^* \in \partial v_i^*} \frac{k(e^*)}{N} = I_c(v_i) - I_0(v_i) = \Delta I(v_i) \tag{E.19}$$

Edge Classification

As in the continuous setting, the homology basis $H(M)$ of M consists of $2g$ generators and $b-1$ borders (we will discuss later the case $b=0$). As the turning numbers of the cycles in $H(M)$ do not correspond to singularities, we will leave the turning numbers of these cycles free. We remind the reader that our discrete direction field is defined on M_h, which is the mesh M with holes at every interior vertices v_i allowing a singularity to occur at v_i. Hence M_h has a homology basis $H(M_h) = H(M) \cup \{\partial v_i^*\}$. Each removed interior vertex v_i adds one cycle around it to the homology basis, the most natural being the dual cell boundary ∂v_i^*. The turning numbers along these additional cycles in $H(M_h)$ correspond to singularities located on interior vertices, so we want to control them explicitly.

To achieve this kind of explicit control, one has to find the values of the period jumps which are a discrete one-form [DKT06] (on the dual edges) that is defined by the sums of the period jumps along cycles of the homology basis $H(M_h)$. The sum of the period jumps over an cycle is either constrained by the index of a vertex (treated as a border) or a free variable k_i^{free} (one of the $2g$ generators of the surface of genus g). To find this discrete one-form, one should solve a linear system whose variables are the period jumps. The constraints are the linear relations between the sum of the period jumps around a vertex and its desired index (E.19), and sums of period jumps along generators equal to k_i^{free}. However, period jumps are discrete variables which are tricky to handle. Therefore, in the following, we propose a greedy algorithm called the *Zipping*, which benefits from the structure of the mesh. After the Zipping, all the period jumps are determined as either a fixed value or as function of $2g$ free period jumps k_{free}. Note that although these $2g$ k_{free} affect the topology of the direction field, they are linearly independent to each other and affect the direction field around a vertex only geometrically. Therefore, their best values are left to be solved together with the θ in the geometric step.

The Zipping progressively classifies the dual edges in three sets :

1. The set E_0^* of *null* edges, whose associated period jumps we can set to 0 without constraining any turning number. Hence we have $k(e_0^*) = 0$
2. The set E_{dep}^* of *dependent* edges, whose associated period jumps constrain the turning numbers of cycles ∂v_i^* around a single vertex v_i, hence constraining the index of a the singularity at v_i.

3. The set E^*_{free} of *free* edges, whose associated period jumps correspond to the turning number of a cycle homologic to a cycle in $H(M)$, hence corresponding to a TDOF but not to a singularity. We choose the name *free* because these other TDOF will be left free in an optimization.

The Zipping is a greedy algorithm which makes this iterative classification, and computes an expression of the period jumps of edges in E^*_{dep} as a function of the free period jumps of edges in E^*_{free}, such that the indices have their constrained value.

1. Fill E^*_0 : While it is possible, add to E^*_0 edges that do not close any cycle. The result of this step is a spanning tree of the dual graph G^*.
2. Fill E^*_{dep} : While it is possible, add to E^*_{dep} edges that close dual cells (cycles around a single primal vertex) and computes the corresponding period jumps such that they satisfy (E.19).
3. Add an edge to E^*_{free} : any remaining edge necessarily closes a cycle (else it would have been added to E^*_0) which does not enclose a single vertex (else it would have been added to E^*_{dep}). Hence this cycle is homologic to a border or to a generator of the mesh, which turning numbers we want to keep free, so any such edge can be added to E^*_{free}.
4. After freeing an edge, it becomes possible again to find edges that close dual cells, such that steps 2 and 3 may be iterated until no edge remain. The period jumps computed in step 2 will depend on the period jumps of the edges that have been freed in step 3 (see Figure E.13).

Please see Algorithm 2 for the whole Zipping algorithm.

Computing Period Jumps

In all generality, all period jumps $k(e^*)$ can be expressed through an integer k_0, and a vector of integers $\mathbf{c}$ of size $c = |E^*_{free}|$ such that :

$$k(e^*) = k_0(e^*) + \mathbf{c}(e^*) \cdot \mathbf{k}^{free} \tag{E.20}$$

where $\mathbf{k}^{free} = (k_1^{free}, ..., k_c^{free})$ is the vector of free period jumps. k_0 and $\mathbf{c}$ are both null on E^*_0, and $k_0(e^*_i) = 0$, $c_j(e^*_i) = \delta_{i,j}$ (1 if $i = j$ else 0) on E^*_{free}. For any edge in E^*_{dep} closing a cycle ∂v^*_i around vertex v_i, we ensure (E.19) is satisfied by setting :

$$k_0(e^*_{dep}) = \Delta I(v_i) - \sum_{e^* \in \partial v^*_i \setminus e^*_{dep}} k_0(e^*) \quad , \quad \mathbf{c}(e^*_{dep}) = - \sum_{e^* \in \partial v^*_i \setminus e^*_{dep}} \mathbf{c}(e^*) \tag{E.21}$$

Algorithm 2 Zipping algorithm (see Figure E.12)

Build a recovering tree E_0^* of G^* // *grow* black edges
$\forall e_0^* \in E_0^*$ set $k_0(e_0^*) \leftarrow 0$, $\mathbf{c}(e_0^*) \leftarrow 0$
$E_{dof}^* \leftarrow E^* \backslash E_0^* \quad V_{zip} \leftarrow V$
$i_{free} \leftarrow 0$ // Number of free variables found
while $E_{dof}^* \neq \emptyset$ **do**
  while $V_{zip} \neq \emptyset$ **do**
    Take $v_z \in V_{zip}$ and remove it from V_{zip}
    if $v_z \notin \partial M$ and $\exists!$ unset edge $e_z^* \in \partial v_z^*$ **then**
      // *zip* red (then blue) edges
      Move e_z^* from E_{dof}^* to E_{dep}^*
      Set $k_0(e_z^*) \leftarrow \Delta I(v_z) - \sum_{e^* \in \partial v_z^* \backslash e_z^*} k_0(e^*)$,
      Set $\mathbf{c}(e_z^*) \leftarrow -\sum_{e^* \in \partial v_z^* \backslash e_z^*} \mathbf{c}(e^*)$,
      Add the face opposite to v_z^* across e_z^* to V_{zip}
    end if
  end while
  // *free* green edge
  $i_{free} \leftarrow i_{free} + 1$
  Take $e_{free}^* \in E_{dof}^*$ and move it to E_{free}^*
  Set $k_0(e_{free}^*) \leftarrow 0$, $c_i(e_{free}^*) \leftarrow \delta_{i,i_{free}}$
  Add the 2 faces adjacent to e_{free}^* to V_{zip}
end while

Topological Issues

If M has no border, all singularities on the surface are vertex singularities, so their indices on the base field necessarily satisfy the Poincaré-Hopf theorem E.2.3 ($\sum_V I_{\vec{d}_0}(v) = 2-2g$) We have also :

$$\sum_{E^*} \frac{k(e^*)}{N} = \sum_V \Delta I(v) = \sum_V (I_c(v) - I_{\vec{d}_0}(v)) = 0$$

the $k(e^*)$ being taken in both direction for each edge. This means that on the last vertex whose index is set, we have :

$$\Delta I(v_{last}) = -\sum_{V \backslash v_{last}} \Delta I(v) \quad , \quad I_c(v_{last}) = 2-2g-\sum_{V \backslash v_{last}} I_c(v) \tag{E.22}$$

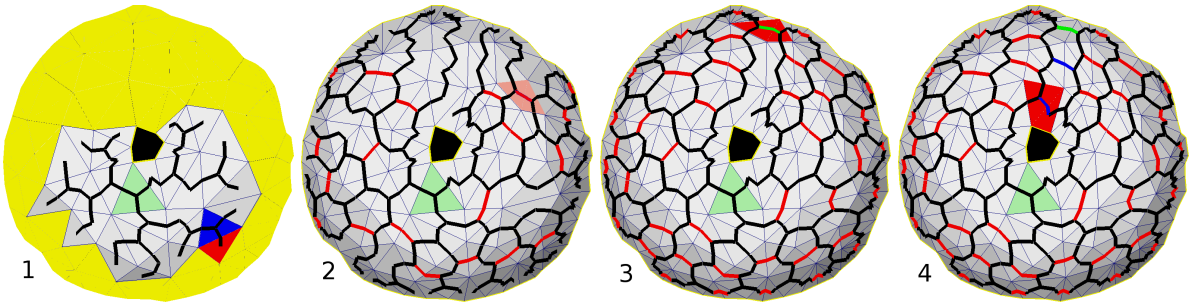


FIG. E.12: *Zipping : 1-**Grow** black edges (width first search) 2-**Zip** red edges 3-**Free** green edge 4-**Zip** blue edges (blue edges depend on the freed green edge)*

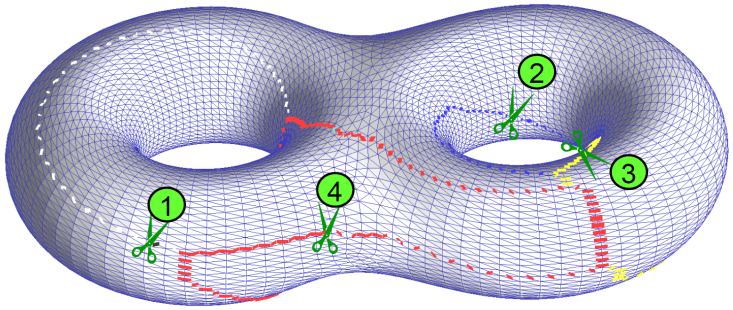


FIG. E.13: *A Genus g surface without borders requires* $2g$ *edges to be freed. The image shows in different colors the edges which period jumps effectively depend on free period jumps (scissors).*

Hence the index of v_{last} adapts to ensure the Poincaré-Hopf theorem. If the indices have been constrained such that :

$$\sum_V I_c(v) = 2-2g$$

then this last index will be 0. As the position of v^*_{last} depends on some choices made arbitrarily in the algorithm, it cannot be easily determined, hence it is highly recommended to run the algorithm with a constrained indices which sum up to $2-2g$ to avoid the apparition of a random (but necessary) singularity.

If M has borders, they are also handled by the Zipping, but the indices of corresponding borders are not constrained. In fact, we can also leave some vertices unconstrained by simply declaring them as border vertices before running the Zipping. However, in this case we cannot guarantee that no undesired singularity appear. A simple way to ensure that no singularity appear when M has borders, is to run the Zipping on M with borders triangulated. The index of a border on such a field will simply be given by the sum of the indices of the border vertices, so as before, no singularity will appear iff the sum of indices of constrained vertices (including border vertices) add up to $2-2g$.

E.5.2 Geometric Step : Discrete Direction Field Interpolation

Once we have fixed the topology of the direction field (i.e. all the period jumps k are determined as either a fixed value or as function of $2g$ free period jumps k_{free}), the user can specify the geometry of the direction field by means of directional constraints at a subset of facets. In this way, the direction field can be made adapted to the anisotropy of the triangular mesh.

Problem setting

In this step, we find the values θ and the $2g$ k_i^{free} such that the so-created direction field has the "smoothest" geometry with the fixed topology (last step) and the specified directional constraints. This is done through a simple quadratic minimization procedure as follows :

$$Min.\, E_{G^*}(\theta, k_{free}) = ||\Delta\vec{d}||^2 = \sum_{e^* \in E^*} \Delta\vec{d}(e^*)^2 = \sum_{e^* \in E^*} \left(\theta(f_{end}(e^*)) - \theta(f_{start}(e^*)) + \theta_0(e^*) + \frac{2\pi k(e^*)}{N} \right)^2 \tag{E.23}$$

subjected to the constraints :

- the period jumps k determined in the topologic step (a fixed value or as function of $2g$ free period jumps k_{free}) ;
- a constrained direction $\vec{d}_c(f)$ given on each facet of a subset $F_c \subset F$.

Notice that because $\kappa_{\vec{d}}$ is squared, the direction of integration does not matter.

Algorithm

The minimization problem is solved in two passes (see Algorithm 3) :

1. Minimize E_{G^*} with respect to θ and k_{free} by assuming k_{free} is continuous
2. Minimize with respect to θ only, with the k_{free} being set to their rounded value of the first pass

We use for both passes a standard formula to solve the problem of minimizing $([A_f, A_l][x_f, x_l] - B)^2$ where x_f are variables and x_l are set, the matrix A of the quadratic form is split accordingly into A_f, A_l :

$$x_f = (A_f^t A_f)^{-1} A_f^t (B - A_l x_l) \tag{E.24}$$

This method is not guaranteed to find the global minimum with respect to the discrete variables k, but offers good results in practice that satisfy all the constraints. For the continuous variables θ, if at least one directional constraint is set, A_f is of maximal rank, therefore, by Gramm's theorem, $A_f^t A_f$ is non-degenerate and our algorithm finds the unique minimum. We also noticed

Algorithm 3 Interpolation algorithm

1. Compute $\theta_0(e^*)$ as the angles between $\vec{d}_0(f_{end})$ and $\vec{d}_0(f_{start})$) for each dual edge.
2. Build the linear system $[A_f, A_l, C][\theta_f, \theta_l, k_{free}]^t = B$ corresponding to (E.23). Each line of the system corresponds to an edge e^* : $[A_f, A_l]$ contains $+1$ and -1 at the indices corresponding to the θ at the two extremities of e^*, C contains $\mathbf{c}(e^*)$ corresponding to k_{free}. B contains the $\kappa_{\vec{d}_0}(e^*) - k_0(e^*)$.
3. Pass 1 :
$$[\theta_f^1, k_{free}^1]^t = ([A_f, C]^t [A_f, C])^{-1} [A_f, C]^t (B - A_l \theta_l)$$
4. Pass 2 :
$$[\theta_f^2]^t = (A_f^t A_f)^{-1} A_f^t (B - [C, A_l][rnd(k_{free}^1), \theta_l]^t)$$
where *rnd* is the rounding to the nearest integer value.
5. Apply rotations θ_f^2 to $\vec{d}_0$ to get a direction $\vec{d}$ on each facet of M.

that we could improve the visual aspect of the direction field near the constrained directions by adding a Laplacian smoothing term to the energy (E.23).

E.6 Results

Our designing method is especially suitable when dealing with surfaces with high geometric variation, e.g. the hair region of the head of Michelangelo's David , where relaxation (see Section D.4) of direction fields fails to remove most of the insignificant singularities of the fields. In Figure E.14, we have shown a 4-symmetry guidance direction field designed using our method. Since the topology can be explicitly controlled, we are able to constrain the field such that there are only 4 singularities of index 1/4 in the hair region. The choice of the placement of singularities is based on the observation that the head is a cube-like object. Moreover, geometric constraints can also be imposed to obtain anisotropic direction fields. As illustrated in Figure E.14-Right, only few constraints are required to create the desired field.

Our direction field designing algorithm has also been tested on large models from the Stanford 3D Scanning Repository called the statue and Lucy. These two models are treated in less than 4 minutes, and smooth fields are obtained (see Figure E.15). Note that these two models have a complex topology (large genus g and number of borders b). The turning number around each border is used to counter the effect of each handle (a small handle is equivalent to a singular

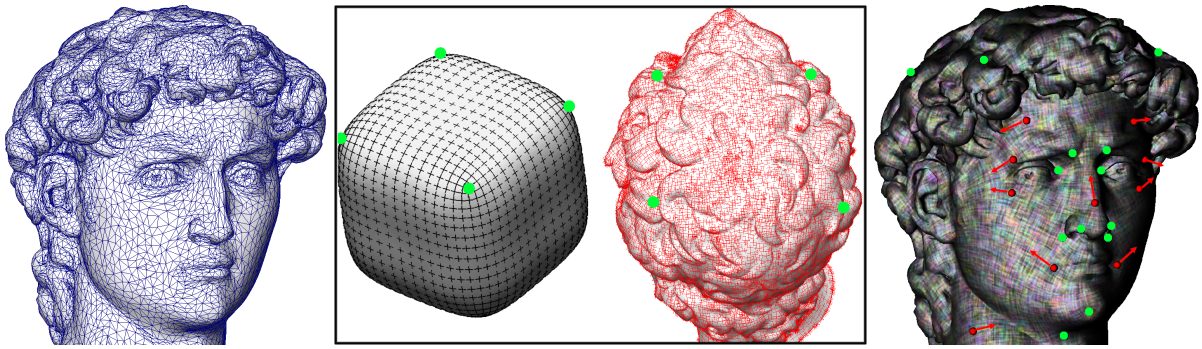


FIG. E.14: *Left : an input mesh with high geometric variation ; Middle : a placement of a set of 4 singularities with index 1/4 in the hair region. The green dots indicate singularities and the red crosses show the generated cross field ; Right : red arrows indicates geometric constraints.*

	Lucy	The statue
genus	45	13
#borders	47	9
#triangles	125000	300000
time	2min 47s	3min 36s

TAB. E.1: *Timings obtained on a Pentium IV 1.7Ghz*

point of index 2).

Moreover, The direction field designing can benefit from an existing direction fields to start with a good initial solution. Direction field smoothing algorithms (see Section D.4), are able to automatically place singular points. However, using these methods, globally insignificant singularities due to local geometry variations cannot always be smoothed out. An automatic fusion (e.g. clustering techniques) of these critical points gives us a nice starting point for setting the constraints. Figure E.17 shows a simplification of the head of the dragon dataset (from the Stanford 3D Scanning Repository) allowing us to remove meaningless singularities. Such a model can then be easily edited without taking into the large part of the direction field topology that is determined by the shape of the surface.

Furthermore, the interpolation algorithm also deals nicely with important constraints that can be applied on geometry (see the rotation constraints in Figure E.16) and on topology (see Figure E.7).

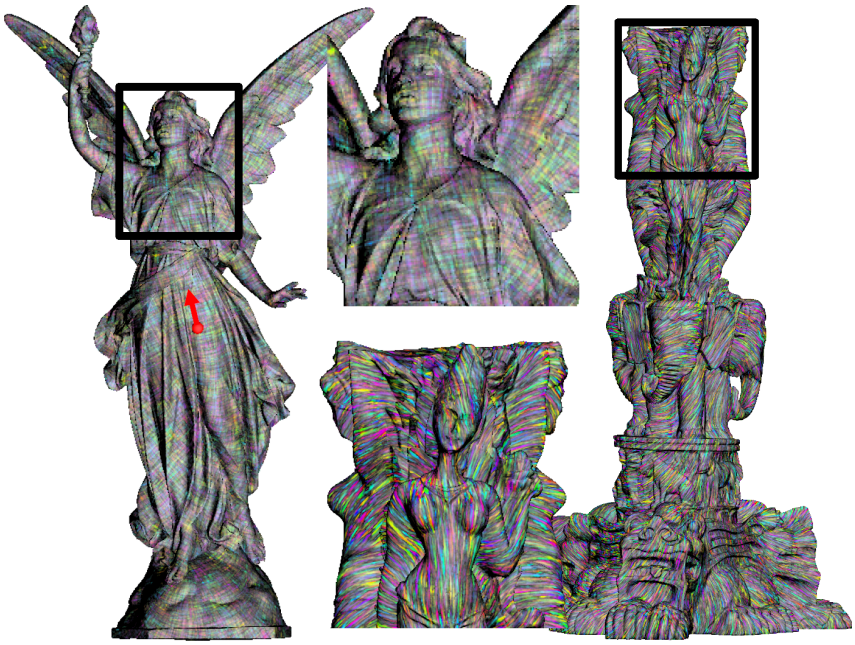


FIG. E.15: *Large models with many borders and high genus efficiently processed by our framework.*

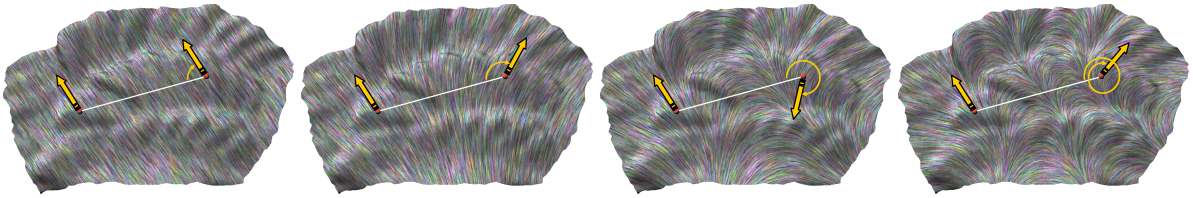


FIG. E.16: *Direction constraints are applied to a direction field. Notice that the direction is given by a rotation of the base field that can be greater than* 2π.

E.7 Conclusion

In this chapter, the goal is to design high quality guidance directions fields to steer our periodic global parameterization. We have first studied the underlying mathematically structure of N-symmetry direction fields, i.e. turning number. We have shown the linear relation between the index of a singularity and the turning number of a cycle around the singularity. Using the notion of turning number, we have provided a simple proof of Poincaré-Hopf index theorem. We have also shown that the singularities of a direction field do not hold all the topological degrees of freedom, i.e. the topology of the field is also governed by the turning number of the generators of the surface. Namely, the topology of the direction field is determined by the turning numbers of the element cycles of the homology basis of the surface.

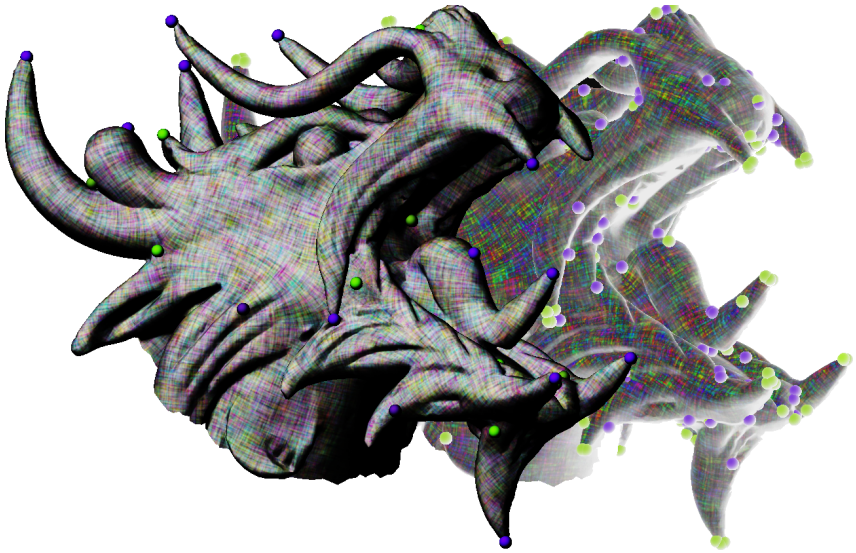


FIG. E.17: *The dragon in the background was smoothed by a classical algorithm. In the foreground : our algorithm only keeps significant singularities (balls).*

With the understanding in the continuous case, we have introduced the notion of the period jump to build a discrete representation of direction fields on triangular meshes. With this representation, the singularities can only occur at vertices of the mesh, and can have arbitrary index.

This discrete representation allows the topology and geometry of the direction field to be processed separately. For the topologic step, being given a placement of singularities with desired non-zero indices of multiples of $1/N$, one can constrain the indices of the vertices with desired values through the linear relation between the index and the sum of the period jumps along the dual facet of a vertex. We have proposed a greedy algorithm called the Zipping to compute the period jumps over the whole mesh that satisfy all the index constraints by benefiting from the mesh structure.

As for the geometric step, which is an interpolation process, being given a set of directional constraints on facets, we solve for the θ on each facet through a simple quadratic form, which gives the direction on each facet and hence the geometry of the direction field.

Using this designing mechanism, we have tested and discussed the results on several datasets. The designed direction fields will be used as guidance direction fields to steer our periodic global parameterization, which will be introduced in the next chapter.

Annexe F

Periodic Global Parameterization

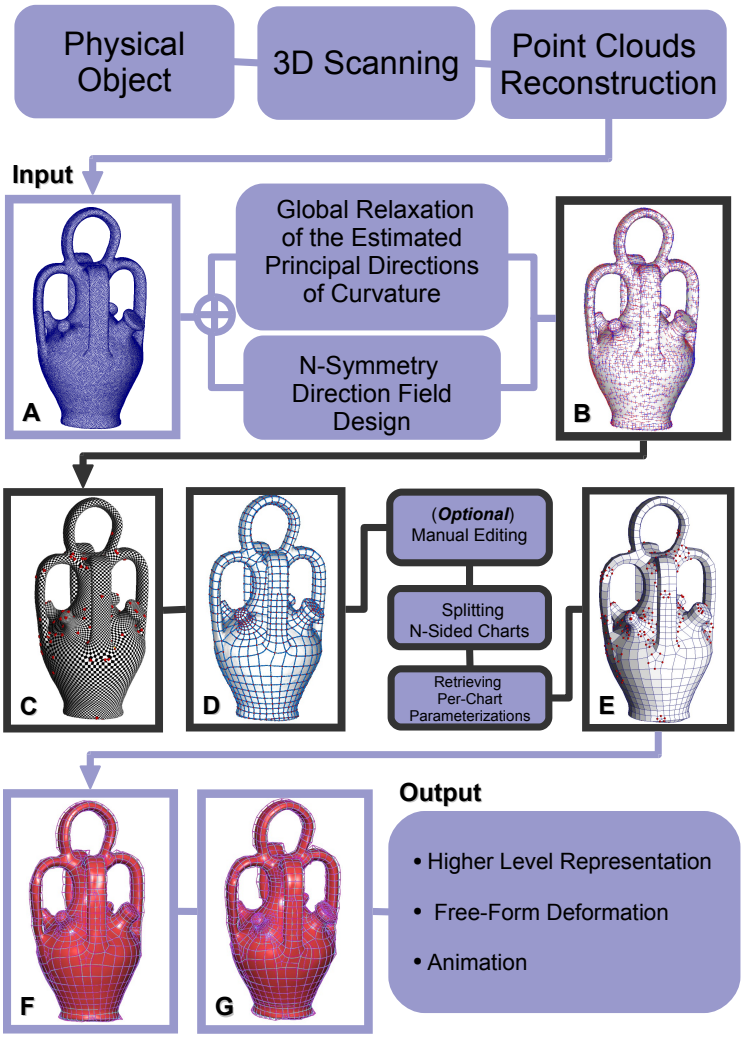


FIG. F.1: *Periodic Global Parameterization.*

F.1 Introduction

Once the guidance direction fields are computed, we now introduce a method called *periodic global parameterization* to generate a coordinate system of the object, aligned with the direction fields. The basic idea of our new method is to find a "geometry-meaningful" parameterization. In the subsequent steps, the iso-value lines of this parameterization will be extracted to define an initial control mesh. The advantage of our global parameterization based control mesh extraction method is threefold :

1. There is a fundamental difference between mesh and our target spline representation. A mesh is an enumerated sampled representation of the geometry, whereas a spline surface requires more structure, i.e. a spline needs a parameterization of the object. We fill in the gap between both representations by constructing an abstract representation of the object, i.e. a parameterized surface.
2. There are numerous way to parameterize a surface. In order to produce a "geometry-meaningful" parameterization that fulfills the anisotropy, the iso-value lines of the parameterization is guided by a pair of orthogonal anisotropy-adapted direction fields. Since the parameterization is adapted to the geometry, the iso-value lines define a *natural quadrilateral control mesh* of the surface, optimum from an approximation theory point of view [d'A00].
3. The parameterization is found by a *global* minimization of an energy functional using the trigonometric functions (hence periodic) of the actual parameters s and t of the surface (see Section F.2). Therefore, we do not encounter the remeshing drawbacks as in [ACSD+03] (unevenly-spaced edges of the control mesh, and open loops).

Before proceeding to the presentation of our method, we will give a quick review of the existing methods to extract a quadrilateral control mesh from a surface :

Today, the easiest way to create this anisotropic control mesh is to let the user manually design it, as often done in industrial packages [Cyb, Rap]. Thanks to the flexibility of manual drawing to adapt the anisotropy, it is widely used nowadays in the industry. This manual control mesh designing, implemented in a 3D modeler, is indeed quite similar to the digitization process by manual acquisition in the seventies (Figure F.2-Right). In practice, the user draws the control mesh onto a virtual representation of the object (Figure F.2-Left). In both cases, one designs the edges of the mesh by hand and tries to adapt them to the features of the object.

Although being more flexible, manual drawing is always time-consuming and tedious for surfaces with complex geometry and/or topology. Therefore, many automatic methods have

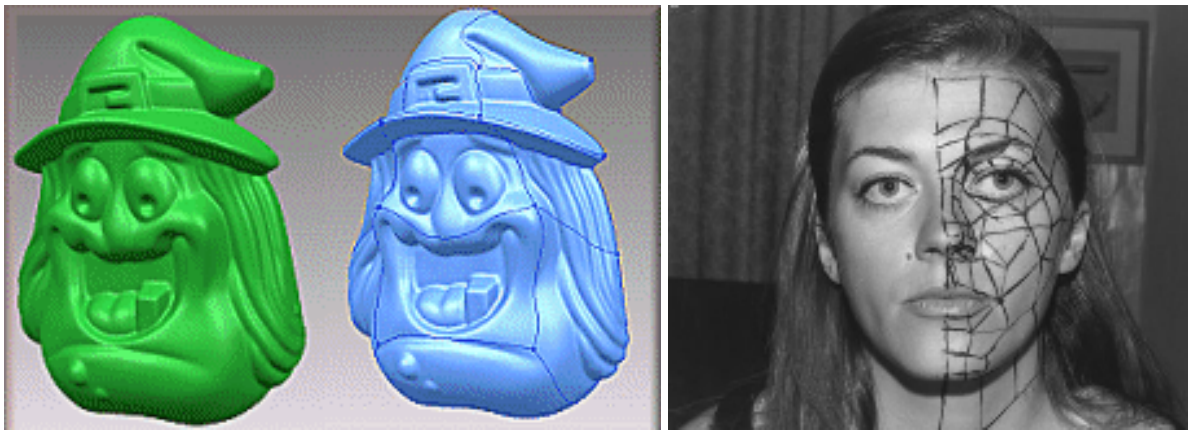


FIG. F.2: *Left : manual design of a CAGD/CAM control mesh [Rap] from a triangular mesh ; Right : digitization of a real object by manual acquisition (the image is taken from Gouraud's thesis [Gou71b]).*

been proposed, for instance, the pioneering work by Eck and Hoppe [EH96]. Unfortunately, using their automatic approach, unwanted oscillations are observed quite often on the spline surfaces built from the control meshes obtained by their method (Figure F.3-Left). It is due to the fact that the control meshes are not anisotropy-adapted. Recently, Alliez *et al.* [ACSD+03] proposed an anisotropy-adapted method to obtain control meshes by explicitly integrating curvature lines of the principal directions on the surface. Although it is anisotropy-adapted, their method does not give satisfactory results in term of remeshing quality due to two major drawbacks : uneven spacing of control mesh edges and open loops (Figure F.3-Right).

The remainder of this section is organized as follows : we will first give an introduction to globally smooth surface parameterization. Then, we will review some related work. Finally, we will give an overview of the algorithm of our periodic global parameterization.

F.1.1 Brief Introduction to Globally Smooth Surface Parameterization

A parameterization defines a correspondence between a surface mesh embedded in 3D and a simple 2D domain, referred to as the *parameter space*. In the general case, a parameterization is expected to be bijective, i.e., one-to-one.

Recent advances in geometry processing algorithms and computer graphics hardware have made possible the use of parameterized surface meshes as valid representations for resampling and remeshing, as well as for mapping complex signals such as texture, normal or light maps for efficient rendering purposes.

Common mesh parameterization methods are restricted to domains with simple topology,

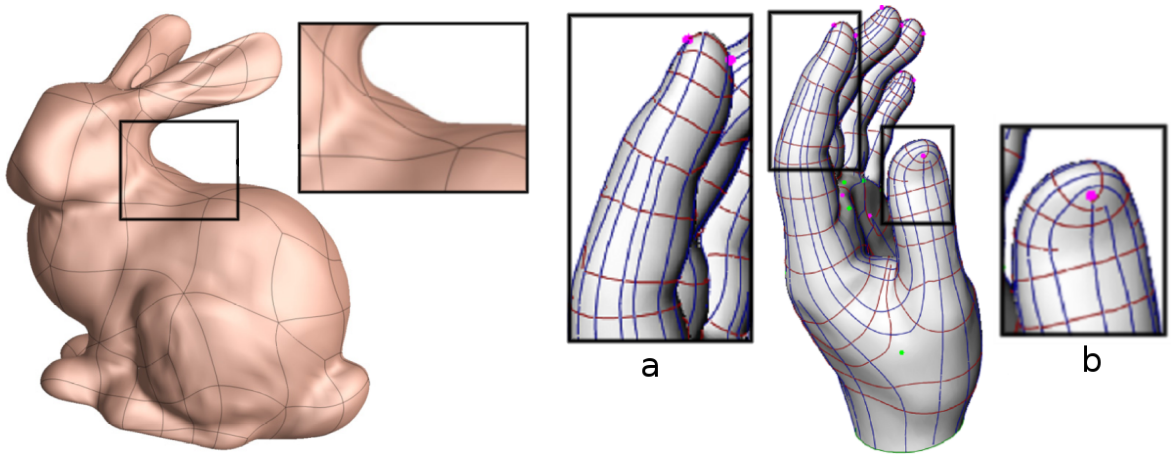


FIG. F.3: *Left : using Eck and Hoppe's method, [EH96] wrinkles arise on the spline surface due to a non-anisotropy adapted control mesh ; Right : using Alliez* et al.*'s method [ACSD+03], (a) cells of the control mesh extracted are quite irregularly-sized ; (b) natural cycles of curvature lines fail to be captured.*

such as disks [Flo97], spheres [GGS03] or torii [GGT06]. Constructing parameterizations for surfaces of arbitrary topology remains a challenging problem. One common solution is to introduce cuts into the initial surface mesh so as to convert it into one [GGH02] or several topological disks [LPRM02]. However, these cuts introduce discontinuities which may be visible as mapping artifacts during rendering, especially when mip-mapping is activated. These cuts also introduce artifacts for remeshing, such as artificial alignments of edges along the boundary, and unwanted variations of element sizes. Last but not least, finding appropriate cuts so as to minimize the artifacts listed above is notoriously difficult.

Globally smooth parameterization techniques aim at reducing such discontinuities [KLS03]. As reviewed in the next section, these techniques are limited in their ability to control the parametric distortion and the number and placement of singularities introduced into the parameterization. Many of these also require an a priori segmentation of the mesh into charts, which remains an open problem.

For many mesh processing applications it is advantageous to have a parameterization aligned with the principal curvature directions on the surface. For example, the extraction of quadrilateral control mesh, as in our case. In particular, as explained in [d'A00], for both surface fitting and remeshing, alignment with curvature improves convergence. To the best of our knowledge, *no existing parameterization method supports such alignment*. To achieve such alignment is one of the goal of our global parameterization. The rest of this section reviews the previous work and gives an overview of our approach.

F.1.2 Previous Work

Quad-Remeshing

– *Manual methods :*
The trivial solution to obtain a quad-remeshed version of the surface is to draw the boundary curves on the mesh manually as proposed in [KL96] and [MBVW95]. In [LLP05] (Chapter C) a method with combinatorial data structure is introduced, which facilitates this kind of curve drawing tasks on meshes (see Chapter C for detailed description). Some commercial softwares, for example, Rapidform [Rap] and Cyslice by Cyberware [Cyb] provide skill designers with tools to patch the objects manually. To provide even more efficient processing, some of them even provide templated-patching for objects with similar shapes and genii. However, manual patching still requires skilled 3D model designers to obtain a satisfactory effect.

– *Regular and semi-regular methods :*
By using a cut-graph that turns a surface into a topological disk, which is then parameterized into a square domain, one can obtain a fully regular control mesh by resampling the geometry image [GGH02]. To produce semi-regular quadrilateral control meshes, Eck and Hoppe [EH96] employed the technique of triangle merging (see Figure F.3-Left). Boier-Martin *et al.* [BMRJ04] proposed a method based on discrete Lloyd relaxation. Most recently, Dong *et al.* [DBG$^+$06] proposed an algorithm that is based on the fact that the Morse-Smale complex induced by any piecewise linear function quadrangulates the surface. In their examples, they used Laplacian eigenfunctions of the surface. Although automatic, the above methods are not anisotropy-adapted.

– *Quad-dominant methods :*
Some methods consider the anisotropy of the surface since it is optimum from a function approximation point of view [d'A00]. Alliez *et al.* [ACSD$^+$03] proposed quad-dominant remeshing method which adapts the anisotropy of the object by directly integrating curvature lines of the principal directions on the surface. Although anisotropy-adapted, the method has several drawbacks. First, globally, the cells of the control mesh extracted are not regularly-sized due to the greedy seeding algorithm of the placement of curvature lines. Second, since explicit numerical integration of curvature lines is used, some local feature, for instance, natural cycles (e.g. around a finger) often fail to be captured (see Figure F.3-Right). Later, we have improved the method by doing the integration directly on the surface without the need of the parameterization. (Marinov and Kobbelt proposed

a similar variant of Alliez *et al.*'s method in [MK04c].) Nevertheless, explicit integration of curvature lines is always plagued by the problems of the seeding and placement of the curvature lines. Attempting to solve these problems, Dong *et al.* [DKG05] used mixed implicit/explicit schemes with harmonic functions to obtain the control mesh. Recently, Marinov and Kobbelt [MK06] proposed a two-step method. First, they segment the surface into patches in a Variational Shape Approximation (VSA) fashion. Then, in each patch, a quad-mesh is generated by minimizing a bending energy of a network of curves in the parametric space. Using this method, sharp features are nicely preserved in the final mesh thanks to the VSA approach in the first step. However, the method is limited speed-wise for large patches due to the high complexity of the optimization in the second step.

Globally Smooth Parameterization

We focus our review on globally smooth parameterization methods, a review of the many other available mesh parameterization techniques being beyond the scope of this thesis. The reader is referred to [FH05] for a complete survey.

To construct a globally smooth parameterization, existing methods use two different strategies. One class of methods first partitions the object into a set of charts, parameterizes each chart independently, and applies a post-relaxation procedure to blur the discontinuities along chart boundaries. The other class of methods directly takes the topology of the surface into account and uses a global formulation to obtain the parameterization.

- *Inter-chart relaxation :*
  - In [KLS03], transition functions are introduced to define a relaxation procedure that optimizes inter-chart continuity. This relaxation is applied simultaneously to all charts.
  - A similar approach is used in *polycube maps* [THCM04]. First, a quadrilateral chart layout is manually constructed by the user. Then, the charts are parameterized using a globally smooth version of the MIPS method [HG00]. The parameterization constructed by our method shares some similarities with a polycube map, with the major difference that in our case, the quadrilateral chart layout is constructed *automatically*.
  - In [KS04, SPPH04] a parameterization between pairs of input models is computed. In both papers a triangular chart layout and a corresponding base-mesh are constructed automatically, and the parameterization is smoothed either onto the base-mesh [KS04] or onto the models themselves [SPPH04]. Note that a triangular chart layout is not suitable for quadrilateral remeshing and is far less suitable for spline fitting.

– *Global contouring using discrete one forms :*
  – By treating surfaces as complex manifolds, Gu and Yau [GY03] proposed to construct the so-called *conformal structure* of a surface. Namely, a basis of holomorphic one-forms, which is built from a basis of harmonic one-forms, which is itself obtained by diffusing a cohomology basis. The basic idea of the diffusion is that for each closed one-form ω in the cohomology basis, they find a 0-form f (on vertices) such that its exact one-form df (on edges) when added to ω gives a harmonic one-form. A set of mutually compatible local parameterizations is extracted from this structure. The continuity is achieved everywhere except at a number of *singular points*. Intuitively, and given a sphere with its common parameterization, those singular points correspond to its two poles. Jin *et al.* [JWGY04] found the unique locations of the singular points that satisfy conformality. They also solve for the optimal conformal transformation that minimizes global stretch. Since the number of singular points remains constant, the resulting parameterizations often still exhibit significant stretch.
  – Similarly Steiner and Fischer [SF05] demonstrated how to use a pair of discrete harmonic one-forms for the parameterization of the torus. Cyclic boundary (along the two homology generators) constraints are fixed. These constraints define the two discrete one-forms of a cohomology basis. These two one-forms are diffused into a harmonic one-form in a way similar to Gu and Yau's method except a slight different that the 0-form is discontinuous along boundary, i.e. two values are kept for each vertex along the boundary. Unlike in Gu and Yau's method, where the pair of one-forms must be holomorphic, in this method any pair of linearly independent harmonic one-forms is allowed. The method for the torus case is generalized to genus-g closed surfaces.
  – Along the same vein, Gortler *et al.* [GGT06], through a discrete one-form formalism, proved that any pair of two linearly independent harmonic one-forms generates a seamless bijective parameterization of the torus. From the properties of these one-forms, the authors develop a discrete counterpart of the Hopf-Poincaré index theorem. This theorem generalizes the notion of singular points mentioned above and allows their splitting and merging. It states that for a surface of genus g, the indices (multiplicities) of all singular points sum to $2-2g$.
  – Recently, Tong *et al.* proposed a method similar to Steiner and Fischer's. In this method, in addition to cyclic boundary constraints, two more types of boundary constraints are allowed. Namely, reverse and switch boundary constraints, which allow reversing of orientation of the 0-form and switching between the two 0-forms across boundaries

respectively. These new boundary constraints allow more complicated cut-graphs to be designed. (Not only restricted to cyclic boundary constricts along homology basis.) Hence, one have a better control of the singularities of the harmonic one-forms (but still respect Poincaré-Hopf theorem) as compared to Steiner and Fischer's method. Because of the reverse and switch boundary constraints, one can have singularities of indices $\pm 1/2$ and $\pm 1/4$.

Our method constructs a class of globally continuous overlapping local parameterizations along with continuity conditions. The main difference is that, motivated by geometry processing applications, we incorporate more geometric information into the problem setting. We optimize an energy functional that both minimizes the mapping distortion and optimizes the alignment of the iso-parametric curves with two orthogonal direction fields.

F.1.3 Algorithm Overview

The input to our algorithm is a triangle surface mesh, together with two orthogonal direction fields $\vec{K}$ and $\vec{K^{\perp}}$. The direction fields are typically the estimated principal directions of curvature, but any pair of user-defined direction fields are also valid as long as they match the orthogonality constraint. For instance, one can use direction fields designed using the method presented in Chapter E.

Our goal is to construct a globally smooth parameterization aligned with the guidance direction fields. More formally, the gradients (iso-parametric curves) of the parameterization optimized by our algorithm will be as tangential as possible to the guidance direction fields. If the direction fields are the directions of principal curvature, the iso-parametric curves will then be automatically aligned to the significant geometric features of the shape (fillets, axes of symmetry, etc.). The principal curvature directions can be estimated by a variety of techniques, such as [CSM03].

One of the main achievements of our algorithm is its ability to automatically extract a quadrilateral chart layout for the global parameterization. The size (side length) of the charts is determined by a user prescribed parameter ω. As demonstrated in Figure F.4, the extracted charts are mostly well shaped and have uniform sizing. They also exhibit a highly regular connectivity, with mostly valence four vertices.

Our method can construct a curvature-adapted globally smooth conformal parameterization or a quasi-isometric parameterization. In the latter case, the near zero distortion is obtained at the expense of introducing more singular points. This trade off is achieved by an additional processing step which controls the curl of the direction field. The algorithm for computing our

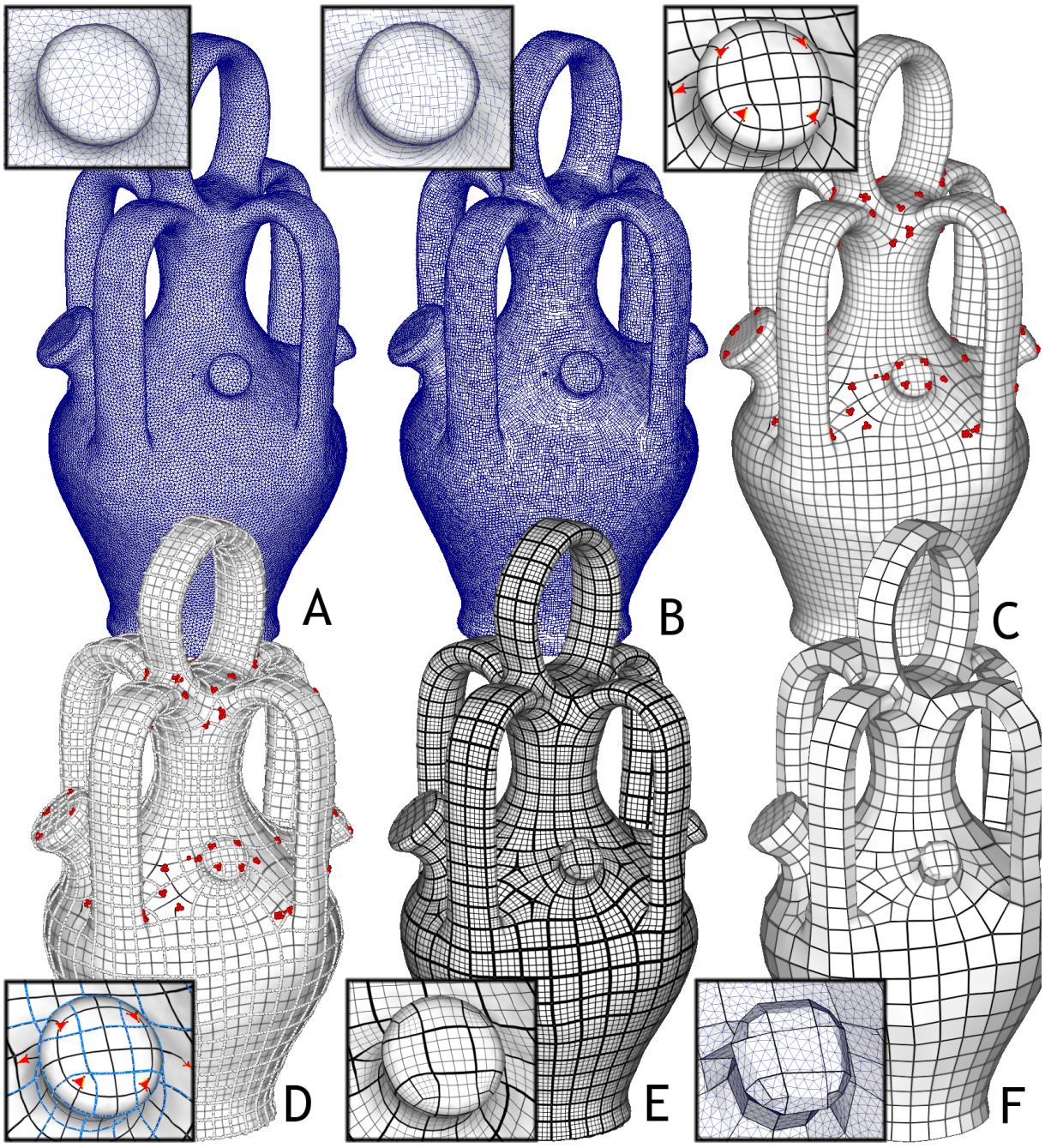


FIG. F.4: *Algorithm overview. A : input mesh model ; B : a pair of orthogonal guidance direction fields ; C : iso-kπ s and t curves. The singular vertices, edges and triangles are highlighted ; D : chart layout (extracted from the iso-2kπ curves) ; E : final result, obtained after fixing the charts with singularities ; F : the resulting base complex.*

global parameterizations consists of three stages :

- **Curl-correction** (optional) : A global isometric parameterization is usually not possible without a certain number of singular points. As explained in Section F.4, in our setting most of these points correspond to areas where the guidance direction fields are not curl-free. Hence, to reduce the number of singular points we introduce an optional procedure which rescales the direction fields so as to reduce their curl. The rescaled direction fields are used as input to the subsequent parameterization step. Notice that if curl-correction is applied, the resulting parameterizations will remain conformal and exhibit much fewer singular points, but will usually exhibit larger stretch (Figure F.11).
- **Parameterization using alternative variables** : This is the main step of our algorithm. To explicitly account for translational and rotational degrees of freedom in the parameterization formulation, we develop an energy functional using alternative variables which are trigonometric functions of the actual parameterization. The derivation of the functional is described in detail in Section F.2. The derived energy functional (Equation F.17) is minimized using the numerical procedure described in Section F.2.5.
- **Extraction of chart layout and chart parameterization** : The final stage of the algorithm computes the actual surface parameterization given the solution in terms of alternative variables (Section F.3) :
  - First, the algorithm extracts the parameterizations for each individual mesh triangle. Figure F.4-C depicts the iso-$k\pi$ curves of these parameterizations.
  - Second, the method detects the singularities present in the computed parameterizations. These are vertices, edges and triangles that do not satisfy the requirements of a valid 2D planar triangulation (highlighted in Figure F.4-C,D).
  - Third, the per-triangle parameterizations are used to define a global chart layout. N-sided chart are split into quadrilateral charts.
  - Finally the algorithm computes the per-chart parameterizations. If a chart does not contain any singularity, it reconstructs its parameterization by assembling the individual triangles in parameter space. Otherwise, the chart is re-parameterized. The final parameterization together with the corresponding chart layout are shown in Figure F.4-C and D respectively.

The result of the procedure is a global conformal parameterization, continuous almost everywhere. The parameterization is aligned with the guidance direction fields. If no curl-correction is performed, the parameterization is even shown to be quasi-isometric.

Our parameterization technique can be used for a variety of mesh processing applications. In

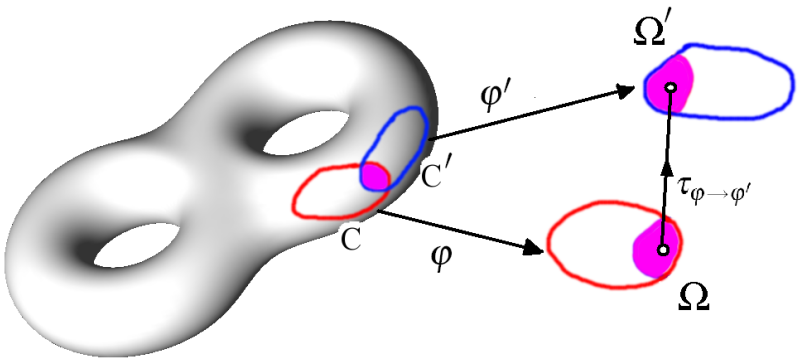


FIG. F.5: *A global parameterization (or manifold) is a set of overlapping parameterizations* $(\varphi, \varphi', \ldots)$ *connected by transition functions* $(\tau_{\varphi\to\varphi'} \ldots)$.

Section F.5, we demonstrate its use for curvature-aligned quad-dominant remeshing. Given our globally smooth curvature-aligned parameterization, the mesh generation procedure is elegant and straightforward. Other applications of our method are smooth surface reconstruction and texture mapping.

F.2 Periodic Global Parameterization

F.2.1 Definition

We first give the definition of a *manifold* (also called a global parameterization in our context). This notion allows us to define a globally smooth parameterization of a surface with arbitrary genus, by combining multiple parameterizations of charts extracted from the surface and linked by transition functions. To our knowledge, the notion of manifold was first used for geometric modeling by Grimm and Hugues [GH95]. More recently, a C^∞ class of surfaces based on manifolds was proposed in [YZ04].

Given a surface $\mathbf{S}$, we consider a set of (possibly overlapping) topological disks $\{\mathbf{C}\}$ called *charts*, and a set of functions $\{\varphi\}$ mapping each chart $\mathbf{C}$ to a 2D domain Ω (see Figure F.5). The coordinates in 2D space will be denoted by s,t in what follows. The set of functions $\{\varphi\}$ is called a global parameterization (or a manifold) if it satisfies the following condition :

Given two charts $\mathbf{C}$ *and* $\mathbf{C}'$*, if their intersection* $\mathbf{C}\cap\mathbf{C}'$ *is a topological disk, then the images of the intersection* $\mathbf{C}\cap\mathbf{C}'$ *in parameter space through* φ *and* φ' *are linked by a simple geometric transform* $\tau_{\varphi\to\varphi'}$ *:*

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

The $\tau_{\varphi \to \varphi'}$ functions are called *transition functions*. (see [KLS03]). Manifolds are called *affine* if all the transition functions are translations. *Complex* manifolds admit a more general class of holomorphic transition functions, including similarities (i.e. rotation + translation + scaling), see [Wei] for a definition of these classes of objects. Whereas previous work focus on affine manifolds [GY03, GGT06], our method constructs a sub-class of complex manifolds, allowing for both translational and rotational degrees of freedom in the transition functions. The extra degree of freedom allows for greater flexibility when aligning the parameterization with the guidance direction fields, as shown below.

Our goal is to construct a global parameterization such that the gradients ∇s, ∇t of the parameter-space coordinates s,t are aligned with two prescribed direction fields (for instance, the principal directions of curvature). We first start with the simplest possible charts, i.e. the triangles. In our initial setting, the global parameterization is defined by the coordinates s_i^T, t_i^T at the corners of the triangles, where the global index i denotes a vertex, and T denotes a triangle. Using the so-defined manifold structure, it is possible to derive a parameterization of more general charts, by assembling the triangles in parameter space, as explained later in Section F.3.3.

We first consider the case of an affine manifold (i.e. the transition functions $\tau_{\varphi \to \varphi'}$ are translations). We will then show how to introduce the rotational degree of freedom. Given two triangles $T = (i,j,k)$ and $T' = (k,j,l)$ sharing the edge (j,k), their parameter-space coordinates (s,t) define an affine manifold if :

$$\begin{pmatrix} s_j^T \\ t_j^T \end{pmatrix} - \begin{pmatrix} s_j^{T'} \\ t_j^{T'} \end{pmatrix} = \begin{pmatrix} s_k^T \\ t_k^T \end{pmatrix} - \begin{pmatrix} s_k^{T'} \\ t_k^{T'} \end{pmatrix} \tag{F.1}$$

We now need to derive an energy functional F, depending on all the (s_i^T, t_i^T) coordinates and characterizing the alignment of the gradients $(\nabla s, \nabla t)$ to the principal directions of curvatures. In our formulation of the energy functional F, instead of expressing Equation F.1 as a constraint, we replace the (s_i^T, t_i^T) variables with alternative variables, associated to the vertices (rather than to the corners of the triangles), and naturally satisfying the constraints. We will then show how to retrieve the (s_i^T, t_i^T)'s from those alternative variables.

We introduce an additional restriction on the transition functions $\tau_{\varphi \to \varphi'}$: the coordinates of the translation vectors connecting two charts should be integer multiples of 2π. With this additional constraint, we have :

$$\begin{pmatrix}\cos s_j^T\\ \sin s_j^T\end{pmatrix} = \begin{pmatrix}\cos(s_j^{T'}+2a\pi)\\ \sin(s_j^{T'}+2a\pi)\end{pmatrix} = \begin{pmatrix}\cos s_j^{T'}\\ \sin s_j^{T'}\end{pmatrix} \quad ; \quad \begin{pmatrix}\cos t_j^T\\ \sin t_j^T\end{pmatrix} = \begin{pmatrix}\cos t_j^{T'}\\ \sin t_j^{T'}\end{pmatrix} \tag{F.2}$$

(this condition is also satisfied at vertex k). As a consequence, and given a vertex i, for all the triangles T incident to i, the values of $\cos s_i^T$ and $\sin s_i^T$ (resp. t) coincide. We introduce the variables $U_i = (\cos s_i^T, \sin s_i^T)$ and $V_i = (\cos t_i^T, \sin t_i^T)$, which no-longer depend on T and are attached to the vertices instead.

We now consider the more general case of a sub-class of complex manifolds where transition functions $\tau_{\varphi\to\varphi'}$ can be combinations of translation *and* rotation. The coordinates of the translations are constrained to be multiples of 2π as before, and the rotation angles are constrained to be multiples of $\pi/2$. We will refer to this configuration as a periodic global parameterization. In this setting, the compatibility condition connecting two triangles (Equation F.2) is replaced with :

$$\begin{array}{l}\exists r \in \{0,1,2,3\}\\ \begin{pmatrix}\cos s_j^T\\ \sin s_j^T\end{pmatrix} = \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}^r \begin{pmatrix}\cos s_j^{T'}\\ \sin s_j^{T'}\end{pmatrix} \quad ; \quad \begin{pmatrix}\cos t_j^T\\ \sin t_j^T\end{pmatrix} = \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}^r \begin{pmatrix}\cos t_j^{T'}\\ \sin t_j^{T'}\end{pmatrix}\end{array} \tag{F.3}$$

As a consequence, given a vertex i, for all the triangles T incident to i, the values of $\cos s_i^T$ and $\sin s_i^T$ (resp. t) coincide up to a change of sign and a swapping of the sine and cosine.

We now show how to express the alignment with the guidance direction fields in terms of the variables (U_i, V_i) (Sections F.2.2,F.2.3,F.2.4) and explain how to retrieve the parameter-space coordinates (s_i^T, t_i^T) from these variables (Section F.3.1). We will then proceed to extract the chart layout (Section F.3.3), and show how to retrieve a parameterization of the charts from the per-triangle coordinates (s_i^T, t_i^T) (Section F.3.4).

F.2.2 Parameterization Alignment

As described in Section F.1.3 the input to our algorithm consists of two orthogonal *control* direction fields $\vec{K}$ and $\vec{K}^\perp$ defined on a surface S, and a chart size parameter ω. The guidance direction fields are defined at the vertices of the surface mesh and are linear across the triangles. The meaning of this parameter ω and the way to choose it are explained below. Our method aims at constructing a complex manifold $\{\varphi^T\} = \{(s^T, t^T)\}$ such that each function φ^T associated to a triangle T satisfies :

$$\nabla s^T = \omega\vec{K} \quad ; \quad \nabla t^T = \omega\vec{K}^\perp \tag{F.4}$$

In addition, the complex manifold should be a periodic global parameterization, i.e. the transition functions $\tau_{T\rightarrow T'}$ should be solely composed of translations multiples of 2π and rotations multiples of $\pi/2$.

When our goal is to construct a parameterization *as isometric as possible*, the magnitude of the guidance direction fields $\|\vec{K}\| = \|\vec{K}^{\perp}\| = 1$. If we want to reduce the curl of the direction fields and hence minimize the number of singularities in the parameterization, the direction fields are scaled as described in Section F.4. This leads to a parameterization which is no longer isometric, but which remains conformal.

Due to this normalization, the parameter ω controls the period of the s and t functions. As described in Section F.3.3, we will use the 2π periods of the parameterization to define the chart layout. Hence, ω will determine the size of the charts. In all our examples, we set ω to ten times the average edge length in the input mesh. Note that if ω is too large, charts that are not homeomorphic to disks may be generated. This can be easily detected by computing the Euler-Poincaré characteristic of the charts, and ω can be automatically decreased if such a configuration is detected.

Since $curl(\nabla\rho) = 0$ for any scalar field ρ, a solution to Equation F.4 exists only if $curl(\vec{K}) = curl(\vec{K}^{\perp}) = 0$ [Nee94]. In general, the guidance direction fields might have non-zero curl. Hence, we have to restate our goal in weaker terms by minimizing the following energy functional :

$$F = \int_S \left(\|\nabla s^T - \omega\vec{K}\|^2 + \|\nabla t^T - \omega\vec{K}^{\perp}\|^2 \right) dS \tag{F.5}$$

Given this problem setting, the main difficulty is to express the alignment of the parameterization gradients with the control direction fields independently from the translational and rotational degrees of freedom. We first introduce translation-invariance into the formulation in Section F.2.3, and then refine the formulation to introduce the rotational degree of freedom (Section F.2.4).

F.2.3 Translation-invariant Energy Functional

The main challenge in the formulation given by Equation F.5 is to find a way to solve for a *periodic* function. As explained in Section F.2.1, to support translational invariance in parameter space, we propose to use the 2π periodicity of the sine and cosine functions. As shown below, it is possible to restate the alignment with the guidance direction fields in terms of the sines and cosines $U = (\cos s, \sin s)$ and $V = (\cos t, \sin t)$ of the parameters s and t. The U and V's will be

the unknowns of our problem. Thus, we will obtain a periodic definition of the minimizer of the energy functional F.

Instead of minimizing F, we minimize the following simpler function F^* :

$$F^* = \sum_T \int_T \left(\|\nabla s^T - \omega \vec{K}_T\|^2 + \|\nabla t^T - \omega \vec{K}_T^{\perp}\|^2 \right) dS \tag{F.6}$$

where $\vec{K}_T$ and $\vec{K}_T^{\perp}$ denote the average value of $\vec{K}$ (resp. $\vec{K}^{\perp}$) across the triangle.

We show that F and F^* have the same minimizer as follows : given a piecewise linear vector field $\vec{K}$ and its average value $\vec{K}_T$ on a triangle T, we consider the two energy functionals $F_T = \int_T (\nabla s - \omega \vec{K})^2 dS$ and $F'_T = \int_T (\nabla s - \omega \vec{K}_T)^2 dS$ have the same minimizer :

$$\begin{aligned} F_T &= \int_T (\nabla s - \omega \vec{K})^2 dS \\ &= \int_T \left((\nabla s - \omega \vec{K}_T) + (\omega \vec{K}_T - \omega \vec{K}) \right)^2 dS \\ &= \int_T (\nabla s - \omega \vec{K}_T)^2 dS + 2 \int_T (\omega \vec{K}_T - \omega \vec{K})^t (\nabla s - \omega \vec{K}_T) dS + \int_T (\omega \vec{K}_T - \omega \vec{K})^2 dS \\ &= \int_T (\nabla s - \omega \vec{K}_T)^2 dS + 2 (\nabla s - \omega \vec{K}_T)^t \int_T (\omega \vec{K}_T - \omega \vec{K}) dS + \int_T (\omega \vec{K}_T - \omega \vec{K})^2 dS \end{aligned} \tag{F.7}$$

The first term of this expression is F'_T, the second term vanishes (by definition of the average value $\vec{K}_T$) and the third term does not depend on s. As a consequence, we have $F_T = F'_T +$ constant. Therefore, F_T and F'_T have the same minimizer.

Since ∇s^T and ∇t^T are constant across each triangle, we have

$$F^* = \sum_T \left(\|\nabla s^T - \omega \vec{K}_T\|^2 + \|\nabla t^T - \omega \vec{K}_T^{\perp}\|^2 \right) A_T \tag{F.8}$$

where A_T is the area of triangle T. We now consider a single entry in this sum :

$$F_T = \left(\|\nabla s^T - \omega \vec{K}_T\|^2 + \|\nabla t^T - \omega \vec{K}_T^{\perp}\|^2 \right) A_T \tag{F.9}$$

Since it is difficult to introduce translational invariance directly into F_T, we will first study $F^s_{T,i}$, the energy along the edges $\vec{e}_i$ of T (Figure F.6) with respect to s. The energy $F^t_{T,i}$ with respect to t is derived in a similar manner. We will then express F_T as a linear combination of the $F^s_{T,i}$'s and $F^t_{T,i}$'s.

Intuitively, when considering the difference along the edge $\vec{e}_i$, we need to consider the difference between the projection of $\vec{K}$ to the edge and the gradient ∇s along the edge. The projection

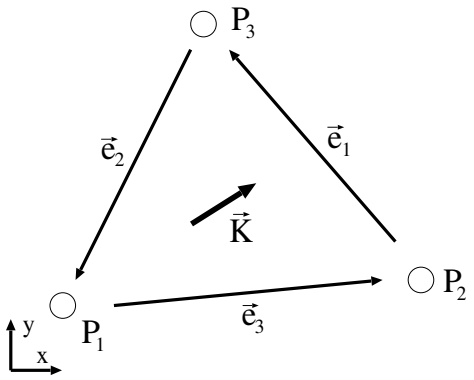


FIG. F.6: *Triangle notations.*

is $\vec{K}\cdot\vec{e}_i/\|\vec{e}_i\|$, and the gradient along the edge is $(s_{i\oplus 2}-s_{i\oplus 1})/\|\vec{e}_i\|$ where $i\in\{1,2,3\}$ denotes a local index in T (see Figure F.6) and $\oplus$ denotes addition modulo 3.

We define the energy along the edge as

$$F^s_{e_i} \quad = \quad \int_{\vec{e}_i}\left(s_{i\oplus 2}-s_{i\oplus 1}-\vec{K}\cdot\vec{e}_i\right)^2/\|\vec{e}_i\|^2 dS \tag{F.10}$$

With a derivation similar to the one given in F.7, since $\vec{K}$ and $\vec{K^{\perp}}$ are linear along the edges, we can replace $\vec{K}$ with $\vec{K}_i=(\vec{K}_{i\oplus 2}+\vec{K}_{i\oplus 1})/2$ and scale the minimized function by $\|\vec{e}_i\|$ without changing the minimizer. The new energy functional that we will minimize per edge is

$$F^s_{T,i} \quad = \quad \left((s_{i\oplus 2}-s_{i\oplus 1})-\omega\vec{K}_i\cdot\vec{e}_i\right)^2 \tag{F.11}$$

Using this energy formulation, it is now easy to introduce the translational invariance, replacing the difference by a difference modulo translation by 2π :

$$F^s_{T,i} \quad = \quad \min_a\left\{((2a\pi+s_{i\oplus 2}-s_{i\oplus 1})-\omega\vec{K}_i\cdot\vec{e}_i)^2\right\} \tag{F.12}$$

By approximating this difference by the norm of the difference of the sine and cosine vectors, corresponding to order 1 Taylor expansion, we obtain :

$$F^s_{T,i} \quad \simeq \quad \left\| U_{i\oplus 2}-\begin{pmatrix}\cos(\omega\vec{K}_i\cdot\vec{e}_i) & -\sin(\omega\vec{K}_i\cdot\vec{e}_i)\\ \sin(\omega\vec{K}_i\cdot\vec{e}_i) & \cos(\omega\vec{K}_i\cdot\vec{e}_i)\end{pmatrix}U_{i\oplus 1}\right\|^2 \tag{F.13}$$

where :

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

Note that using this formulation, we no longer depend on the translational coefficient a (Equation F.12).

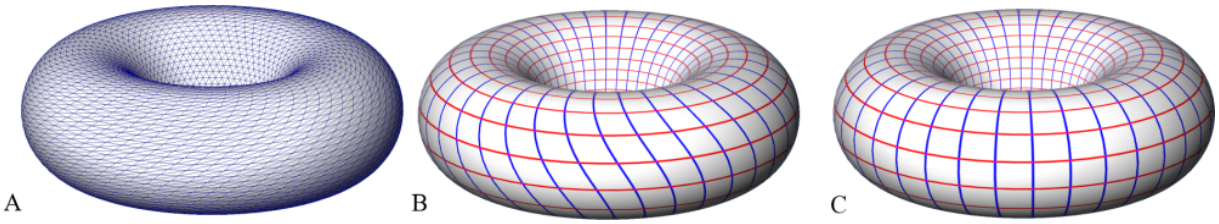


FIG. F.7: *A : a meshed torus with a strong mesh anisotropy ; B : result of the edge-based PGP : the parameterization is influenced by the mesh anisotropy ; C : result of the triangle-based PGP : the parameterization solely depends on the geometry.*

Note that theoretically at this point, we can simply minimize $\sum_{T,i\in 1,2,3}(F^s_{T,i}+F^t_{T,i})$, which corresponds to a discrete, edge-based version of the energy. The resulting method works well for regularly sampled surfaces but is sensitive to anisotropic samplings. For this reason, we minimize the energy F_T integrated over the triangle T (Equation F.9). In the following, we show that F_T can be expressed as a linear combination of the $F^s_{T,i}$ and $F^t_{T,i}$ edge energies :

$$F_T = (\|\nabla s - \omega\vec{K}_T\|^2 + \|\nabla t - \omega\vec{K}^{\perp}_T\|^2)A_T = \sum_{i=1}^{3}\lambda_i^T(F^s_{T,i}+F^t_{T,i})$$

where $(\lambda_1^T,\lambda_2^T,\lambda_3^T)$ are the solutions of :

$$\begin{pmatrix} (e_{1,x})^2 & (e_{2,x})^2 & (e_{3,x})^2 \\ (e_{1,y})^2 & (e_{2,y})^2 & (e_{3,y})^2 \\ 2e_{1,x}e_{1,y} & 2e_{2,x}e_{2,y} & 2e_{3,x}e_{3,y} \end{pmatrix} \begin{pmatrix} \lambda_1^T \\ \lambda_2^T \\ \lambda_3^T \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \tag{F.14}$$

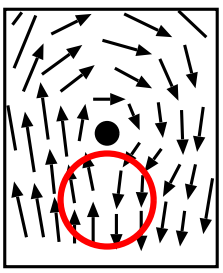
The linear system is obtained by expanding and equating both terms of Equation F.14.

In our experiments this greatly improved the results without adding too much computation overheads. Figure F.7 compares the results obtained with the edge-based and the triangle-based energy on a mesh with a strong anisotropy.

Our current formulation for F^* supports translational invariance. We now proceed to introducing rotational invariance into the formulation.

F.2.4 Rotation-invariant Energy Functional

In general, it is not possible to *globally* orient a direction field in a consistent way (see the circled region). To alleviate this issue, we modify the formulation of the triangle energy (Equation F.9) by *locally* reorienting the direction field in the new formulation (Figure F.8). The orienta-



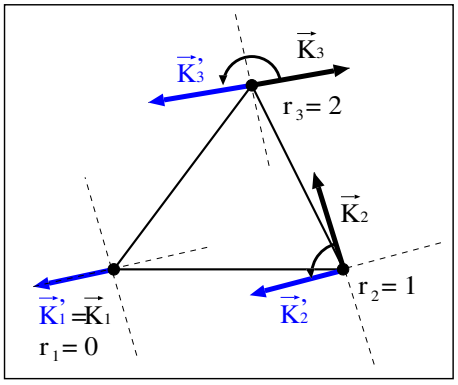


FIG. F.8: *Locally re-orienting the guidance direction field.*

tions of the vectors $\vec{K}_1$, $\vec{K}_2$ and $\vec{K}_3$ at the respective vertices of the triangle (Figure F.8) are now allowed to vary by multiples of $\pi/2$. Thus, $\vec{K}_2$ (resp. $\vec{K}_3$) is aligned with $\vec{K}_1$ by applying r_2 rotations of $\pi/2$ (resp. r_3).

The rotation is applied simultaneously to the guidance direction fields $(\vec{K}_i, \vec{K}_i^{\perp})$ and to the unknowns $(s_i \;\; t_i)$. Note that an odd difference of r_i along an edge means swapping the unknowns (i.e., connecting s's with t's).

To define the objective function F_T on the triangles, we use the same approach as in previous Section. We first express the deviation $F_{T,i}$ along an edge, then express F_T as a linear combination of the $F_{T,i}$'s as defined by Equation F.14. Since the s's and the t's may be coupled, we can no longer separate them.

Adding rotational invariance, Equation F.12 becomes

$$F_{T,i} = \min_{a,b} \left\| \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_{i\oplus 2}} \begin{pmatrix} s_{i\oplus 2} \\ t_{i\oplus 2} \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_{i\oplus 1}} \begin{pmatrix} s_{i\oplus 1} + 2a\pi \\ t_{i\oplus 1} + 2b\pi \end{pmatrix} - \begin{pmatrix} \delta_i \\ \delta_i^{\perp} \end{pmatrix} \right\|^2$$

where :

$$r_i = \underset{r \in \{0,1,2,3\}}{\text{argmax}} \left(\vec{K}_1 . \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r} \vec{K}_i \right) \; ; \; \vec{K}_i{}' = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_i} \vec{K}_i \tag{F.15}$$

$$\delta_i = \omega/2 \left(\vec{K}'_{i\oplus 1} + \vec{K}'_{i\oplus 2} \right) \cdot \vec{e}_i \quad ; \quad \delta_i^{\perp} = \omega/2 \left(\vec{K}'^{\perp}_{i\oplus 1} + \vec{K}'^{\perp}_{i\oplus 2} \right) \cdot \vec{e}_i$$

As in previous Section, to take the periodicity of the (s,t) parameters into account, we solve for the sines and the cosines of these parameters. $F_{T,i}$ as a function of the sines and cosines (using the same order 1 approximation as in Equation F.13) is then given by :

$$F_{T,i} \simeq \left\| M^{r_{i\oplus 2}} X_{i\oplus 2} - \begin{pmatrix} \cos\delta_i & -\sin\delta_i & 0 & 0 \\ \sin\delta_i & \cos\delta_i & 0 & 0 \\ 0 & 0 & \cos\delta_i^{\perp} & -\sin\delta_i^{\perp} \\ 0 & 0 & \sin\delta_i^{\perp} & \cos\delta_i^{\perp} \end{pmatrix} M^{r_{i\oplus 1}} X_{i\oplus 1} \right\|^2 \tag{F.16}$$

$$\text{where :} \quad M = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad ; \quad X_i = \begin{pmatrix} U_i \\ \hline V_i \end{pmatrix} = \begin{pmatrix} \cos s_i \\ \sin s_i \\ \cos t_i \\ \sin t_i \end{pmatrix}$$

As in the previous section, we can simply minimize $F^* = \sum_{T,i\in 1,2,3}(F^s_{T,i} + F^t_{T,i})$, which corresponds to a discrete, edge-based version of the energy, or we can plug this expression into the triangle energy formulation (see Equation F.14). We now have

$$F^* = \sum_T F_T = \sum_T \sum_{i=1}^{3} \lambda_i^T (F^s_{T,i} + F^t_{T,i}) \tag{F.17}$$

where λ_i^T are given by Equation F.14 and $F^s_{T,i}, F^t_{T,i}$ are given by Equation F.16. The minimizer of F^* is computed as described next.

F.2.5 Numerical Solution Mechanism

To obtain the minimizer of F^*, we lock one of the vertices $U_1 = (1,0), V_1 = (1,0)$ and minimize F^* with respect to all the other variables. Since F^* is a quadratic form, this means solving a sparse symmetric system. We use the conjugate gradient algorithm with Jacobi's preconditioner. For models with more than 50K vertices, the norms of the U_i, V_i's quickly decrease when we move far away from the locked vertex, resulting in both weighting biases and numerical instabilities. To stabilize the system, we add a (non-linear) penalty term, preventing the norms of the U, V's from decreasing :

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

This augmented energy functional is minimized using Newton's algorithm. In our tests, $\varepsilon = 10^{-3}$ gives good results. Convergence, to $\nabla F^{**} < 10^{-6}$, is reached after no more than 5 outer-loop iterations for all the models shown in this paper.

The penalty term has an interesting property. The singular points correspond to vectors with zero norm. Since two singular points located in the same region drastically increase the penalty

function in that region, the augmented energy functional attempts to avoid those configurations and evenly distributes the singular points over the surface.

F.3 Parameterization Extraction

The output of the solution mechanism described in previous Section is a set of U_i, V_i variables. These variables correspond to the sines and cosines of the unknown s_i, t_i coordinates that define the global parameterization. To construct a global parameterization from those U_i, V_i variables, we proceed as follows :

1. reconstruct a (s,t) parameterization over the simplest possible charts, i.e. within each individual triangle (Section F.3.1),
2. detect the singular vertices, edges and triangles (Section F.3.2),
3. define the chart layout based on the 2π periods of the per-triangle parameterizations and split N-sided charts into N quadrilaterals (Section F.3.3),
4. compute the per-chart parameterization (Section F.3.4).

F.3.1 Per-triangle Parameterization

Given the U,V variables at the vertices of a triangle $T = (i,j,k)$, finding the s_i, t_i (resp. j,k) coordinates means determining the integer translational (a_i, b_i) and rotational r_i degrees of freedom. We explicitly determine the values of r,a,b that minimize the edge-energy term (Equation F.15) within each triangle as follows.

To define the global position and orientation of the triangle in parameter space, we set the degrees of freedom r_i, a_i, b_i of the first vertex i to $(0,0,0)$. Thus, the s_i^T, t_i^T coordinates at vertex i are given by $s_i^T = \text{angle}(U_i)$ and $t_i^T = \text{angle}(V_i)$ where $\text{angle}(U_i) = \text{sign}(U_{i,y})\arccos(U_{i,x}/\|U_i\|)$.

We now assign the coordinates at the two other vertices j and k by determining the differences a_e^T, b_e^T, r_e^T of the translational and rotational degrees of freedom along the edge $e = (i,j)$ (resp. $(i,k),(j,k)$), given by $a_e^T = (a_j^T - a_i^T), b_e^T = (b_j^T - b_i^T)$ and $r_e^T = (r_j^T - r_i^T)$. We first consider the edge (i,j). Given the coordinates (s_i^T, t_i^T) at vertex i and the guidance direction field values $(K_i, K_i^\perp)$ and $(K_j, K_j^\perp)$ at the vertices i,j, Algorithm 4 explicitly computes the differences a_e^T, b_e^T, r_e^T, then the values of s_j^T and t_j^T. The computation for the two other edges is obtained by a circular permutation of indices (i,j,k).

The algorithm first determines whether or not the guidance direction fields undergo a rotation along the edge. In this case, we sometimes change the correspondence between s,t and

Algorithm 4 reconstruction of s,t along an edge

propagate from i **to** j **along** $e=(i,j)$:

// determine and apply rotation r_e

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

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

$$\begin{matrix} s_j^T \leftarrow \text{angle}(U_j^T) \\ t_j^T \leftarrow \text{angle}(V_j^T) \end{matrix} \quad ; \quad \left(\begin{matrix} s_j^T \\ \hline t_j^T \end{matrix}\right) \leftarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{r_e^T} \left(\begin{matrix} s_j^T \\ \hline t_j^T \end{matrix}\right)$$

// determine and apply translations s,t

$\vec{n} \leftarrow \vec{e}/\|\vec{e}\|$

$$s_e^T \leftarrow \operatorname{argmin}_a \left| s_i^T - (\pi/\omega)\vec{n}\cdot(\vec{K}_i+\vec{K}_j) - s_j^T + 2a\pi \right|$$

$$t_e^T \leftarrow \operatorname{argmin}_b \left| t_i^T - (\pi/\omega)\vec{n}\cdot(\vec{K}_i^{\perp}+\vec{K}_j^{\perp}) - t_j^T + 2b\pi \right|$$

$$s_j^T \leftarrow s_j^T + 2a_e^T\pi \quad ; \quad t_j^T \leftarrow t_j^T + 2b_e^T\pi$$

U,V. For instance a rotation of $\pi/2$ corresponds to switching U and V. In this case, s becomes a function of V and t a function of $-U$. The algorithm then determines the difference a_e^T, b_e^T of the translational degree of freedom by explicitly minimizing the edge energy.

F.3.2 Characterization of Singular Vertices, Edges and Triangles

Once we have reconstructed the parameterization separately in each triangle, we need to check if these triangles can be assembled in parameter-space in such a way that they form a valid planar triangulation. We already know that the solution of the continuous version of the equation presents singularities where the derivatives of the solution vanish. In our discrete setting, these singularities appear as vertices, edges and triangles that violate the conditions of

a valid planar triangulation. These singular vertices, edges and triangles can be characterized as follows (see [SdS01]) :

- **Singular vertices :** a vertex v is singular if the angles at the corners of the triangles incident to v do not sum to 2π. In practice, a singular vertex v can also be characterized by the fact that applying Algorithm 4 to the one-ring neighborhood of v results in an open path.
- **Singular edges :** an edge $e = (i, j)$ is singular if its length in parameter-space mismatches with the one of $e' = (j, i)$.
- **Singular triangles :** a triangle T is singular if applying Algorithm 4 to its three edges results in an open path or if T has a negative area.

We explicitly test for those conditions. Example of all three types of singularities can be seen in Figures F.4 and F.12.

F.3.3 Extracting Chart Layout

Once we have computed the local parameterization in each triangle T, we construct the chart layout. In our setting, the chart boundaries are defined to be the iso-$2k\pi$ lines of s and t. This defines a set of segments in each triangle. We show below that the set of all the iso-$2k\pi$ lines of s and t is invariant under our transition functions. As a consequence, the end-points of these independent segments match along the non-singular edges of the triangulation, and the segments form continuous polygonal lines.

- **invariance of the set of iso-lines under valid translations :** if a triangle T is traversed by an iso-$2k\pi$ line of s (resp. t), this triangle translated by $2a\pi$ will be traversed at the same location by the iso-$2(k+a)\pi$ line of s (resp. t).
- **invariance of the set of iso-lines under valid rotations :** if a triangle T is traversed by an iso-$2k\pi$ line of s (resp. t), this triangle rotated by $\pi/2$ will be traversed at the same location by the iso-$2(-k)\pi$ line of t (resp. iso-$2k\pi$ line of s). The same argument applies to rotation by any multiple of $\pi/2$.

Note that given two adjacent mesh triangles T_1 and T_2 sharing a non-singular edge $e = (i, j)$ the transition function $\tau_{T_1 \to T_2}$ between the per-triangle parameterizations can be computed as

Algorithm 5 chart boundaries construction

compute chart boundaries :
for each triangle T
  if $T$is non-singular
    for $k \in \mathbb{N}$ such that $2k\pi \in [\min_T(s), \mathrm{Max}_T(s)]$
      Line $l \leftarrow$ line of equation$(s = 2k\pi)$
      Segment $S \leftarrow l \cap T$ //*in parameter space*
      store S in T
      store the end-points of S in the corresponding edges of T
    end//*for*
    repeat the above procedure for t
  end//*if*
end//*for*
for each edge e
  merge the segment end-points stored in e that have the same geometric location in 3D
end//*for*
for each triangle T
  compute the intersections between the edges stored in T
end//*for*
recursively remove all dangling segments

follows :

$$\begin{aligned} \tau_a &= a_i^{T_1} - a_i^{T_2} \\ \tau_b &= b_i^{T_1} - b_i^{T_2} \\ \tau_r &= r_e^{T_1} - r_e^{T_2} \\ \tau(p) &= R^{\tau_r} p + (\tau_a, \tau_b) \end{aligned} \tag{F.18}$$

where R is rotation by $\pi/2$. Hence the transition functions satisfy the invariance criteria above and therefore the end points of the iso-lines match.

Algorithm 5 computes the chart boundaries. Each triangle stores a list of segments, and each edge stores a list of segment end-points. The algorithm computes the individual segments defined by the intersections of the triangles with the $(s = 2k\pi)$ and $(t = 2k\pi)$ lines. Both 2D and 3D locations at the end-points of the segments are computed. The algorithm merges the

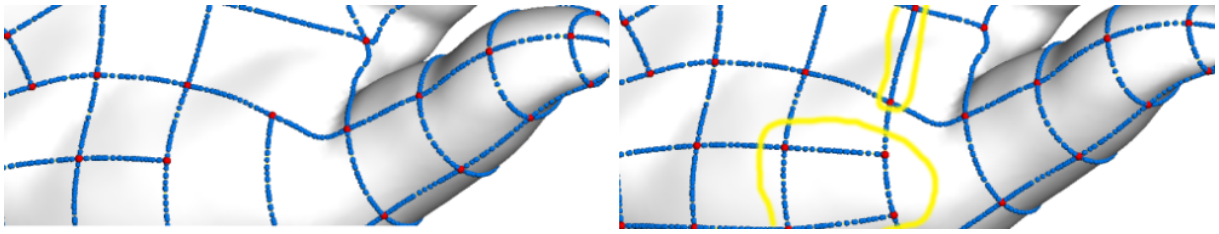


FIG. F.9: *Manually editing the initial control mesh by adding geodesics (circled). Note that all the intersections are kept up-to-date.*

segment end-points along the edges and intersects the segments inside the triangles adding the intersections as new end-points. All the dangling segments are removed (each segment end-point of valence 1 is "nibbled" until an end-point of valence higher than 2 is encountered). All the above operations to extract the chart layout are implemented using our embedded cellular complex data structure introduced in Chapter C.

Manual Editing and Geodesic Design

Since the automatically extracted chart layout is stored in our embedded cellular complex data structure, *optionally*, one can easily and efficiently improve over the chart layout manually through the following three operations (Figure F.9) in the spirit of "Map-Sketching" (see Chapter C.5) :

- edge straightening : an edge of the control mesh is replaced with a geodesic ;
- edge insertion : a new edge (a geodesic) is created between two points picked on the surface. All the intersections are kept up-to-date ;
- edge deletion.

Note that all the charts with edges modified by the user are marked as singular.

N-sided charts splitting

After the automatic extraction and optional manual editing, the chart layout now consists of quadrilaterals and a small number of N-gons. Since our target representation is T-spline, in order to obtain a chart layout that corresponds to a valid T-Mesh, we split each N-sided charts as explained Section B.4.5 into quadrilateral charts, which are marked as singular. At this point, the chart layout defines a control mesh conform with the T-Mesh validity requirements.

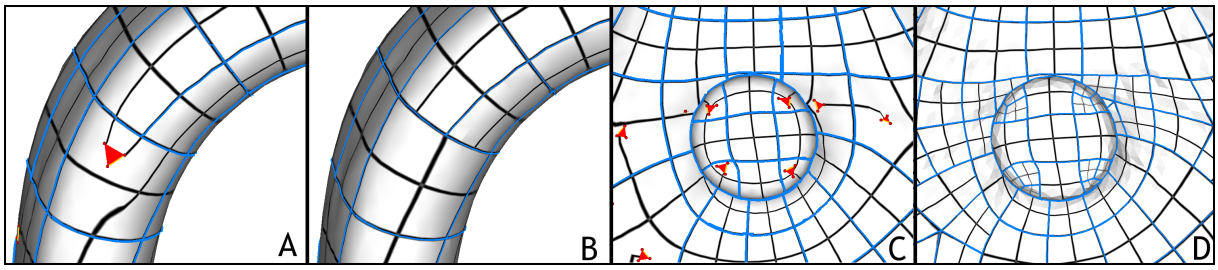


FIG. F.10: *Re-parameterizing charts with singularities. A,B : charts with four corners are re-parameterized using mean-value coordinates. C,D : N-sided charts are split into quadrilateral charts and those are re-parameterized.*

F.3.4 Per-Chart Parameterization

Charts without singularities

The parameterization of the charts that do not contain any singularity can be retrieved by assembling the triangles in 2D space by a classic greedy algorithm (see e.g. [SdS01]).

Re-parameterization of charts with singularities

Charts that contain singularities, are re-parameterized, using the mean value coordinates method [Flo03] (Figure F.10 A & B). The parameterization of the boundary vertices is adjusted to preserve C^0 continuity along the chart boundaries. If desired, the cross-boundary continuity can be improved by applying local relaxation as described in [KLS03, SPPH04].

In our experiments, only a small fraction (between 2 and 5 %) of the charts contain singularities and require this additional processing.

By combining the Periodic Global Parameterization (Section F.2) with the reconstruction algorithm presented in this Section, now we have a global parameterization of a surface. The global chart layout defines a valid control mesh for the T-splines. The next section presents one of the two optional pre-processing stage, curl correction, applicable to the guidance direction fields. For the other optional pre-processing stage, please see Section D.4.

F.4 Curl Correction

In the formulation presented in Section F.2, the emphasis was on constructing a parameterization as isometric as possible. This formulation is suitable for many applications, such as 3D paint systems, where parametric distortion is the dominant consideration.

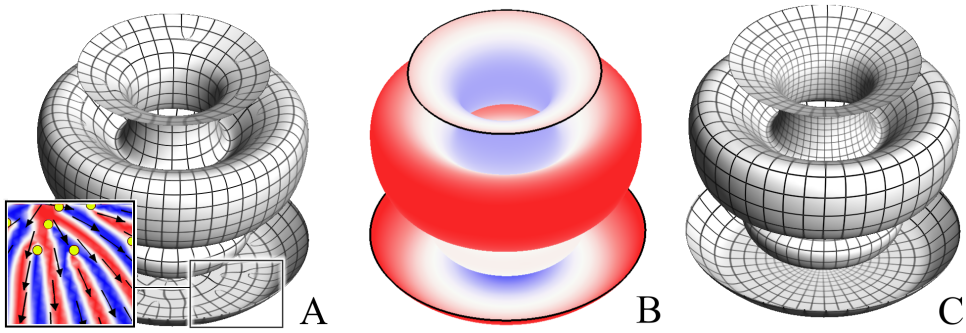


FIG. F.11: *Curl-correction applied to an object of revolution. A : the quasi-isometric parameterization contains singularities (shown as dots) which correspond to sources of the diverging direction field ; B : scaling factor - the magnitude of the curl-corrected direction fields ; C : solution obtained with the curl-corrected direction fields.*

For other applications isometry is less important, while the number of singularities in the parameterization is a serious concern. For instance, in quad-dominant remeshing, each branching point results in an undesirable N-gon in the mesh.

We therefore introduce an optional pre-processing technique, that scales the direction fields prior to parameterization in order to minimize the number of N-gons. As a result of the scaling the resulting parameterizations remain conformal (since the fields remain orthogonal and have the same norm) but are no longer isometric.

Since non-zero curl in the direction field leads to singularities in the parameterization, the goal of the proposed rescaling is to minimize the curl of the control vector fields $\vec{K}$ and $\vec{K}^{\perp}$. Since in our initial setting (Section F.2.2) the guidance direction fields are of unity norm, curl can arise only from non-parallel vectors (directional curl). As a consequence, eliminating the curl means rescaling the direction fields in such a way that the modular curl, arising from variations of the norm $\|\vec{K}\|$, cancels the directional curl. Note that our problem is different from computing a Hodge decomposition (see e.g., [TLHD03]), since we want to preserve the directions of the guidance fields.

Given a unit vector field $\vec{K}$ defined over a surface S, we want to find a scalar field v such that $curl(v\vec{K}) = \vec{0}$. The vectors will become shorter in converging regions $(v < 1)$ and longer in diverging regions $(v > 1)$. Note that since $\vec{K}$ and $\vec{K}^{\perp}$ are coupled by relationship $curl(\vec{K}) = div(\vec{K}^{\perp})\vec{N}$ (where $\vec{N}$ denotes the normal to S), the same scaling v needs to be applied to both $\vec{K}$ and $\vec{K}^{\perp}$. In terms of complex analysis, this coupling can be explained also by the fact that the θ and ϕ functions determine a *complex potential* of $\vec{K}$, which is necessarily a *conformal* function (see [Nee94]).

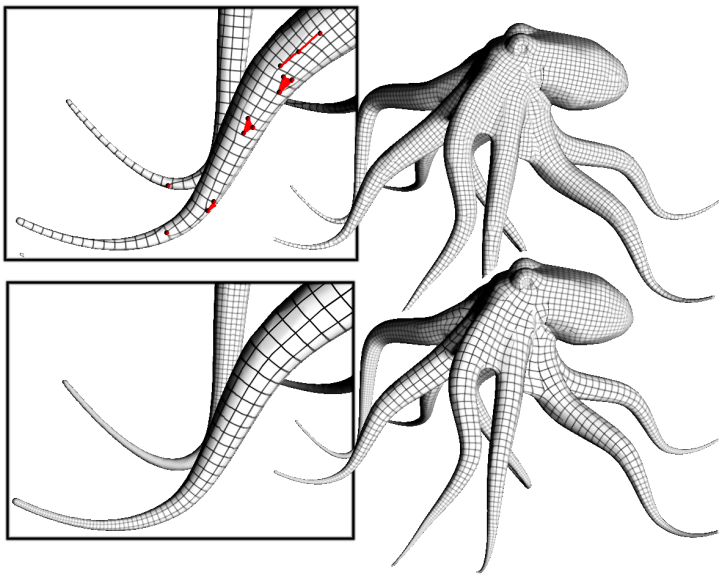


FIG. F.12: *Curl-correction applied to a model with sharp features. Note how all the singularities (triangles, edges, and vertices) on the tentacles (top) are removed by the curl correction (bottom).*

To develop a simple linear formulation for computing v we use a different setting for defining $\vec{K}$ (and $\vec{K}^{\perp}$) than the one used in the main parameterization procedure (Section F.2). Note that since the two procedures are stand-alone, this has no baring on the final result. For curl-correction purposes we assume that the *direction* of $\vec{K}$ varies linearly over T. In other words, given a local orthonormal frame (x,y) of T, we can parameterize the vector $\vec{K}$ by the angle γ between $\vec{K}$ and the x axis : $\vec{K} = (\cos(\gamma), \sin(\gamma))$, with $\gamma = ax + by + c$ (γ varies linearly over T). Replacing $\vec{K}$ with this expression we can express the zero-curl requirement per triangle as :

$$\begin{aligned} &curl(v\vec{K}) \cdot \vec{N} = \left(-\frac{\partial v}{\partial y} + va\right)\cos(\gamma) + \left(\frac{\partial v}{\partial x} + vb\right)\sin(\gamma) = 0 \\ &\text{where } \gamma = ax + by + c \end{aligned} \tag{F.19}$$

We search for solutions of Equation F.19 which are independent of rotations applied to the direction field $\vec{K}$, i.e., independent of the constant c. The solutions of the following system of PDEs meet this requirement :

$$\begin{cases} -\partial v/\partial y + va &= 0 \\ \partial v/\partial x + vb &= 0 \end{cases} \tag{F.20}$$

The solutions of Equation F.20 have the form $v = Ce^{ay-bx}$, where C denotes a constant. To

solve for the values v_i of v at the vertices *globally*, we express the condition satisfied by the variations of v :

$$\begin{aligned} \log(v) &= log(C) + ay - bx \\ \nabla \log(v) &= \nabla(ay - bx) = (-b \;\; a) \end{aligned} \tag{F.21}$$

Since, a solution with zero-curl over the entire surface might not exit, we solve for the $\tilde{v}_i = \log(v_i)$'s using a least squares formulation :

$$G(\tilde{v}) = \sum_T A_T \left\| J_T \begin{pmatrix} \tilde{v}_1 \\ \tilde{v}_2 \\ \tilde{v}_3 \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} J_T \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} \right\|^2 \tag{F.22}$$

where :

$$J_T = 1/2A_T \begin{pmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix}$$

where (x_i, y_i) are the coordinates of the vertices of T in the local frame.

Since the solution is independent of a global scaling applied to all the v_i's, we set $\tilde{v}_1 = 0$ and solve for all the other $\tilde{v}_i$'s. Then, we compute the scaling coefficients $v_i = \exp(\tilde{v}_i)$, and normalize them by dividing them by $\max(v_i)$.

The computed scaling coefficients v_i are introduced into Equation F.5

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

and the parameterization algorithm proceeds as described in Section F.2.

F.5 Results and Discussions

The Periodic Global Parameterization method (PGP) was implemented as part of the Graphite [Gra] mesh processing package. Throughout the paper we demonstrate the parameterizations computed with PGP on different complex models.

Figures F.4, F.14 show the final parameterization and the extracted chart structure. Figure F.13 shows the iso $k\pi$ lines obtained with the "David" dataset. Those Figures demonstrate that even on very complex models the number of singularities generated by our method remains very small ($2-3\%$ of the triangles). The number is particularly low when curl-correction is applied (zero in the example in Figure F.11 and 27 for the octopus in Figure F.12). When no

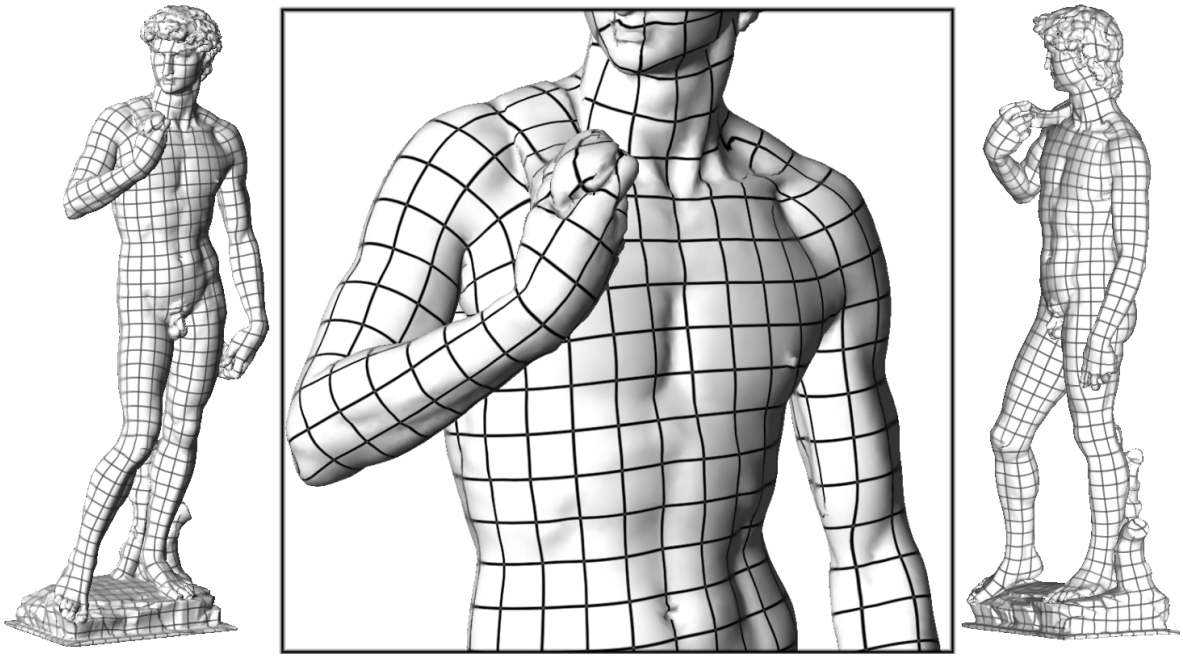


FIG. F.13: *Quasi-isometric global parameterization of the 200K facets "David" data set (iso $k\pi$ lines).*

curl-correction is applied, we obtain quasi-isometric parameterization and in this case the singularities are located where we intuitively expect them - in regions where the local-feature-size changes. Figures F.14,F.13 demonstrate that our method can work on models of any genus, as well as models with boundaries.

Table F.1 provides distortion statistics and timings for our approach. The times were measured on a 1.7 GHz machine.

For two models we compare the distortion caused by our method, with that caused by global conformal parameterization [GY03]. Both with and without curl-correction the stretch introduced by our method is drastically lower compared to the other method, while the shear is slightly larger. We also compared our results to those generated using stretch-minimizing parameterization [SGSH02] after cutting the model using [SH02]. The cuts were used to generate disk topology and reduce the stretch. The number of singularities for the models parameterized using this techniques is the number of boundary vertices, since these are the points of discontinuity in this context. Even though our parameterization technique is much more constrained due to the requirement of direction field alignment the distortion introduced by the two methods is comparable. For the Camel model, PGP introduces less distortion than stretch-minimizing parameterization, both in terms of stretch and in terms of shear. For the Bull model, our method

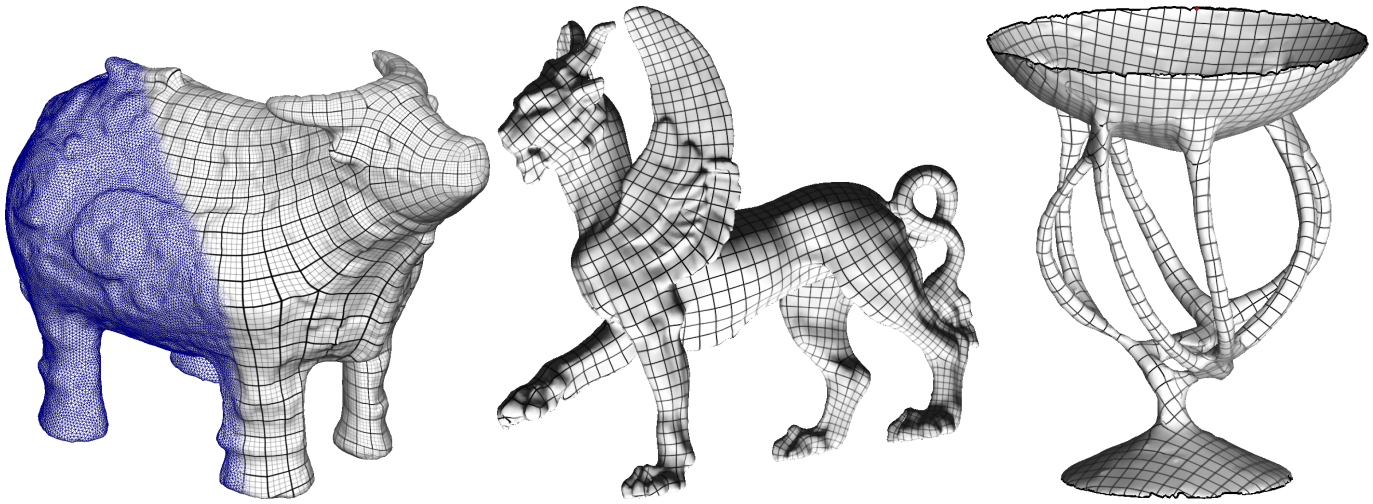


FIG. F.14: *Periodic global parameterizations with various genii (iso kπ lines).*

Model	♯Δ	Algorithm	Stretch	Shear	time
Horse	20K	Gu et al.	6.777	0.07	NA
		PGP	1.07	0.20	45 s.
		ccPGP	1.176	0.12	53 s.
Bunny	25K	Gu et al.	2.65	0.042	NA
		PGP	1.029	0.167	58 s.
		ccPGP	1.14	0.14	1 min. 12 s.
Bull	34.5K	Sander et al.	1.030	0.1558	1 min. 11 s.
		PGP	1.064	0.1774	1 min. 26 s.
		ccPGP	1.209	0.0885	1 min. 35 s.
Camel	78K	Sander et al.	1.053	0.227	3 min. 51 s.
		PGP	1.048	0.1596	5 min. 46 s.
		ccPGP	1.654	0.0711	6 min. 51 s.
David	200K	PGP	1.121	0.2398	17 min. 35 s.
		ccPGP	1.270	0.1310	20 min. 43 s.
Lion	400K	PGP	1.123	0.1728	33 min. 42 s.
		ccPGP	1.425	0.0826	45 min. 18 s.

TAB. F.1: *Statistics and timings of our method without and with curl-correction (PGP and ccPGP respectively). The numbers are compared, when data is available, to Gu et al.'s global parameterization and to Sander et al.'s method. The number of singularities for Sander et al. is the number of vertices on the cut.*

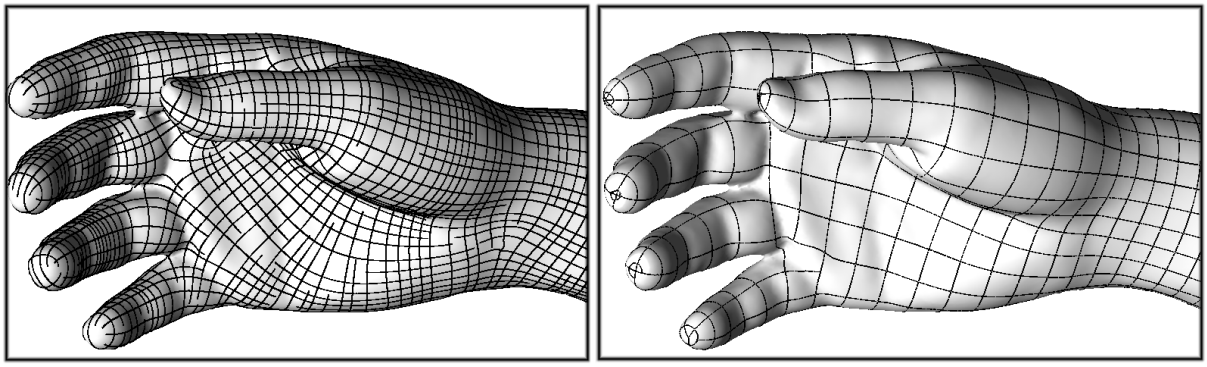


FIG. F.15: *Left : explicit remeshing using placement of lines of curvature generates an uneven sampling density and misses natural cycles ; Right : control mesh extraction using our method generates a more regular sampling and captures natural cycles.*

introduces slightly more distortion.

We extract the chart layout as an initial control mesh of the surface from the periodic global parameterizations (see e.g. Figure F.15). One can see that using our method, we do not we do not encounter the remeshing drawbacks as in [ACSD+03] (unevenly-spaced edges of the control mesh, and open loops).

F.6 Conclusion

In this chapter, aiming to extract high quality quadrilateral control meshes of triangulated surfaces of arbitrary genus, we have proposed a new periodic global parameterization. A chart layout can be extracted as the iso-lines of the global parameterization. The main advantage of the method over previous global parameterization techniques is its ability to align the parameterization with orthogonal guidance direction fields. When using the principal directions of curvatures as guidance direction fields, the extracted chart layouts are anisotropy-adapted. The parameterization is obtained by a global optimization of two periodic scalar functions so that their gradients are as tangential as possible to the guidance direction fields. This way, the quality of our control meshes are much improved over previous methods (e.g. [ACSD+03]), as the cells of our control meshes are more regularly sized and natural cycles in the original surfaces are nicely captured in the control meshes. With the automatic result of the control mesh, we provide a set of operations to improve it by manually editing.

Since our guidance direction fields are of unity norm, the resulting parameterization is isometric but at the expense of a number of N-sided charts. We have explained that through a

pre-processing of the guidance direction fields, called curl correction, we can rescale the norm of the direction field such that the curl of the resulting vector field is minimum. This process can significantly reduce the number of N-sided charts in the chart layout but the parameterization is no longer isometric.

At this point, the chart layout defines a control mesh that satisfies the T-spline control mesh validity requirements (i.e. a valid T-Mesh). Moreover, an one-to-one correspondence between the original triangulated and the T-spline surfaces is established by retrieving the parameterization for each chart. In the next chapter, we will explain how to fit a T-spline surface to the the original triangular mesh.

Annexe G

Fitting to the Original Surface

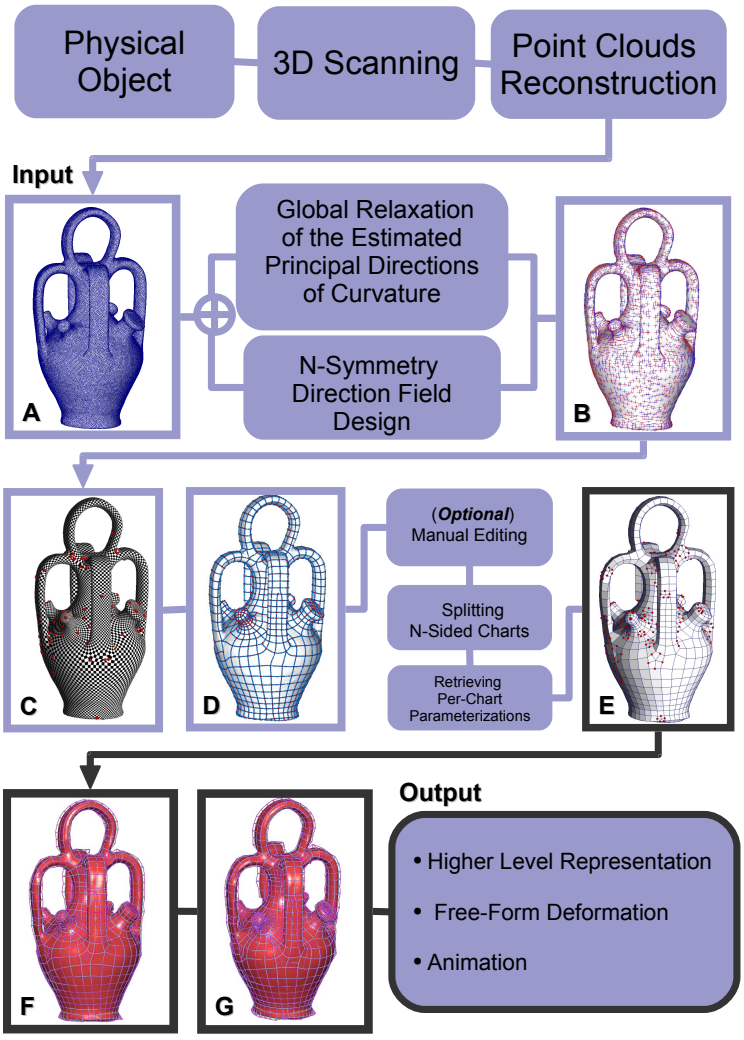


FIG. G.1: *Fitting the T-spline surface to the original one by adaptive local refinement.*

G.1 Introduction

In the previous chapter, we have explained how to obtain a valid T-Mesh using our periodic global parameterization. Moreover, an one-to-one correspondence between the original and the T-spline surfaces is established through the chart parameterizations. Let us denote this parameterization of the original triangular mesh as $M(s,t)$.

In this chapter, our goal is to find the degrees of freedom (the positions of the control points P_i) in such a way that the approximation error between the initial triangulated surface and the T-spline surface is minimized. The basic idea is to allow the control mesh to move and to find an optimal control mesh position through an optimization procedure.

Fixed degree of freedom is sometimes not sufficient to reconstruct a faithful approximation of the original surface. In this case, one needs more degrees of freedom, i.e. control points, in the T-Mesh. The idea is to add additional control points only at regions with high approximation error, i.e. local refinement of the control mesh. As pointed out in Section B, NURBSs are the most popular representation in CAGD/CAM. However, local refinement cannot be done with this representation, since a single control point cannot be inserted without propagating an entire row or column of control points (recall that a NURBS surface is defined by two NURBS curves corresponding to the s and t directions). Among schemes (as will be described in the next section) that allow local refinement of the control mesh, we have chosen T-spline representation. The main reason is due to its simplicity. Moreover, thanks to the availability of some industrial T-spline software [Ts], the fitted control meshes of our method can be employed directly in these software. Furthermore, T-spline surfaces can be easily converted to NURBS and Catmull-Clark subdivision surfaces.

Previous Work

The problem of local refinement, i.e. inserting a single control point in a quadrilateral control mesh without propagation of an entire row or column of control points has been addressed by different ways. For instances,

- in [FB88, FW98], a notion of hierarchy is introduced.
- in [EH96], special subdivision by introducing extraordinary vertices on the control mesh are introduced to help hinge the propagation of control points. However, inserting an additional vertex sometimes still requires the additional insertion of a quite large number of extra vertices.
- in [GKS02], the idea of refining the basis functions instead of the elements is introduced.

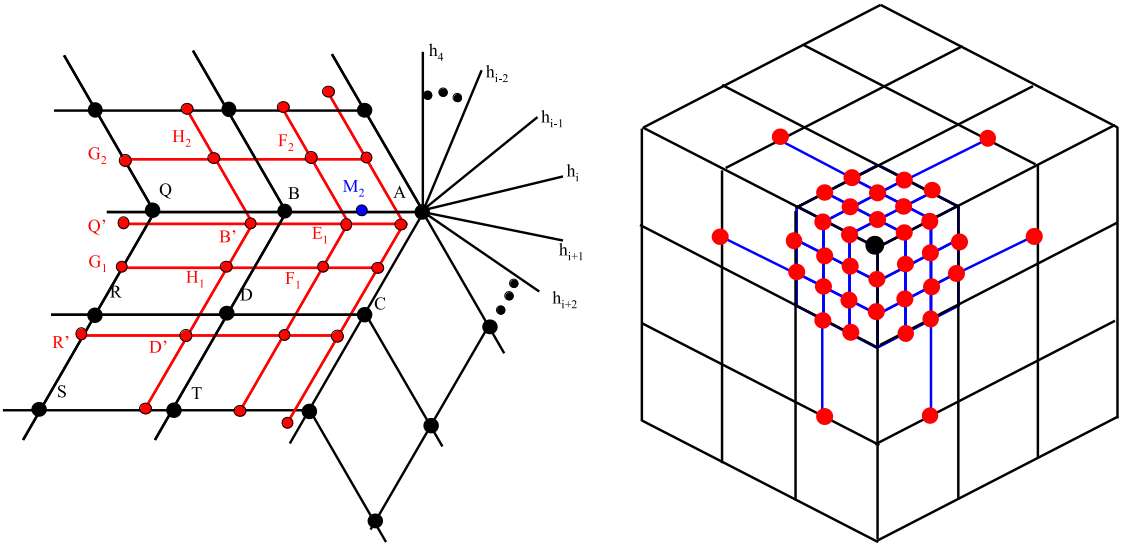


FIG. G.2: *Left : local subdivision about an valence-n control point ; Right : the neighborhood of a valence-3 extraordinary vertex, after two levels of subdivision.*

– in T-splines [SZBN03, SCF+04], thanks to the notion of point-based spline (see Section B.4), local refinement is possible since T-junctions in the control mesh are allowed.
– in [MK04c], Loop subdivision surface is used. However, the method is only limited to triangular control meshes.

The remainder of this chapter is organized as follows. We will first explain how to build a T-spline surface from an initial T-Mesh. Then, we will explain the fitting procedure (Section G.3), and how adaptive local refinement is carried out when additional control points are needed during fitting (Section G.4). Finally, we will discuss some results of the conversion in Section G.5 which is followed by conclusion.

Contributions

In the best of our knowledge, we are the first ones who study the T-spline fitting problem with surfaces of *arbitrary topology*. T-spline fitting in more specific case, e.g. terrains (disk topology), was studied in [ZWS05] to fit Z-Map models.

G.2 Extraordinary vertices and T-NURCCs

In Section B.4 in Chapter B, we have given an overview of the mathematical definition of T-spline surface. Unfortunately, the definition is not applicable for the configurations of the

extraordinary vertices of a T-Mesh.

Extraordinary vertices are vertices in the T-Mesh that are not incident on four edges. They create N-sided holes on a spline surface. Many solutions in the literature have been proposed to tackle this N-sided hole problem. For instances, in [Pra97], Prautzsch used a G^2 scheme for filling N-sided holes in a quad mesh with 4N bidegree 6 tensor product patches. In this scheme, extraordinary vertices must be separated by at least 3-rings. In [Loo04], Loop used N bidegree 7 tensor product patches to fill a N-sided hole to achieve second order continuity.

In this thesis, we have adopted the solution suggested in the original T-splines paper [SZBN03] to fill the N-sided holes by using T-NURCCs. T-NURCCs are NURCCs (Non-Uniform Rational Catmull-Clark Surfaces) with T-junctions in the spirit of T-splines. NURCCs are a generalization of both tensor product non-uniform B-spline surfaces and Catmull-Clark surfaces. Basically, NURCCs are a modification of cubic NURSSes [SZSS98]. The difference between the two representations is that NURCCs enforce the constraint that opposing edges of each four-sided face have the same knot interval. The enforcement of this constraint makes the local subdivision of NURCCs using T-junctions possible. The basic idea of T-NURCCs is simply as follows. By subsequently subdividing the T-Mesh around an extraordinary vertex, one reduces the size of the N-sided hole. This idea is best illustrated in Figure G.2. In our implementation, we conceptually apply to each extraordinary vertex two steps of local subdivision (Figure G.2-Right shows a valence-3 vertex). This generates the additional control points marked in red, which are expressed by linear combinations of the initial control points. The coefficients of these linear combinations (that depend on the valence of the vertex) are given in Sederberg *et. al*'s paper (and not repeated here). In practice, we keep a version of the original mesh, and apply local subdivision to a copy. While applying the subdivisions, we store in the newly created control points the list of original control point they depend on together with the coefficients. This representation can be directly used in the subsequent fitting steps, as explained in the next section.

G.3 Fitting

Once the T-spline surface $\mathbf{S}(s,t)$ is defined everywhere, we fit the T-spline surface to the original triangular mesh. For surface approximation, in order to measure a defined error metric, one needs to have a correspondence between the approximating and the original surface. Parameterization-free methods [MK04c, DIS03], mostly meant to fit point clouds, geometrically project each sample onto the approximating surface. Parameterization-based methods es-

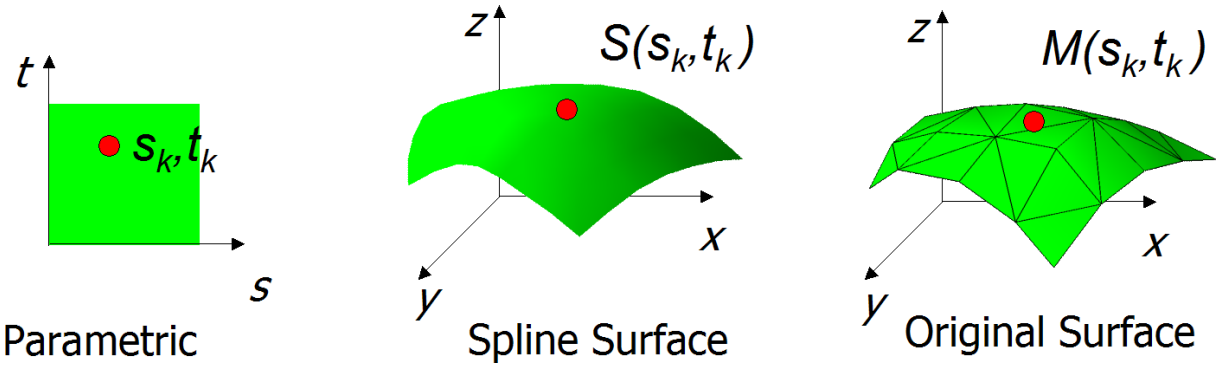


FIG. G.3: *Points on the spline and original surfaces are identified through the parametric space.*

tablish the correspondence by parameterizing the original surface and then identifying the parameter values. Our approach belongs to this latter class of methods since we have obtained the parameterization of the original triangular mesh as the chart parameterizations $\mathbf{M}(s,t)$ as explained in Chapter F.

Now that we have both the T-spline surface parametric representation $\mathbf{S}(s,t)$ and the parameterization of the triangular mesh $\mathbf{M}(s,t)$, which are both mapping of $\mathbb{R}^2 \rightarrow \mathbb{R}^3$, we minimize the following energy functional, as done in classical regularized fitting methods (see e.g. [Gre94]) :

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

$$\text{where : } \begin{cases} E_{fit} &= \int \|\mathbf{S}(s,t) - \mathbf{M}(s,t)\|^2 dsdt \\ E_{fair} &= \int \left((\frac{\partial^2 \mathbf{S}}{\partial s^2})^2 + 2(\frac{\partial^2 \mathbf{S}}{\partial s \partial t})^2 + (\frac{\partial^2 \mathbf{S}}{\partial t^2})^2 \right) dsdt \end{cases} \tag{G.1}$$

As often done in splines fitting, we approximate the fitting term E_{fit} by using a discrete set of m samples :

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

For each sample (x_k,y_k,z_k) of the original surface, (s_k,t_k) denotes its coordinate in parameter space. The natural idea would be to use the original vertices of the surface, but it is better to re-sample it so that the operation is less sensitive to the resolution of the mesh. The re-sampling is done by using a regular grid of samples in each face in the parameter space (we used 10×10 samples per face in our implementation). The corresponding points on the surface is found easily by linear interpolation in the facets. The identification of points on the original and spline surfaces is illustrated in Figure G.3.

The thin-plate energy E_{fair} avoids wiggles of the spline surface. However, if the coefficient

σ is set to be too large, the final spline surface may fit less to the original surface. In our examples, we used $\sigma = 0.05$.

Note that by construction, our control mesh and associated knot vector defines a standard (or semi-standard) T-Spline. Therefore, the denominators of the T-Spline is identically one, and we can focus on the numerator :

$$\mathbf{S}(s,t) = \sum_{i=1}^{n} \mathbf{P}_i \mathbf{B}_i(s,t)$$

Each coordinate x, y, z can be processed independently. For the x coordinate, the fitting term is given by :

$$E_{fit}^{x} = \sum_{k=1}^{m} \left(\sum_{i=1}^{n} X_i \mathbf{B}_i(s_k,t_k) - x_k \right)^2 \tag{G.2}$$

where X_i (resp Y_i, Z_i) denote the coordinates at the control point $\mathbf{P}_i$. The X_i's that minimize Equation G.2 are also the solution of a linear system $A^t AX = A^t b$, where the coefficients of the $m \times n$ matrix A are given by $a_{k,i} = B_i(s_k,t_k)$ and right-hand side by $b_k = x_k$. The unknown vector X corresponds to all the x coordinates of the control points. Adding the fairing term E_{fair}, the linear system becomes :

$$\left(A^t A + \sigma(A_{ss}^t A_{ss} + 2A_{st}^t A_{st} + A_{tt}^t A_{tt})\right) X = A^t b \tag{G.3}$$

where the coefficients $(.)_{k,i}$ of the $m \times n$ matrices A_{ss}, A_{st}, A_{tt} are the second order derivatives $B_{iss}(s_k,t_k), B_{ist}(s_k,t_k)$ and $B_{itt}(s_k,t_k)$ of the basis functions B_i respectively.

To solve the regularized fitting problem, we need first to determine the parameters (s_k,t_k) associated with the vertices of the original surface. How to obtain this parameterization is explained in Section F.3. Then, we accumulate the contributions of all the basis functions to construct the matrices $A, A_{ss}, A_{st}, A_{tt}$ and the right-hand-side b, which will be explained in the following.

G.3.1 Constructing and Solving the Linear System

To solve our regularized fitting problem (Equation G.1), the most natural way would be to proceed on a patch-by-patch basis. This would traverse the matrices $A, A_{ss}, A_{st}, A_{tt}$ row by row, and would make it possible to directly construct the final matrix of the linear system without storing these intermediate matrices.

However, constructing the pre-images of each patch is non-trivial. For this reason, we prefer to iterate on the control nodes. This means we consider one basis functions $\mathbf{B}_i(s,t)$ at a time, with a simpler pre-image. As a consequence, we store the matrices $A, A_{ss}, A_{st}, A_{tt}$ and construct

them column by column. After the traversal of all basis functions, the final matrix of the system is finally assembled (see Equation G.3).

The basis functions are piecewise defined in a neighborhood around the pre-image of each control point $\mathbf{P}_i$ (see Figure B.10). Each basis function $\mathbf{B}_i$ is completely defined by the T-Mesh and associated knot vectors around the control point $\mathbf{P}_i$. To retrieve them, we first fix arbitrary coordinates (s_0, t_0) to P_i and greedily propagate the knot intervals around it until the region of influence $D_i = [s_{i0}, s_{i4}] \times [t_{i0}, t_{i4}]$ is completely determined. The pre-image looks like the one shown in Figure B.10.

According to this local parameterization, P_i influences the T-spline patches that correspond to the faces intersected by D_i. The patch of each face is mapped to $[0,1]^2$ with $(0,0)$ set at a corner of the face. Therefore, when the influence of P_i is added to the matrices, its pre-image needs to be pre-scaled accordingly. For example, in the pre-image of the T-Mesh, if the size of the rectangle of an influenced face, F, are d and e in the s and t directions respectively, the s and t knot vectors of P_i with respect to F should be scaled by $1/d$ and $1/e$ respectively. Once the knot-vectors and region of influence D_i of the basis function B_i are determined, we update the corresponding column in the matrices A, A_{ss}, A_{st} and A_{tt}. After all control points are processed, the final matrix and right-hand side of the system are constructed. All the matrices are represented by column-major sparse data structures (CCS format, for Column Compressed Storage). We use the readily sparse direct solver TAUCS. As a consequence, the inverse of the matrix can be reused to find the X, Y and Z components of the control mesh coordinates. Note that sparse direct solvers perform so well that inversing the matrix is faster than solving a linear system with preconditioned conjugate gradient (see [BBK05], [Lev05] and the timings in the results section).

G.4 Adaptive L^∞ fitting

Global L^2 fitting operates with a fixed number of control points and thus a fixed degree of freedom is sometimes not sufficient to reconstruct the original surface. Therefore, more degree of freedom must be added. Generally, this is done by global refinement of the control mesh, which adds superfluous control points to already low approximation error regions. On the contrary, since we are using T-splines with support for local refinement, new control points can be inserted *locally* in regions of high approximation error (see Figure G.4). Thanks to the local support of T-splines, there is no need to carry out the global L^2 fitting every time a new control point is added. Only a smaller linear system needs to be solved involving only the

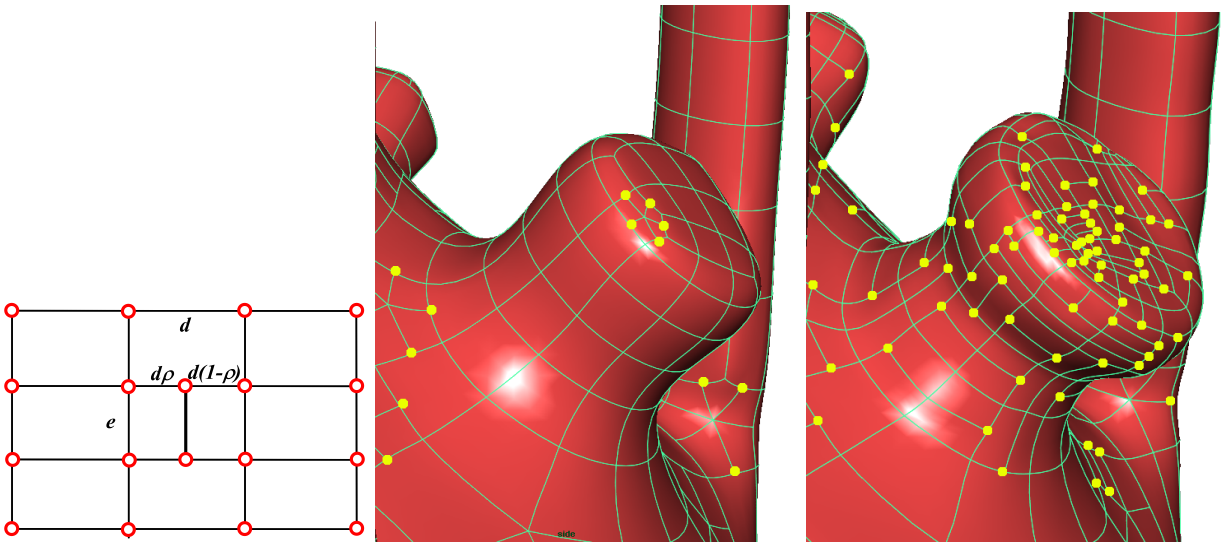


FIG. G.4: *Adaptive local refinement splits some faces to better capture complex geometry.*

patches of the control mesh affected by the local refinement operation. The L^∞ metric is defined as follows :

$$L^\infty(\mathbf{S},\mathbf{M}) = max_S \|(\mathbf{S}(s,t) - \mathbf{M}(s,t))\|^2 \tag{G.4}$$

where $\mathbf{M}(s,t)$ denotes a parameterization of the original surface. This error metric is evaluated by regularly sampling the parameter space of each face.

We iteratively apply the local refinement procedure described below to the face of worst L^∞ approximation error until it drops below a user-defined threshold.

The local refinement of T-spline is one of its invaluable properties. It is also called local knot insertion (please see [SCF+04] for more details). New control points are inserted into the T-Mesh without changing the geometry of the original T-spline surface. The algorithm recovers the T-Spline validity constraints (Section B.4) by iteratively inserting new control points :

1. Insert new control point(s) into the T-Mesh.
2. If any basis function is missing a knot dictated by Rule 1 for the current T-Mesh, perform the necessary knot insertions into that basis function.
3. If any basis function has a knot that is not dictated by Rule 1 for the current T-Mesh, add an appropriate control point into the T-Mesh.
4. Repeat Steps 2 and 3 until there are no more new operations.

The face with highest L^∞ approximation error is split into two rectangles. Knot intervals are updated accordingly (i.e. set to 0.5 for the subdivided edges). We compute the refinements

	No. of vertices	Control nodes	Control nodes (locally refined)
rocker	23k	2021	3692
botijo	41k	1471	2644
horse	10k	2046	2724
vase	2k	2120	2891

TAB. G.1: *Number of control nodes for various models.*

in both directions, and choose the one which performs best in reducing the approximation error. Note that the knot-insertion algorithm may introduce a few additional control points by propagation into the T-Mesh (see Figure G.4).

The T-spline local refinement algorithm preserves the geometry of the original T-spline surface. However, our goal is to use these newly introduced degrees of freedom to approximate better the original meshed surface. Therefore, a local fitting process is performed after the local refinement. Note that since the T-Splines function have local degrees of freedom, a smaller linear system needs to be solved. The vector X gathering all the X_i coordinates of the control nodes is split into X_f, the set of control points influenced by the new control point ($f$ree to move) and X_l, the set of control points that will remain $l$ocked. The new degrees of freedom and coupling terms on the boundary of the refined patch are determined by the sparsity pattern of the matrix A. The fitting term is given by :

$$F_{fit}(X_f) = \left\| \left[A_f | A_l\right] \left[\frac{X_f}{X_l}\right] - b \right\|^2$$

where A is split into A_f and A_l according to X_f and X_l. The new degrees of freedom are then given by the solution of the linear system :

$$A_f^t A_f X_f = A_f^t A_l X_l - A_f^t b$$

The terms introduced by the fairing energy have the same structure. Since we have a small number of coefficients and since the location of the new control points is not far away from the optimum, we use a conjugate gradient algorithm, that converges in a few iterations.

G.5 Results

Figure G.5 compares Eck and Hoppe's results with ours (note that Eck and Hoppe's images reproduced here only show patch boundaries, each patch has a 4x4 control nodes array, therefore

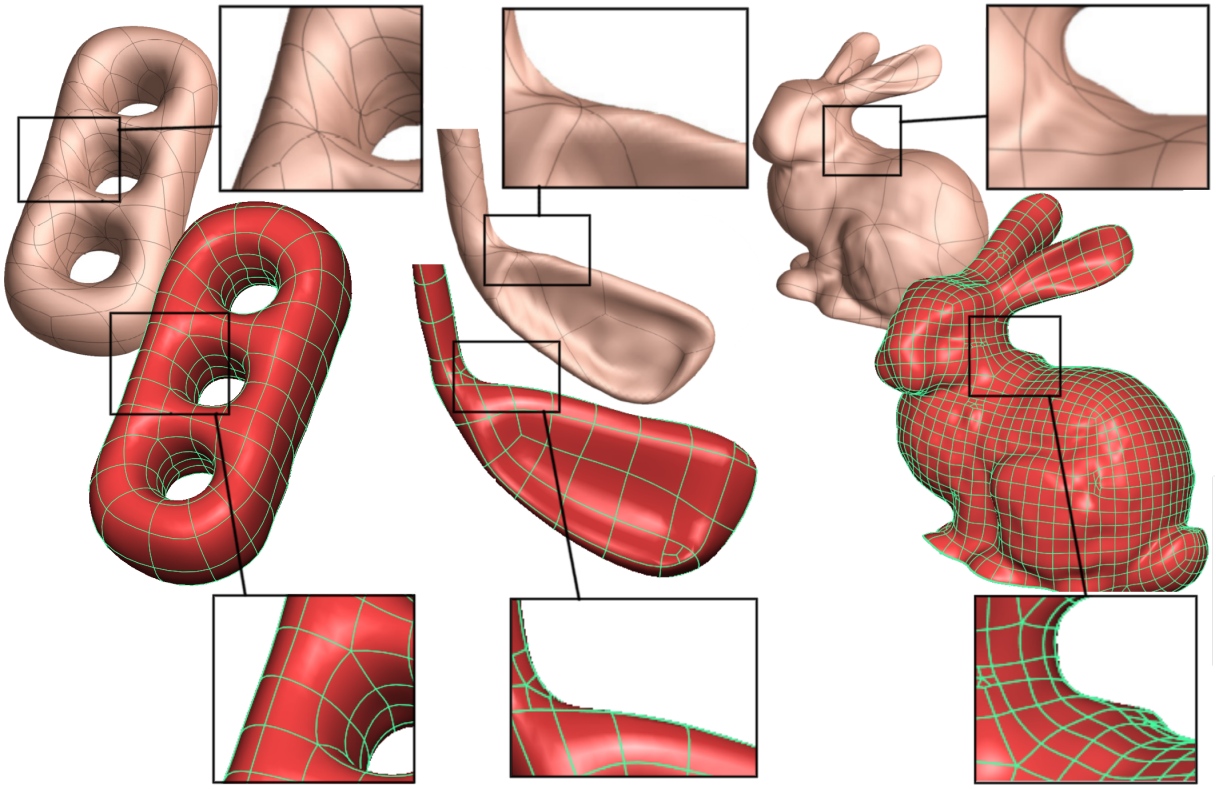


FIG. G.5: *Comparison between the results of [EH96] and ours : note how the symmetry and anisotropy are respected by our approach.*

control mesh sizes are comparable).

As can be seen, our method better respects the symmetries (see e.g. the three-holes torus) and the anisotropy of the objects, as a designer would do. As a consequence, the resulting surfaces do not have wrinkles (see closeups). We show in Figure G.7, G.8 and G.6 our method applied to data sets of various topologies and geometries. For all these examples, less than 15 edges were added by the user.

Note that the rocker (Figure G.6) is topologically equivalent to a torus. Therefore, it would be possible to create a control mesh without any singularity. However, we think that the control mesh constructed by our method is more natural, since it better takes the geometry of the object into account.

Table G.1 gives the number of control points obtained for all the models, without and with adaptive local refinement (the L^∞ threshold was set to 0.2% of the bounding box diagonal). As far as timings are concerned, the adaptive fitting algorithm did converge in less than one minute for all these models.

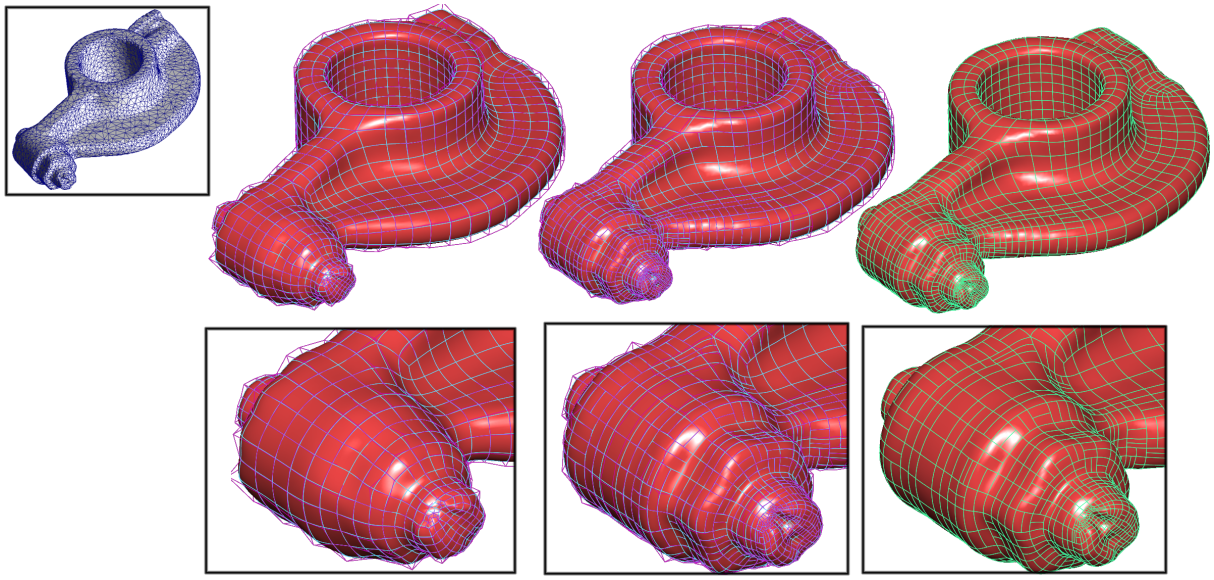


FIG. G.6: *Converting a scanned mesh into a T-Spline. From left to right : L^2 fitting, L^∞ fitting with local refinement (with and without the control mesh super-imposed).*

G.6 Conclusion

In this chapter, we have presented how to fit a T-spline surface to the original triangulated surface. The initial T-Mesh is created by using our periodic global parameterization presented in the previous chapter.

From the results, one can see that thanks to the anisotropic control meshes and the low-distortion parameterizations of the original surfaces, the spline surfaces generated after fitting these control meshes to the original surfaces have much less wrinkles as compared to spline surface fitting methods which use local parameterizations and non-anisotropy-adapted control meshes.

To achieve L^∞ fitting, we have also explained an automatic adaptive T-Mesh local refinement procedure to add extra control points in the T-Mesh to increase the degrees of freedom.

Using these automatic and manual tools, a complex model can be converted in less than 15 minutes and loaded in industrial software (see Figure **??**-E&F). This is significantly faster than fully manual solutions existing in commercial software.

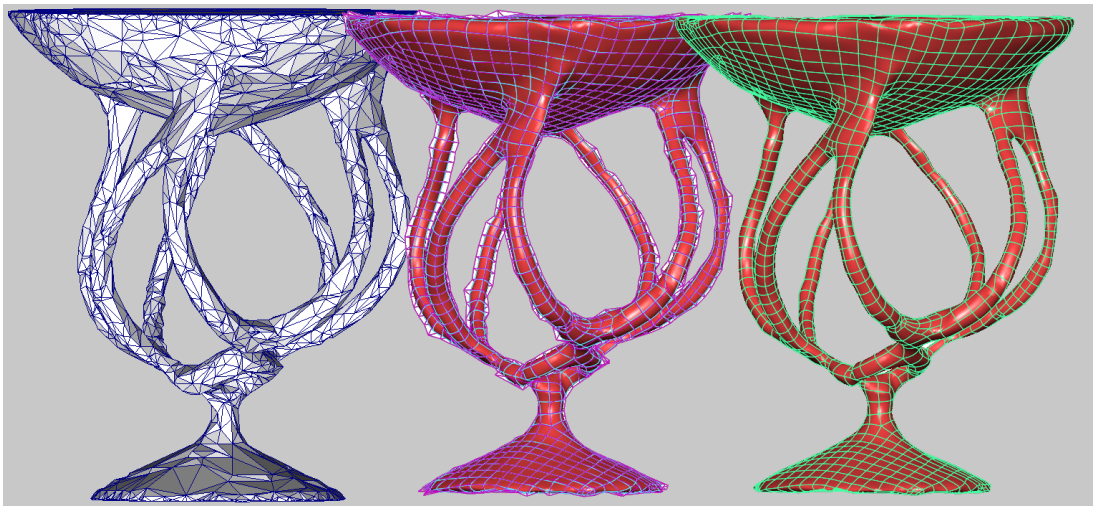


FIG. G.7: *Our method applied to a high-genus object. From left to right : initial mesh, fitted control mesh and surface. This example also shows the robustness of our method to mesh with poor quality.*

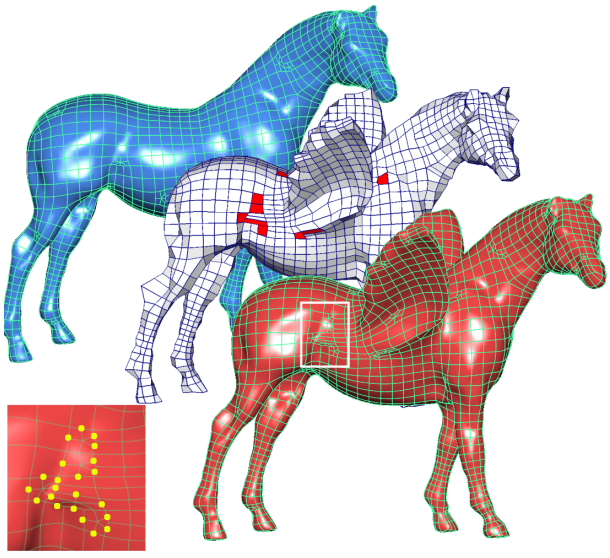


FIG. G.8: *Conversion from classical mesh models and interactive editing of T-Splines in Maya. The closeup shows how N-sided cells are replaced with T-NURCCs. We also show how this facilitates model editing (pasting the wings of the gargoyle onto the horse).*

Conclusion

Parametric representations of surfaces, and more specifically tensor-product splines, are widely used in CAGD (Computer-Aided Geometric Design) and CAM (Computer-Aided Manufacturing). In these representations, the geometry of a surface is usually defined by a quadrilateral control mesh.

In the context of CAGD/CAM, in order to obtain a spline representation of a real object, one must find a good quadrilateral control mesh from it. In this thesis, we have introduced two criteria for such good control meshes :

– the edges should be orthogonal, and
– aligned with the principal directions of curvature of the surface (anisotropy-adapted).

However, to obtain control meshes with such qualities is *non-trivial*. In this thesis, we have invented a new automatic algorithm, called *periodic global parameterization* (see Section F.2 and results published in [RLL$^+$]) to convert from the output of the 3D scanning process (a triangular mesh) into a good control mesh. The control meshes thus obtained have not only fulfilled the above two criteria but also shown better remeshing results than previous methods.

The basic idea of our new algorithm is to find a "geometry-meaningful" parameterization guided by a pair of orthogonal anisotropic direction fields. Then, the iso-value lines of this parameterization will be extracted, using our new embedded cellular complex data structure (see Chapter C and results published in [LLP05]), to define an initial control mesh.

To obtain anisotropic guidance direction fields, we have introduced two new methods :

– global relaxation of the estimated principal directions of curvature (see Section D.4 and results published in [RLL$^+$]) ;
– design of direction fields by direct singularity control (see Chapter E and results published in [RVLL06, LV$^+$06]).

With the initial control mesh, we have explained how to construct a T-spline approximation of the initial triangulated surface through a L^∞ fitting by adaptive local refinement (see Chapter G and results published in [LRL06]). Among all the possible parametric representations, we chose T-spline since it supports local refinement and since it is compatible with standard tensor-product splines used in CAGD/CAM.

We have discussed some results of the triangular mesh to T-spline conversion (see Chapter G). From the results, one can see that thanks to the anisotropy-adapted control meshes, and the low-distortion parameterizations of the original surfaces, the spline surfaces generated after fitting these control meshes to the original surfaces have much less wrinkles as compared to previous spline surface fitting methods which use local parameterizations and non-anisotropy-adapted control meshes.

In this thesis, we have proposed a method for automatic and interactive mesh to T-spline conversion. Our algorithm proposes an initial solution, that can be manually refined by the user. Using these automatic and manual tools, a complex model can be converted in less than 15 minutes and loaded in industrial software. This is significantly faster than fully manual solutions existing in commercial software.

Our control mesh extraction method is a step forward towards an approach that mimics the way a skilled designer will design a control mesh by taking into account the geometry of the object. There are several directions for the future work. First, we think that a better mathematical characterization of the singularities in the periodic global parameterization may avoid appearing of N-sided charts in the chart layout, and hence lead to a faster and fully automatic solution. Second, for specific applications of the method (e.g. reverse engineering [Rap]), improving our method to allow creating a control mesh that adapts well to the sharp edges of the original triangular mesh (e.g. a car engine component) may be very useful.

Annexe H

H.1 Turning Numbers Fundamental Properties

We give here an outline of the proofs of two fundamental turning number properties.

H.1.1 Boundary Property

The boundary property states that for any direction field on a 2-manifold S, we have $T(\partial S) + \chi(S) = 0$ which generalizes the Poincaré Hopf theorem to N-symmetry direction fields. We first establish three simple results, then prove the equation by structural induction.

Lemma H.1.1. *The reversal of a cycle has opposite turning number :*

$$T(-\gamma) = -T(\gamma) \tag{H.1}$$

Proof : integrating in opposite direction changes ds in $-ds$ □

Lemma H.1.2. *If we call A and B two sub-manifolds of S_h (see Figure H.1), the turning numbers of ∂A and ∂B are linked by the equation :*

$$T(\partial A) + T(\partial B) = T(\partial(A \cup B)) + T(\partial(A \cap B)) \tag{H.2}$$

Proof : The term in $\kappa_{\vec{d}}$ is preserved because integrated on the same set $\partial A \cup \partial B = \partial(A \cup B) \cup \partial(A \cap B)$. The term in κ_γ, is preserved by application of the Gauss-Bonnet formula to the equality $\int_A + \int_B = \int_{A \cup B} + \int_{A \cap B}$. □

Lemma H.1.3. *Contractible boundaries have a turning number of -1.*

Proof : Let γ^C be a contractible boundary. It is by definition the boundary of a topological disk D, so there exists a continuous bijective application $p : D \to \mathscr{D}_2$ where $\mathscr{D}_2$ is the unit disk

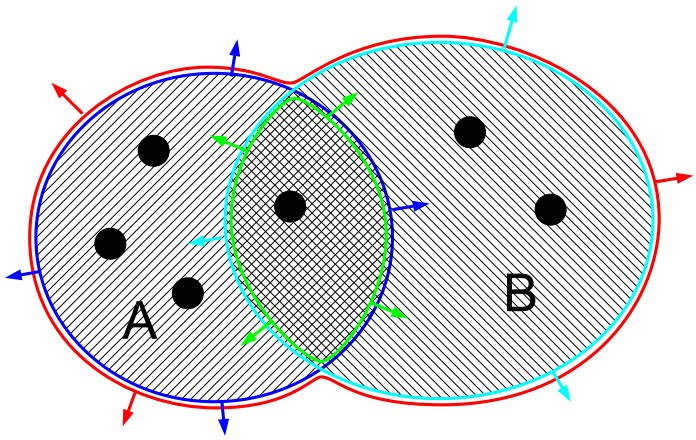


FIG. H.1: *Equivalence between $\partial A \cup \partial B$ and $\partial(A \cup B) \cup \partial(A \cap B)$*

in $\mathbb{R}^2$. Using this application, we can define the image field $\vec{d'} = p(\vec{d})$ of the direction field on $\mathscr{D}_2$.

$$T_{\vec{d'}}(\partial \mathscr{D}_2) = \frac{1}{2\pi} \oint_{\partial \mathscr{D}_2} (\kappa_{\vec{d'}} - \kappa_\gamma) ds = \frac{1}{2\pi} \int_{s=0}^{2\pi} (\frac{\partial \vec{d'}}{\partial s} \cdot \vec{d'}^{\perp} - \frac{\partial \vec{t}_\gamma}{\partial s} \cdot \vec{n}_\gamma) ds$$

Because we are on the unit disk, the arclength s is equivalent to the angle on the disk boundary, so we have $\vec{t}_\gamma = (\cos(s), \sin(s))^T$ and $\vec{n}_\gamma = (-\sin(s), \cos(s))^T$, which gives :

$$\kappa_\gamma = \frac{\partial \vec{t}_\gamma}{\partial s} \cdot \vec{n}_\gamma = 1$$

Let $\vec{x} = (x = \cos(s), y = \sin(s))^T$ be the position vector on the disk boundary. We have :

$$\kappa_{\vec{d'}} ds = \frac{\partial \vec{d'}}{\partial s} \cdot \vec{d'}^{\perp} ds = \left(\frac{\partial \vec{d'}}{\partial x} \cdot \vec{d'}^{\perp}, \frac{\partial \vec{d'}}{\partial y} \cdot \vec{d'}^{\perp} \right)^T d\vec{x}$$

we can easily verify that the first term is curl free, hence its integral on a closed loop is null. We can now compute the turning number :

$$T_{\vec{d'}}(\partial \mathscr{D}_2) = T_{\vec{d}_p}(\partial \mathscr{D}_2) = \frac{1}{2\pi} \oint_{\partial \mathscr{D}_2} (\kappa_{\vec{d'}} - \kappa_\gamma) ds = \oint_{\partial \mathscr{D}_2} \nabla \theta d\vec{x} - \frac{1}{2\pi} \int_{s=0}^{2\pi} 1 \cdot ds = -1 \square$$

These lemmas allow us to give a proof of Theorem E.2.1 :

$$T_{\vec{d}}(\partial S) + \chi(S) = 0 \tag{H.3}$$

Proof : It is well known in topology that any orientable surface (with borders) can be cut along g cycles to obtain a sphere (with borders). Hence we start by proving (H.3) for a sphere with $b > 0$ borders.

If S is a topological disk ($b = 1, \chi = 1$), its boundary is contractible so its turning number is -1 by Lemma H.1.3 which proves (H.3) for $b = 1$.

If S is a topological cylinder ($b = 2, \chi = 0$), its boundary is composed of 2 borders : $\partial S = \gamma_1 + \gamma_2$. γ_1 and $-\gamma_2$ are trivially homotopic so they have the same turning numbers, so by (H.1) we have :

$$T(\partial S) = T(\gamma_1) + T(\gamma_2) = T(\gamma_1) - T(-\gamma_2) = 0$$

which proves (H.3) for $b = 2$.

For higher numbers of borders, we prove the property by reccurence : assume (H.3) is true $\forall b \leq B$, and let S have $B+1$ borders $\gamma_1 ... \gamma_{B+1}$. We apply (H.2) where A and B contain γ_B and γ_{B+1} and intersect in a topological disk. $S \backslash A \cup B$ has B borders, so it satisfies (H.3) :

$$T(\partial(S \backslash A \cup B)) = \sum_{i=1}^{B-1} T(\gamma_i) + T(\partial(A \cup B)) =$$

$$\sum_{i=1}^{B-1} T(\gamma_i) + T(\partial A) + T(\partial B) - T(\partial(A \cap B)) =$$

$$\sum_{i=1}^{B+1} T(\gamma_i) + 1 = \chi(S \backslash A \cup B) = \chi(S) + 1$$

which proves (H.3) by reccurence as we have proved it for $B = 2$.

We can now get to the most general case where S has genus g and b borders $\gamma_1 ... \gamma_b$. By definition of the genus, there exists a family $\gamma_1^G, ... \gamma_g^G$ of single cycles such that $S \backslash \{\gamma_1^G, ... \gamma_g^G\}$ is connected. This operation does not change the Euler characteristic of the surface ($b \leftarrow b + 2g$, $g \leftarrow 0$ preserves $\chi = 2 - 2g - b$). The result is a sphere S_{cut} with $b + 2g$ borders $\gamma_1, ... \gamma_b, \gamma_1^G, -\gamma_1^G, ... \gamma_g^G, -\gamma_g^G$, so we can apply (H.3) :

$$T(\partial S_{cut}) = \sum_{i=1}^{n} T(\gamma_i) + \sum_{i=1}^{g} T(\gamma_i^G) + T(-\gamma_i^G) = 2 - 2g - b$$

as the second sum is null by (H.1). This finally proves Theorem E.2.1 in the general case □

H.1.2 Topological Equivalence

We will need a simple lemma to prove the topological equivalence :

Lemma H.1.4.

$$T_{\vec{d}_1}(\gamma) = T_{\vec{d}_2}(\gamma) \quad \forall \gamma \in \mathscr{C}(S) \Leftrightarrow T_{\vec{d}_1}(\gamma) = T_{\vec{d}_2}(\gamma) \quad \forall \gamma \in H(S)$$

where $\mathscr{C}(S)$ is the set of all cycles on S and $H(S)$ is a homology basis on S. In other words, the turning numbers of all cycles on S depend only on the turning numbers of a homology basis.

Proof : The direct way is trivial as $H(S) \subset \mathscr{C}(S)$. For the other implication, let $\gamma \in \mathscr{C}(S)$ be a cycle. By definition of homology basis, γ is homologic to a cycle $\sum_i a_i \gamma_i^B$ where $\gamma_i^B \in H(S)$ are basis cycles, such that $\sum_i a_i \gamma_i^B - \gamma$ is a boundary. Hance, by Theorem E.2.1, we have :

$$T_{\vec{d}_0}(\gamma) = \sum_i a_i T_{\vec{d}_0}(\gamma_i^B) + \chi(S) = \sum_i a_i T_{\vec{d}_1}(\gamma_i^B) + \chi(S) = T_{\vec{d}_1}(\gamma)$$

because the turning numbers are equal along the cycles of the homology basis. □

We can now prove the Theorem E.2.2 which states that two direction fields are homotopic if and only if they have the same turning numbers along the cycles of a homology basis of S.

Proof : The direct implication is trivial because turning numbers are preserved by homotopy. For the reciproque, let $\vec{d}_0$ and $\vec{d}_1$ be two fields with the same turning numbers along the cycles of a homotopy basis, let O be a point in S, and θ_0 be the angle between $\vec{d}_0$ and $\vec{d}_1$ at O. Let P be any other point in S, and γ_0 and γ_1 be two paths between O and P. Let us define $\theta_i(P) = \theta_0 + \int_{\gamma_i} \kappa_{\vec{d}_1} - \kappa_{\vec{d}_0}$. By Lemma H.1.4, $\vec{d}_0$ and $\vec{d}_1$ have the same turning numbers along all cycles, so in particular for the cycle $\gamma_{1-0} = \gamma_1 - \gamma_0$, such that :

$$\theta_1(P) - \theta_0(P) = \oint_{\gamma_{0-1}} \kappa_{\vec{d}_1} - \kappa_{\vec{d}_0} = \oint_{\gamma_{0-1}} \kappa_{\gamma_{0-1}} - \kappa_{\vec{d}_1} - \oint_{\gamma_{0-1}} \kappa_{\gamma_{0-1}} - \kappa_{\vec{d}_1} = T_{\vec{d}_1}(\gamma_{1,2}) - T_{\vec{d}_2}(\gamma_{1,2}) = 0$$

This proves that θ_i is independent of the choice of the path to integrate, so we can define a continuous function θ such that $\vec{d}_2 = R(\vec{d}_1, \theta)$ where R is the rotation around the surface normal. This allows us to build a continuous function $\Gamma : [0,1] \to \mathscr{D}_N(S)$ defined by $\vec{d}_2 = R(\vec{d}_1, t\theta)$, such that $\Gamma(0) = \vec{d}_0$ and $\Gamma(1) = \vec{d}_1$ which proves that the direction fields are homotopic. □

Bibliographie

[ACSD+03] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Levy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics (SIGGRAPH Proceedings '03)*, 22(3) :485–493, 2003.

[Bau75] B. Baumgart. A Polyhedron Representation for Computer Vision. In *Proceedings of AFIPS National Computer Conference*, volume 44, pages 589–596, June 1975.

[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.

[BG89] P. Baudelaire and M. Gangnet. Planar maps : An interaction paradigm for graphic design. In *Proceedings of CHI*, pages 313–318, 1989.

[BMRJ04] Ioana Boier-Martin, Holly Rushmeier, and Jingyi Jin. Parameterization of triangle meshes over quadrilateral domains. In *Proceedings of EuroGraphics SIGGRAPH Symposium on Geometry Processing*, pages 197–207, 2004.

[Boo72] C. De Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6 :50–62, 1972.

[BSH97] H. Battke, D. Stalling, and H. Hege. Fast line integral convolution for arbitrary surfaces. In *Visualization and Mathematics*, pages 181–195, Heidelberg, 1997. Springer-Verlag.

[Car76] Manfredo Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[CGAL] CGAL Computational Geometry Algorithms Library. `http://www.cs.ruu.nl/CGAL/index.html`.

[CH90] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Proceedings of Sixth ACM Symposium on Computational Geometry*, 1990.

[CH02] N. A. Carr and J. C. Hart. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics*, 21(2) :106–131, April 2002.

[CL93] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 263–272, 1993.

[Cox72] M.G. Cox. The numerical evaluation of b-splines. *Journal of Inst. Mathss. Applics.*, 10 :134–149, 1972.

[CSM03] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of SoCG*, 2003.

[Cyb] Cyberware. `http://www.cyberware.com/products/cyslice.html`.

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

[DBG+06] S. Dong, P-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. In *ACM TOG (SIGGRAPH)*, 2006.

[DH94] T. Delmarcelle and L. Hesselink. The topology of symmetric, second-order tensor fields. In *Proceedings of IEEE Visualization*, pages 140–147, 1994.

[DIS03] N. Dodgson, I. Ivrissimtzis, and M. Sabin. Curve and surface fitting. Nashboro Press, 2003.

[DKG05] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 22(5) :392–423, 2005.

[DKT05] Mathieu Desbrun, Eva Kanso, and Yiying Tong. Discrete differential forms for computational modeling, 2005. Chapter in ACM SIGGRAPH'06 Course Notes on Discrete Differential Geometry.

[DKT06] Mathieu Desbrun, Eva Kanso, and Yiying Tong. Discrete differential forms for computational modeling. Chapter in ACM SIGGRAPH'06 Course Notes on Discrete Differential Geometry, 2006.

[DMLC02] J.-M. Dichler, K. Maritaud, B. Levy, and D. Chazanfarpour. Texture particles. *Computer Graphics Forum*, 21(3), 2002.

[Edm60] J. Edmonds. A Combinatorial Representation for Polyhedral Surfaces. *Notices Amer. Math. Soc.*, 7, 1960.

[EH96] M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proceedings of ACM SIGGRAPH*, 1996.

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

[FB88] D. Forsey and R. Bartels. Hierarchical B-spline refinement. *Computer Graphics (Proceedings ACM SIGGRAPH 1988)*, 22 :205–212, 1988.

[FC95] L. Forssell and S. Cohen. Using line integral convolution for flow visualization : Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2) :133–141, June 1995.

[FH05] M. S. Floater and K. Hormann. Surface parameterization : a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, Heidelberg, 2005.

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

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

[For94] L. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proceedings of IEEE Visualization*, pages 240–247, 1994.

[FW98] D. Forsey and D. Wong. Multiresolution surface reconstruction for hierarchical B-splines. In *Graphics Interface*, pages 57–64, 1998.

[GGH02] X. Gu, S. Gortler, and H. Hoppe. Geometry images. *ACM TOG (SIGGRAPH)*, pages 355–361, 2002.

[GGS03] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM TOG (SIGGRAPH)*, 22 :358–363, 2003.

[GGSC96] S. Gortler, R. Grzeszczuk, R. Szeliski, and M.-F. Cohen. The lumigraph. In *Proceedings of ACM SIGGRAPH*, pages 43–54, 1996.

[GGT06] S.J. Gortler, C. Gotsman, and D. Thurston. One-forms on meshes and applications to 3d mesh parameterization. *Computer Aided Geometric Design*, 23(2) :83–112, 2006.

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

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Computer Graphics*, 31(Annual Conference Series) :209–216, 1997.

[GHPT89] Michel Gangnet, Jean-Claude Hervé, Thierry Pudet, and Jean-Manuel Van Thong. Incremental computation of planar maps. *Computer Graphics*, 23(3) :345–354, 1989.

[GI04] Jack Goldfeather and Victoria Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1) :45–63, 2004.

[GKS02] E. Grinspun, P. Krysl, and P. Schroder. CHARMS : a simple framework for adaptive simulation. In *Proceedings of ACM SIGGRAPH*, pages 281–290, 2002.

[GL05] M. Grabner and R. S. Laramee. Image space advection on graphics hardware. In *Proceedings of the 21st Spring Conference on Computer Graphics and its Applications*, pages 75–82, 2005.

[Gou71a] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20 :623–628, 1971.

[Gou71b] Henri Gouraud. *Computer display of curved surfaces*. PhD thesis, University of Utah, 1971.

[Gra] Graphite. `http://www.loria.fr/$\sim$levy/Graphite/index.html`.

[Gre94] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum (Proceedings EuroGraphics 1994)*, 13(3) :143–154, 1994.

[GS85] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4 :74–123, April 1985.

[GY03] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 127–137, 2003.

[Hat01] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

[HG00] K. Hormann and G. Greiner. MIPS : An efficient global parametrization method. In P.-J. Laurent, P. Sablonni‘ere, and L. L. Schumaker, editors, *Curve and Surface Design*. Vanderbilt University Press, 2000.

[HK06] A. Hornung and L. Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *EG/ACM Symposium on Geometry Processing*, 2006.

[Hop96] Hugues Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series) :99–108, 1996.

[HP04] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *ACM TOG (SIGGRAPH)*, pages 284–293, 2004.

[HZ00] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH*, pages 517–526, 2000.

[ICE] ICEM. `http://www.icem.com/`.

[Int97] V. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 109–116, 1997.

[JL97] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings of the Eurographics Workshop. Visualization in Scientific Computing*, pages 43–56, 1997.

[JWGY04] M. Jin, Y. Wang, X. Gu, and S.-T. Yau. Optimal global conformal surface parameterization for visualization. In *Proceedings of IEEE Visualization*, pages 267–274, 2004.

[Kap99] S. Kapoor. Efficient computation of geodesic shortest paths. In *Proceedings of ACM Symposium on Theory of Computing*, 1999.

[KBH06] M. Kazhdan, M. Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *EG/ACM Symposium on Geometry Processing*, 2006.

[Ket98] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proceedings of Symposium on Computational Geometry*, pages 146–154, 1998.

[KL96] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. *Computer Graphics*, 30(Annual Conference Series) :313–324, 1996.

[KLS03] Andrei Khodakovsky, Nathan Litke, and Peter Schroder. Globally smooth parameterizations with low distortion. *ACM TOG (SIGGRAPH)*, 2003.

[KO00] B. Kaneva and J. O'Rourke. An implementation of chen and han's shortest paths algorithm. In *Proceedings of Canadian Conference on Computational Geometry*, 2000.

[KS98] R. Kimmel and J. Sethian. Fast marching methods on triangulated domains. In *Proceedings of National Academy of Sciences, vol. 95*, pages 8341–8435, 1998.

[KS01] T. Kanai and H. Suzuki. Approximate shortest path on a polyhederal surface and its application. *Computer-Aided Design*, 33 :801–811, 2001.

[KS04] V. Kraevoy and A. Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM TOG (SIGGRAPH)*, 23(3), 2004.

[Lev05] Bruno Levy. Numerical methods for digital geometry processing. In *Israel Korea Bi-National Conference*, 2005.

[Lie94] P. Lienhardt. N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds. *Journal on Computational Geometry and Applications*, 4(3) :275–324, 1994.

[LLP05] Wan Chiu Li, Bruno Levy, and Jean-Claude Paul. Mesh editing with an embedded network of curves. In *IEEE International Conference on Shape Modeling and Applications*, 2005.

[Loo04] C. Loop. Second order smoothness over extraordinary vertices. In *Proceedings of the EuroGraphics Symposium on Geometry Processing*, pages 169–178, 2004.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project : 3D scanning of large statues. In *Proceedings of ACM Siggraph*, pages 131–144, 2000.

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM TOG (SIGGRAPH)*, pages 362–371, 2002.

[LRL06] Wan-Chiu Li, Nicolas Ray, and Bruno Levy. Automatic and interactive mesh to t-spline conversion. In *EG/ACM Symposium on Geometry Processing*, 2006.

[LSH03] R. S. Laramee, J. Schneider, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *Proceedings of IEEE Visualization*, pages 131–138, 2003.

[LV+06] Wan-Chiu Li, , Bruno Vallet, Nicolas Ray, and Bruno Levy. Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization '06)*, 12(5), September-October 2006.

[LvWJH04] R. S. Laramee, J. van Wijk, B. Jobard, and H.Hauser. ISA and IBFVS : Image space-based visualization of flow on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6) :637–648, 2004.

[MAD05] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization*, pages 479–486, 2005.

[Mas91] W.S. Massey. *A Basic Course in Algebraic Topology*. Springer Verlag, 1991.

[MBVW95] Michael J. Milroy, Colin Bradley, Geoffrey W. Vickers, and D. J. Weir. G1 continuity of b-spline surface patches in reverse engineering. *Computer-Aided Design*, 27(6) :471–478, 1995.

[McG] M. McGuire. The half-edge data structure. `http://www.flipcode.com/articles/article_halfedge.shtml`.

[MDSB02] Mark Meyer, Mathieu Desbrun, Peter Schroder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proceedings of VisMath*, 2002.

[MK04a] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proceedings of Pacific Graphics*, pages 207–216, 2004.

[MK04b] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proceedings of Pacific Graphics*, pages 207–216, 2004.

[MK04c] Martin Marinov and Leif Kobbelt. Optimization techniques for approximation with subdivision surfaces. In *ACM Symposium on Solid Modeling and Applications*, pages 113–122, 2004.

[MK06] M. Marinov and L. Kobbelt. A robust two-step procedure for quad-dominant remeshing. *Computer Graphics Forum (Eurographics 2006 proceedings)*, 25(3) :537–546, 2006.

[MMP87] J.S.B. Mitchell, D.M. Mount, and C.H. Paradimitriou. The discrete geodesic problem. *SIAM J. Computing*, 16(4) :647–668, 1987.

[NC99] F. Neyret and M. P. Cani. Pattern-based texturing revisited. In *Proceedings of ACM SIGGRAPH*, 1999.

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

[NGH04] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics (SIGGRAPH Proceedings '04)*, 23(3) :613–622, 2004.

[Nie79] G. Nielson. The side-vertex method for interpolation in triangles. *Journal of Approximation Theory*, 25(4) :318–336, 1979.

[OHB01] Y. Ohtake, M. Horikawa, and A. Belyaev. Adaptive smoothing tangential direction fields on polygonal surfaces. In *Proceedings of Pacific Graphics*, page 189, 2001.

[Pet01] S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 2, 2001.

[PFH00] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of ACM SIGGRAPH*, pages 465–470, 2000.

[PH97] Jovan Popovic and Hugues Hoppe. Progressive simplicial complexes. In *Proceedings of ACM SIGGRAPH*, pages 217–224, 1997.

[PP02] K. Polthier and E. Preuss. Identifying vector fields singularities using a discrete hodge decomposition. In H.C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 113–134. Springer Verlag, 2002.

[Pra97] H. Prautzsch. Freeform splines. *Computer Aided Geometric Design 14*, 14 :201–206, 1997.

[Rap] RapidFrom. `http://www.rapidform.com/`.

[RLL+] N. Ray, W.C. Li, B. Levy, A. Sheffer, and P. Alliez. Global periodic parameterization. *ACM Transactions on Graphics (to appear)*.

[RSHCE99] C. Rezk-Salama, P. Hastreiter, T. Christian, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In *Proceedings of IEEE Visualization*, pages 233–240, 1999.

[RVLL06] N. Ray, B. Vallet, W.C. Li, and B. Levy. N-symmetry direction fields on surfaces of arbitrary genus. Technical report, INRIA-ALICE, January 2006.

[SCF+04] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM TOG (SIGGRAPH)*, 2004.

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

[SF05] D. Steiner and A. Fischer. Planar parameterization for closed manifold genus-g meshes using any type of positive weights. *Journal of Computing and Information Science in Engineering (JCISE)*, 5, 2005.

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

[SH95] D. Stalling and H. Hege. Fast and resolution independent line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 249–256, 1995.

[SH02] A. Sheffer and J. Hart. Seamster : Inconspicuous low-distortion texture seam layout. In *Proceedings of IEEE Visualization*, 2002.

[SPPH04] J. Schreiner, A. Prakash, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM TOG (SIGGRAPH)*, 23(3), 2004.

[SSK+05] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, and H. Hoppe. Fast exact and approximate geodesic on meshes. *ACM TOG (SIGGRAPH)*, 2005.

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

[SZSS98] T. W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform subdivision surfaces. In *Proceedings of ACM SIGGRAPH*, 1998.

[Tau95] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of International Conference on Computer Vision*, pages 902–907, 1995.

[TB96] Greg Turk and David Banks. Image-guided streamline placement. *Computer Graphics*, 30(Annual Conference Series) :453–460, 1996.

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

[The02] H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Proceedings Eurographics 2002)*, 21(3) :595–604, 2002.

[TL04] Rodrigo Toledo and Bruno Levy. Extending the graphic pipeline with new gpu-accelerated primitives. Technical report, INRIA, 2004.

[TLHD03] Yiying Tong, Santiago Lombeyda, Anil Hirani, and Mathieu Desbrun. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics (Proceedings SIGGRAPH '03)*, 22(3) :445–452, 2003.

[Tri02] X. Tricoche. *Vector and Tensor Topology Simplification, Tracking, and Visualization*. PhD thesis, Schriftenreihe Fachbereich Informatik (3), University of Kaiserslautern, 2002.

[TRZS04] H. Theisel, C. Rossl, R. Zayer, and H.-P. Seidel. Normal based estimation of the curvature tensor for triangular meshes. In *Proceedings of Pacific Graphics '04*, pages 288–297, 2004.

[Ts] T-spline. `http://www.tspline.com`.

[TSH00] X. Tricoche, G. Scheuermann, and H. Hagen. Higher order singularities in piecewise linear vector fields. In *Proceedings of IMA Conference on the the Mathematics of Surfaces*, pages 99–113, 2000.

[TSH03] X. Tricoche, G. Scheuermann, and H. Hagen. Topology simplification of symmetric, second-order 2D tensor fields. In *Hierarchical and Geometrical Methods in Scientific Visualization*. Springer, 2003.

[Tur01] Greg Turk. Texture synthesis on surfaces. In *Proceedings of ACM SIGGRAPH*, pages 347–354, 2001.

[VKP00] Vivek Verma, David T. Kao, and Alex Pang. A flow-guided streamline seeding strategy. In *IEEE Visualization*, pages 163–170, 2000.

[vW03] J.J. van Wijk. Image based flow visualization for curved surfaces. In *Proceedings of IEEE Visualization*, pages 123–130, 2003.

[Wat93] A. Watt. *3D Computer Graphics*. Addison-Wesley, 1993.

[Wei] Eric W. Weisstein. Manifold.

[Wei85] K. Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE CG&AComputer-Aided Design*, 5(1) :21–40, 1985.

[WL01] Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of ACM SIGGRAPH*, pages 355–360, 2001.

[WW94] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *Proceedings of ACM SIGGRAPH*, pages 247–256, 1994.

[WWT+06] Ke Wang, Weiwei, Yiying Tong, Mathieu Desbrun, and Peter Schröder. Edge subdivision schemes and the construction of smooth vector fields. *ACM TOG (SIGGRAPH)*, 2006.

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

[ZG04] S. Zelinka and M. Garland. Jump map-based interactive texture synthesis. *ACM Transactions on Graphics*, 23(4) :930–962, 2004.

[ZHT05] E. Zhang, J. Hays, and G. Turk. Interactive design and visualization of tensor fields on surfaces. Technical report, Oregon State University, 2005.

[ZMT04] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. Technical report, GVU 04-16, Georgia Institute of Technology, 2004. Accepted by ACM Transactions on Graphics, pending revision.

[ZWS05] J. Zheng, Y. Wang, and H. S. Seah. Adaptive t-spline surface fitting to z-map models. In *Proceedings of GRAPHITE*, 2005.

Résumé

Afin de convertir un maillage triangulaire en une surface spline de CAGD/CAM, cette thèse adresse l'un des problèmes les plus cruciaux du processus de conversion : extraire un "bon" maillage de contrôle quadrilatéral de la surface. Ce que nous entendons par "bon" est que les arêtes du maillage de contrôle se croisent perpendiculairement et sont alignées avec les principales directions de la courbure de la surface. Ces deux propriétés du maillage de contrôle permettent de fournir une bonne approximation de la surface avec peu de points de contrôles. D'ailleurs, ils aident considérablement à réduire des oscillations non-désirées sur la surface spline finale.

Pour résoudre ce problème, nous proposons un nouvel algorithme automatique, appelé *paramétrisation globale périodique*. L'idée fondamentale de cet algorithme est de trouver une paramétrisation qui ait un "sens d'un point de vue géométrique", pour ce faire, elle doit être guidée par la courbure de la surface, représentée par une paire de champs de direction orthogonaux. Les iso-lignes de cette paramétrisation sont ensuite extraites pour définir un maillage de contrôle qui ait les propriétés requises.

Ce maillage de contrôle, nous permet de construire une approximation en surface T-spline de la surface triangulée initiale. Nous exposons plusieurs résultats de cette conversion d'un maillage triangulée en surface spline. Les résultats montrent que, grâce aux maillages de contrôle anisotropes, les surfaces spline finales ont beaucoup moins d'oscillations que celles construites par les méthodes précédentes qui ne tiennent pas compte de l'anisotropie de la surface.

Mots-clés: infographie, traitement géométrie, surface paramétrisation, surface spline, visualisation, surface topologie

Abstract

Aiming at converting a triangular mesh into a CAGD/CAM spline surface, this thesis focuses on one of the most crucial problems of the conversion process, i.e. extracting a "good" quadrilateral control mesh of the surface. What we mean by good is that the edges of the control mesh should be orthogonal and aligned with the principal directions of curvature of the surface. These two properties make the control mesh optimum in an approxmiation point of view, and greatly help to reduce unwanted oscillations on the final spline surface built from it.

To solve this problem, we propose a new automatic algorithm, called *periodic global parameterization*. The basic idea is to find a "geometry-meaningful" parameterization guided by a pair of orthogonal anisotropic direction fields. Then, the iso-value lines of this parameterization will be extracted to define an initial control mesh, that satisfies the two criteria of a good control mesh.

With the initial control mesh, we explain how to construct a T-spline approximation of the initial triangulated surface. We show several examples of the triangular mesh to T-spline conversion. The results show that thanks to the anisotropic control meshes, the final spline surfaces generated have much less oscillations as compared to results of previous methods, that do not take into account of the anisotropy.

Keywords: computer graphics, geometry processing, surface parameterization, spline surface, visualization, surface topology

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Madame Dominique BECHMANN, Professeur, Equipe IGG, LSIIT, Illkirch

Monsieur Leif KOBBELT, Professeur, RWTH, Informatik VIII, Aachen, Germany

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur LI Wan Chiu

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Conversion automatique de maillages en surface splines"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : « **Informatique** »

Fait à Vandoeuvre, le 02 novembre 2006
Le Président de l'I.N.P.L.,
L. SCHUFFENECKER

Par délégation
Le Secrétaire Général,

J.Y. RIVIERE

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55