L'interpolation de formes

Tran Kai Frank Da

Université de Nice, Sophia-Antipolis, UFR sciences
École doctorale STIC

L'interpolation de formes

THÈSE

présentée et soutenue publiquement le 21 janvier 2002

pour l'obtention du titre de

DOCTEUR EN SCIENCES

(spécialité : sciences de l'ingénieur)

par

Tran Kai Frank DA

Directeur de thèse : Jean-Daniel Boissonnat

Composition du jury

Président :	Michel Barlaud
Rapporteurs :	Dominique Bechmann
	Jean Sequeira
Examinateurs :	Dominique Attali
	Jean-Daniel Boissonnat
	Bernhard Geiger
	André Lieutier
	Sylvain Petitjean

Projet PRISME

À mon arrière-grand-mère Henriette,
à qui j'aurais tant aimé pouvoir
montrer ce travail achevé.

Remerciements

Une fois achevées, les choses paraissent si faciles... Cependant, que de doutes, de découragements parfois, avant d'en arriver là et bizarrement ma première pensée va vers ceux qui, comme moi, se sont engagés sur ce chemin. Étrange impression vraiment d'avoir terminé sa thèse, où la satisfaction de conclure se mêle à une certaine incrédulité tant je me suis posé de questions durant ce travail.

Avant tout, je tiens à remercier tout particulièrement Jean-Daniel Boissonnat pour m'avoir accordé et surtout conservé sa confiance tout au long de ces trois années. Je tiens à souligner la qualité de son encadrement et sa disponibilité constante malgré ses responsabilités. Je remercie également les membres de mon jury de s'être intéressés à mon travail : Dominique Bechmann [professeur à ULP Strasbourg] et Jean Sequeira [professeur à l'UFR sciences de Luminy] pour avoir accepté d'être les rapporteurs de mon mémoire, Michel Barlaud [professeur à l'ESSI Sophia-Antipolis], Dominique Attali [chercheur CNRS à l'INPG Grenoble], Sylvain Petitjean [chercheur CNRS au LORIA Nancy], Bernhard Geiger [Siemens Princeton] et André Lieutier [Dassault Systèmes Aix-en-Provence] pour leur participation à ma soutenance de thèse.

Un grand merci aussi à Agnès Clément-Bessière pour son aide précieuse dans les moments importants, sans qui je ne me serais certainement pas sorti des formalités inhérentes à la préparation d'une thèse jusqu'à l'organisation concrète de ma soutenance. J'ai beaucoup apprécié sa disponibilité et son efficacité à résoudre et souvent même à prévenir les problèmes, les soucis, qui se sont présentés à moi.

Parmi les autres membres du projet PRISME, je remercie aussi tout particulièrement David Cohen-Steiner, sans qui la méthode présentée au chapitre 3 n'aurait jamais vu le jour. J'ai énormément apprécié cette collaboration fructueuse qui fut un des moments les plus enrichissants de mon travail de thèse. Par ailleurs, je regretterai nos divagations philo-socio-politico-géo-stratégiques quasi quotidiennes.

Je veux aussi remercier spécialement Sylvain Pion, qui m'a beaucoup aidé au début et avec qui je pense avoir eu une collaboration constructive autour de CGAL et de la reconstruction par la suite. J'ajoute que j'ai aussi vraiment apprécié la qualité de son travail sans lequel l'essentiel des résultats qui vont suivre serait de qualité moindre.

Je remercie évidemment aussi les membres du projet PRISME pour leurs remarques, leurs critiques et leurs avis toujours constructifs et avisés : Mariette Yvinec et Frédéric Cazals, avec qui j'ai aussi travaillé et qui m'ont chacun beaucoup apporté en méthode et en rigueur, Olivier Devillers et Monique Teillaud, Raphaëlle Chaine et Pierre Alliez, Andreas Fabri avec qui nous avons souvent parlé *business*, Jean-Marie Morvan, Menelaos Karavelas, Radu Ursu et Dimitrii Zinoviev.

Je pense aussi à mes collègues thésards (ou maintenant docteurs) de PRISME : Stéphane Nullans, Pierre-Marie Gandoin, Sophie Balaven, Julia Flötotto, Philippe Guigue et Dominique Amar.

Une pensée pour les anciens du projet : Hervé Brönnimann, François Rebufat avec qui j'aimais bien discuter autour d'un café-clope et Alexandra Fronville, et certains p'tits stagiaires avec qui j'ai bavardé aussi, Tharavy, Mihaela, Olivier, Rachid, Carine...

Je souhaite aussi rendre hommage (et le mot est faible !) à tous ceux qui m'ont supporté (plus ou moins longtemps) sans qui ce long séjour sur la Côte d'Azur aurait été vraiment difficile pour le parisien que je suis. Sans aucun ordre de mérite (s'il y en a un à me supporter ?) et en étant déjà persuadé d'en oublier :
Clément, Monica et la petite Hélène, Nath et la old, Guillaume G., Christophe et son acolyte M Paul Casa, Laetitia et évidemment les amis du 33 : Brahim et Robert ; Radu et notre «idéal» commun, Hugues, Paulo, Aurélie, Stéphane, mais aussi le Rellier, Laurent et l'amicale du 36 : Saboab, Nordin, Michel ; José, Samami, Oscar, Alex, Marie-Laure, Sandrine et Alexandra, Mathias, Caroline L. et son homonyme (laquelle est vexée ?), André, Fred et ses soirées à Cagnes, Nicolas D., Alain le canadien, Robert le danois, Semi, David, Julien, Hervé, Blaise, Manu, tiens le Laborde aussi, Fabienne, Delphine...
À mes amis de toujours ou presque (qui, pour certains, ont aussi profité du soleil avec moi), que j'espère ne jamais perdre de vue : Philippe, Guillaume C., re-Clem, Marco, Catherine, Pierre, Thierry, Émile, Jérome, Marcin, Viet-Linh, Kim, Azad, Shervin, Christophe, David et Fleur, Thibault, Remi, Xavier ... Une pensée pour Sophie qui m'a suivi un moment dans cette aventure.

Enfin pour finir, je voudrais remercier les membres de ma famille sans qui tout cela n'aurait que peu d'importance.
Tout d'abord, mon père *et* ma mère, qui m'ont toujours soutenu et aidé en toutes circonstances tout en me laissant libre de mes choix bons ou mauvais, attentitifs et jamais contraignants, simplement par amour ; mes frères aussi, avec qui j'espère toujours garder cette complicité qui nous unit.
Merci aussi à ma grand-mère Denise qui m'a fait le plaisir de venir assister à ma soutenance et à travers elle, à toute ma famille du Pas de Calais, qui représente une partie de mes racines : Émile et Réjane, Alice, François, Mathilde ; Alain et Marcelle ; la famille Thumerel (merci à Thérèse et Véronique d'être passées me voir) ; la famille Caron ; la famille Martinage (merci à Paul et Éliane de m'avoir rendu visite) ; la famille Boyaval ; la famille Kozal ; Madeleine et Denis qui a certainement influencé mon parcours...
Je ne peux m'empêcher de songer à ces personnes qui ne liront malheureusement jamais ces remerciements : mon arrière-grand-mère Henriette qui a tant compté pour moi, ma grand-tante Paule, mon grand-père André, qui nous ont quittés l'année dernière.
J'adresse, enfin, une pensée toute particulière à ma famille chinoise, à ma grand-mère Amah, aujourd'hui disparue et à laquelle je pense souvent, à mon oncle et ma tante de Paris, mes cousins et cousines, mais aussi à mes oncles et tantes de Chine ainsi que leurs enfants que je connais moins...
Je pense aussi à ceux que je considère presque comme des membres de ma famille : Marie, Julienne, Lucien et Adrienne...
Ainsi, bien sûr, qu'aux amis de notre famille : Aline et Luc, Martine et Jean-Claude, Clotilde et Hervé et tous leurs enfants...

Sommaire

Table des figures

Introduction

§1 Description du problème

Le travail que nous allons présenter a pour objet l'étude d'une application pratique au problème très général de l'interpolation. Cette question, bien que formulée de longue date par les mathématiques, réapparaît, aujourd'hui, sous un jour nouveau grâce aux possibilités actuelles offertes par l'informatique. Le problème consistant à définir un objet interpolant un ensemble de données trouve, dorénavant, une interprétation plus concrète que jamais pour la reconstruction de modèles géométriques. En effet, pour de nombreuses applications, il est aujourd'hui nécessaire de disposer d'une représentation informatique des objets réels. Les progrès constants des procédés de numérisation et l'importance grandissante de l'outil informatique dans la conception industrielle ou encore le calcul scientifique exigent maintenant de disposer aussi de méthodes fiables et efficaces permettant de reconstruire un modèle géométrique à partir de données partielles numérisées sur un objet. Le développement de solutions concrètes à ce problème mêle différents aspects, des plus théoriques aux plus pratiques. Il s'agit en effet, non seulement de cerner les conditions mathématiques autorisant l'existence d'une reconstruction correcte d'un objet, mais aussi de savoir proposer des alternatives concrètes dans le cas où ces exigences ne sont pas vérifiées. Aussi, par ces enjeux antagonistes, le problème auquel nous allons nous confronter révèle une richesse et une complexité que bien des utilisateurs potentiels, pour qui la reconstruction n'est qu'un préalable à d'autres applications, ne peuvent soupçonner.

§2 Domaines d'application

Le champ des applications possibles au problème de la reconstruction est vaste. Tentons d'en faire un tour rapide et non exhaustif, simplement pour illustrer la pression actuelle pour la recherche et le développement de solutions pratiques performantes. On peut distinguer deux grands domaines d'applications portant le même intérêt grandissant vis à vis de la modélisation géométrique d'objets réels à partir de données numérisées. D'une part, on peut identifier certaines applications dont les besoins sont de l'ordre de la visualisation d'objets réels, tel le cinéma et plus particulièrement les effets spéciaux, mais aussi l'industrie, en pleine expansion, des jeux vidéos ou encore les applications en développement de réalité virtuelle. On citera, entre autres, le projet MICHELANGELO [LPC+00] se proposant de numériser plusieurs statues de Michel ANGE. D'autre part, on peut regrouper des applications plus scientifiques dont les besoins sont plus précis et pour lesquelles il est nécessaire d'apporter des garanties sur les modèles reconstruits. On peut songer aux différentes applications médicales, à des fins d'aide au diagnostic, de simulation ou de préparation d'interventions chirurgicales. À titre d'illustration, on peut penser au projet *visible human project* [oM86] grâce auquel on dispose de la numérisation complète d'un homme

et d'une femme. Évoquons aussi les multiples besoins de l'industrie ayant trait à la conception d'objets, des applications CAO jusqu'au *reverse engineering*. Plus généralement, au peut aussi songer aux nombreux domaines concernés par la simulation numérique, utilisant le calcul scientifique afin de modéliser les interactions des objets et de leur milieu : l'industrie aéronautique, automobile, ou encore l'océanographie. Par ailleurs, certaines applications, tels les systèmes d'informations géographiques (SIG), entremêlent les aspects de visualisation et certaines exigences de cohérence sur les modèles manipulés. Évidemment, la diversité de ces domaines d'application a pour conséquence une variété tout aussi grande pour la provenance des données que l'on aura à considérer. Évoquons, sans entrer dans le détail, les procédés de numérisation les plus connus : les scanners laser, la stéréo-vision, l'échographie, les différentes techniques de tomographie (p. ex. CT), l'imagerie par résonance magnétique (IRM), la microscopie électronique (EM) ou encore l'analyse d'images satellitaires.

§3 *Organisation de ce manuscrit*

Ce document est organisé en quatre chapitres, la volonté étant de présenter les différentes notions abordées des plus générales aux plus particulières.

Le premier chapitre a pour objet de fournir les éléments essentiels à la compréhension des idées évoquées par la suite. Il regroupe quelques rappels sur des structures de Géométrie algorithmique qui sont au cœur des méthodes de reconstruction qui seront développées ensuite et donne un panorama représentatif des différentes approches formulées jusqu'à présent pour répondre à notre problème. Cet état de l'art traitera, tout d'abord, des méthodes généralistes, qui ne présupposent que le «minimum» sur les données qu'elles ont à considérer, mais aussi de méthodes spécialisées, qui mettent à profit une structure supposée sur les entrées.

Le deuxième chapitre présente notre contribution à la librairie CGAL (*Computational Geometry Algorithms Library*). Après une présentation générale de CGAL à ce jour, nous détaillerons les différentes implantations réalisées dans le cadre d'un module d'aide à la reconstruction, aujourd'hui intégré au corps de la librairie : les Alpha-formes dans le plan et l'espace ainsi que leur version pondérée, et le calcul des coordonnées barycentriques associées aux voisins naturels dans le plan.

Le troisième chapitre propose une méthode de reconstruction généraliste par déploiement de surface. Ce déploiement est tempéré par un contrôle topologique qui nous permet de garantir l'orientabilité de la reconstruction proposée dans le cas d'objets réels. Il nous est ainsi possible de reconstruire une grande classe de modèles, surfaces orientables ou non, fermées ou à bords. Cette approche se révèle particulièrement efficace, tant en terme de qualité des résultats obtenus qu'en terme de rapidité de calcul.

Le quatrième chapitre décrit une méthode de reconstruction spécialisée dans le cas de données organisées en sections. Cette approche consiste en l'approximation du niveau zéro d'une fonction implicite définie à l'aide du schéma d'interpolation barycentrique des voisins naturels. Cette définition fonctionnelle nous permet de construire une transition lisse entre deux sections consécutives. En outre, tous les calculs étant réalisés dans le plan, les performances de notre méthode d'interpolation lisse entre sections sont comparables à celles des méthodes de référence ne proposant qu'une reconstruction linéaire.

En conclusion générale, nous reviendrons sur les différents points abordés pour lesquels nous proposerons quelques développements et perspectives avant de conclure ce travail.

Chapitre 1

Quelques notions utiles

Sommaire

§1.1 *Eléments de Géométrie algorithmique*

1.1.1 *Introduction*

La Géométrie algorithmique est une discipline née au du milieu des années soixante dix. Cependant, on peut, en quelque sorte, la considérer comme une digne héritière de l'étude des constructions géométriques «à la règle et au compas». Prenant naissance dans l'antiquité, mais s'appuyant principalement sur des résultats mathématiques initiés à la fin du XIX^{e}siècle, son objet est de proposer une approche constructive de la géométrie adaptée aux contingences informatiques. En effet, nombreux sont les domaines d'application informatique exigeant de savoir représenter et manipuler des objets de nature géométrique. On peut songer à la robotique, l'imagerie médicale, la vision par ordinateur, la conception assistée par ordinateur, l'informatique graphique, la métrologie ou encore à la réalité virtuelle. La disparité de ce vaste champ d'applications a paradoxalement donné naissance à une discipline à part entière que l'on ne peut négliger aujourd'hui. En effet, les résultats développés depuis l'origine de la Géométrie algorithmique, s'appuyant au départ sur des techniques de l'Algorithmique classique, ont donné lieu à l'émergence de techniques algorithmiques spécifiques. On peut, par exemple, penser à la technique de balayage de BENTLEY et OTTMAN. Par ailleurs, l'analyse de la complexité des algorithmes géométriques met souvent en évidence la structure combinatoire des objets construits. Ainsi interviennent, de manière intrinsèque, des résultats de Géométrie combinatoire tels le fameux théorème de la borne supérieure de MC MULLEN par exemple.

Dans la suite de cette partie, loin de prétendre présenter un panorama complet de la Géométrie algorithmique, nous nous contenterons uniquement d'exposer quelques notions essentielles à la compréhension de nos travaux. Ainsi, nous détaillerons particulièrement plusieurs sortes de *triangulations* possédant des propriétés particulières et qui interviendront de manière répétée dans la suite du manuscrit. Au centre de ce développement, on trouvera la triangulation de DELAUNAY, qui occupe une place de choix dans les approches de résolution du problème de la reconstruction en Géométrie algorithmique (cf. 1.2). Le rôle primordial de cette structure, plus encore que pour notre problème précis, est illustré par les différentes variantes que nous présenterons dans cette partie. L'ensemble des résultats développés dans ce manuscrit ne fait que rendre hommage, à leur mesure, aux travaux de M. DELAUNAY.

Aussi bien pour une présentation plus précise, que pour une introduction à des concepts élémentaires de Géométrie algorithmique, on pourra se référer à [BY95]. Il est, en effet, souhaitable pour le lecteur, qu'il se familiarise au préalable avec les définitions de simplexe, de complexe cellulaire, de triangulation et de dualité. Dans la suite du manuscrit, on considérera par ailleurs, comme acquises, différentes notions telles les structures de données et les algorithmes classiques en Algorithmique générale ou des notions de complexité algorithmique.

1.1.2 *La triangulation de Delaunay*

On considère $\mathcal{S}$ un échantillon de points de $\mathbb{R}^d$, pour un ensemble $\mathcal{T} \in \mathcal{S}$ constitué de k points affinement indépendants, avec $1 \leq k \leq d$, on note $\sigma_{\mathcal{T}}$ le simplexe de dimension $k-1$ ayant pour sommets les points de $\mathcal{T}$. Afin de définir ce qu'est une triangulation de DELAUNAY, il est utile d'exposer la propriété suivante :

Propriété 1.1 *Étant donné un échantillon de points $\mathcal{S}$, un simplexe $\sigma_{\mathcal{T}}$ vérifie la* propriété de la sphère vide *s'il existe une sphère circonscrite à $\mathcal{T}$ dont l'espace intérieur est vide de points de $\mathcal{S}$.*

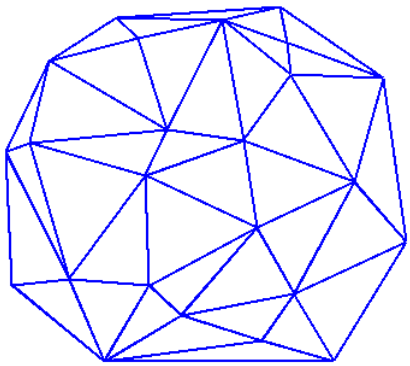


FIG. 1.1 – Illustration : une triangulation de DELAUNAY.

Alors, on dit d'une triangulation qu'elle est *de* DELAUNAY si tous ses simplexes vérifient la *propriété de la sphère vide*.

Cette propriété a plusieurs conséquences importantes. Citons, pour mémoire, les propriétés les plus connues de ce type de triangulation.

* La triangulation de DELAUNAY pour un ensemble de points en position générale (*i.e.* il n'existe pas $d+2$ points cosphériques en dimension d) est unique.
* La frontière extérieure d'une triangulation de DELAUNAY d'un ensemble de points $\mathcal{S}$ coïncide avec l'enveloppe convexe de $\mathcal{S}$.
* En dimension 2, la triangulation de DELAUNAY est la triangulation qui maximise l'angle minimum de ses triangles, parmi toutes les triangulations possibles pour un ensemble de points donné.

Par ailleurs, l'étude de son graphe dual, que nous allons aussi présenter (cf. 1.1.3), fournit d'autres propriétés remarquables. Nous reviendrons, dès la deuxième partie de ce chapitre et tout au long de ce manuscrit, à différents aspects de cette structure vis à vis du problème de la reconstruction. Ces développements étayeront, une fois de plus, les conséquences subtiles de la propriété 1.1.

1.1.3 *Le diagramme de Voronoï*

Le diagramme de VORONOI est une partition de l'espace en cellules contenant chacune un point de l'échantillon. Chaque cellule est alors une région où les points de l'espace ont pour plus proche voisin, dans l'échantillon, le point associé à cette cellule de VORONOI (figure 1.2). Pour une définition plus formelle, notons d la distance euclidienne. Pour tout point p d'un échantillon de points $\mathcal{S}$, on définit la cellule de VORONOI de p, notée $\mathfrak{V}(p)$, ainsi :

$$\mathfrak{V}(p) = \{x \in \mathbb{R}^d / \ \forall q \in \mathcal{S} \setminus \{p\},\ \mathrm{d}(p,x) \leq \mathrm{d}(q,x)\}$$

Remarquons que la cellule $\mathfrak{V}(p)$ est l'intersection des demi-espaces contenant p délimités par les plans médiateurs entre p et chacun des points de $\mathcal{S}$. Tout point de l'espace x possède un plus proche voisin dans $\mathcal{S}$ et donc, appartient à une cellule de VORONOI. De plus, un tel point x ne peut appartenir qu'à l'intérieur d'une seule cellule. En effet, si $x \in \mathfrak{V}(p_1)$ et $x \in \mathfrak{V}(p_2)$ on

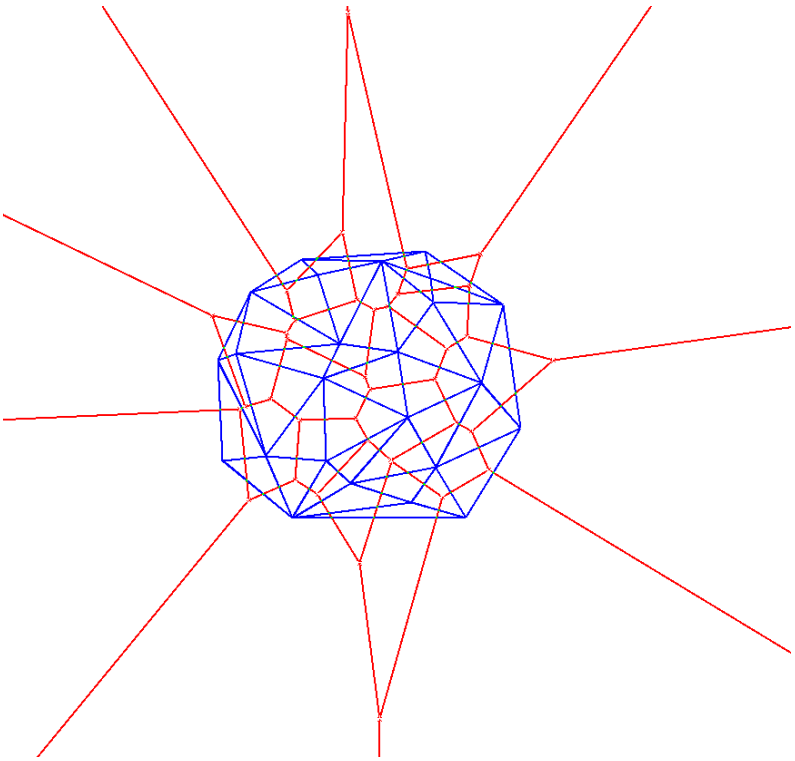


FIG. 1.2 – Illustration : une triangulation de DELAUNAY et son graphe dual, le diagramme de VORONOI.

déduit que $\mathrm{d}(p_1, x) = \mathrm{d}(p_2, x)$ et le point x est sur l'hyperplan médiateur de p_1 et p_2, frontière des cellules $\mathfrak{V}(p_1)$ et $\mathfrak{V}(p_2)$.
On appelle alors diagramme de VORONOI, $\mathfrak{V}(\mathcal{S})$, de l'échantillon de points $\mathcal{S}$, le complexe cellulaire dont les cellules de dimension maximale sont les $\mathfrak{V}(p)$ pour tous les points p de $\mathcal{S}$. On dispose ainsi effectivement, d'après les remarques du paragraphe précédant, d'une partition de l'espace.

Par ailleurs, pour établir le lien de *dualité* entre triangulation de DELAUNAY et diagramme de VORONOI, il suffit de remarquer que, pour des points $\mathcal{S}$ en position générale et $\mathcal{T} \subset \mathcal{S}$ un sous-ensemble de k points ($k \leq d+1$, d la dimension), il y a équivalence entre l'adjacence des cellules de VORONOI associées aux points de $\mathcal{T}$ et l'existence d'une sphère vide circonscrite à ces mêmes points. La relation étroite de dualité entre ces deux structures met en lumière le rôle particulier de la triangulation de DELAUNAY dans les problèmes faisant intervenir une notion de proximité et donc le diagramme de VORONOI.

Détaillons, pour mémoire, quelques autres propriétés du diagramme de VORONOI. Celles-ci, plus faciles à exposer pour VORONOI, ont évidemment leur traduction par dualité. Par souci de simplification, on considère des points en position générale.

* La cellule de VORONOI, $\mathfrak{V}(p)$, est infinie si et seulement si p est un sommet de l'enveloppe convexe de $\mathcal{S}$.
* Chaque cellule de VORONOI, $\mathfrak{V}(p)$, est un polytope convexe.
* Les sommets du diagramme de VORONOI sont les centres des sphères circonscrites aux sommets des simplexes de dimension maximale de la triangulation de DELAUNAY duale.

1.1.4 La triangulation régulière

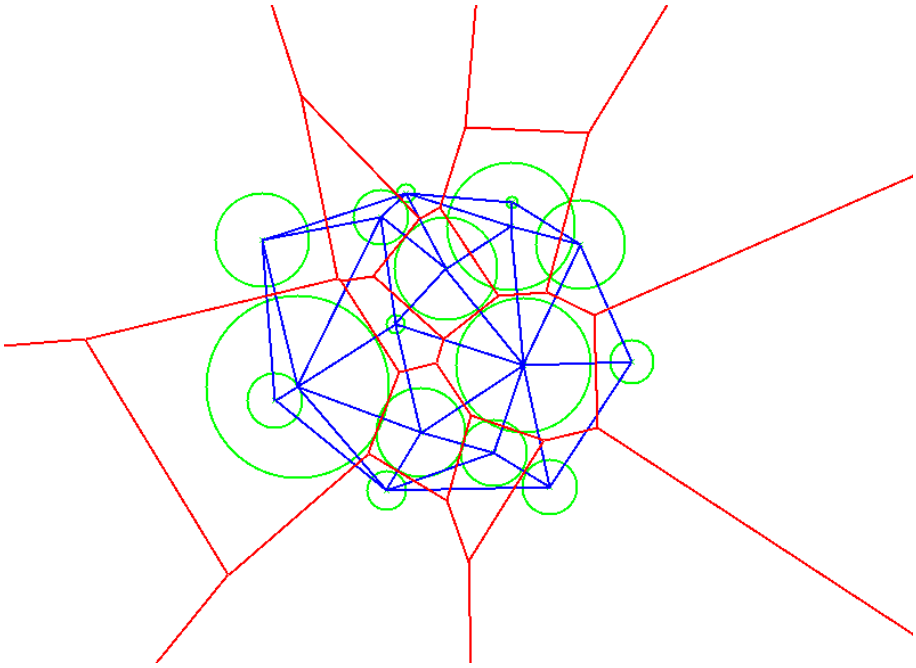


FIG. 1.3 – Illustration : une triangulation régulière et son graphe dual.

La notion de *triangulation régulière* peut être considérée comme une généralisation de la triangulation de DELAUNAY. Il s'agit, en fait, d'étendre la notion de triangulation de DELAUNAY pour une métrique non euclidienne. Plus précisément, considérons, $\mathcal{S}_w$, un ensemble de points *pondérés*, *i.e.* des points affectés d'une quantité positive nommée *poids*. On peut définir la notion d'orthogonalité entre points pondérés.

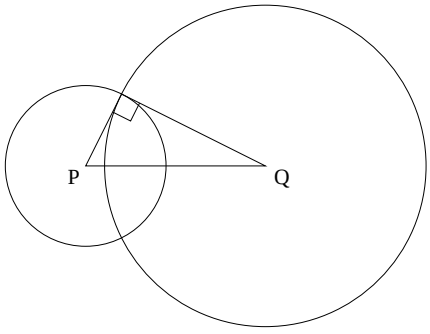


FIG. 1.4 – Illustration : orthogonalité de deux points pondérés.

Définition 1.1 *Deux points pondérés P_w et Q_w (*i.e. *des points p et q de poids respectif w_p et w_q) sont dits* orthogonaux *si et seulement si :*

$$\pi(P_w, Q_w) = \|\overrightarrow{pq}\|^2 - w_p - w_q = 0$$

Remarquons que l'on peut aussi donner une version plus géométrique de cette notion d'orthogonalité. En effet, si l'on se représente un point pondéré P_w par la sphère centrée en p de rayon $\sqrt{w_p}$. Alors, l'orthogonalité des points pondérés P_w et Q_w n'est autre que le cas où les deux sphères représentant P_w et Q_w s'intersectent *orthogonalement* (figure 1.4). Par abus de langage et pour simplifier l'exposé, on identifiera les deux représentations.

On peut, à partir de l'orthogonalité, considérer une nouvelle grandeur, la *puissance* entre deux points pondérés. Cette grandeur, bien que n'étant pas, à proprement parler, une distance, peut être utilisée pour étendre les structures présentées ci-avant au cas des points pondérés.

Définition 1.2 *On définit la* puissance *de deux points pondérés* $P_w = (p, w_p)$ *et* $Q_w = (q, w_w)$ *comme :*

$$\pi(P_w, Q_w) = \|\overrightarrow{pq}\|^2 - w_p - w_q$$

Notons qu'il existe une sphère orthogonale à un ensemble $\mathcal{T}_w \in \mathcal{S}_w$ constitué de k points pondérés, avec $1 \leq k \leq d$, d étant la dimension. On note $\sigma_{\mathcal{T}_w}$ le simplexe de dimension $k-1$ ayant pour sommets les points de $\mathcal{T}_w$.

On peut alors définir la notion de triangulation *régulière*, de manière analogue à une triangulation de Delaunay, à ceci-près que la propriété de la sphère vide se reformule ainsi :

Propriété 1.2 *Étant donné un échantillon de points pondérés* $\mathcal{S}_w$*, un simplexe* $\sigma_{\mathcal{T}_w}$ *possède la* propriété de la sphère orthogonale vide *s'il existe une sphère, représentation d'un point pondéré* C_w*, orthogonale à tous les points de* $\mathcal{T}_w$*, tels que* $\pi(C_w, P_w) > 0,\ P_w \in \mathcal{S}_w \setminus \mathcal{T}_w$.

Une triangulation *régulière* est donc une triangulation, dont tous les simplexes vérifient cette version de la *propriété de la sphère vide* pour des points pondérés.

Remarquons que cette notion coïncide avec la définition d'une triangulation de Delaunay dans le cas où tous les poids sont nuls. Par ailleurs, il y a, là encore, une notion duale, que nous ne développerons pas ici, le *diagramme de puissance*. On évoquera seulement que les cellules du diagramme de puissance ont, elles aussi, la propriété d'être convexes.

1.1.5 *La triangulation de Delaunay contrainte*

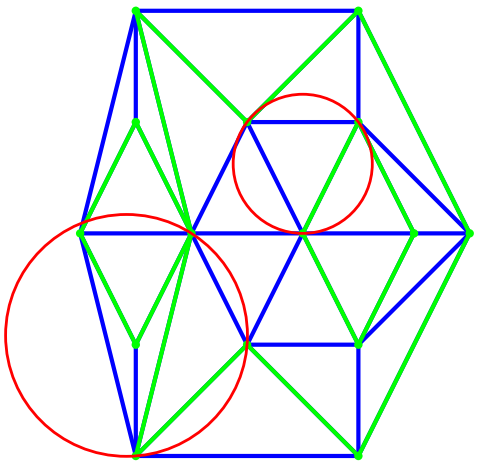


Fig. 1.5 – Illustration : une triangulation de Delaunay contrainte par les arêtes vertes.

Nous allons maintenant présenter une variante de la triangulation de DELAUNAY. On étudiera, ici, uniquement le cas de la dimension 2, d'autant plus que l'on peut montrer que l'approche proposée ne se généralise pas en dimension 3. On considère, dorénavant, disposer d'un ensemble de segments, appelés *contraintes*, noté $\mathcal{C}$. Le problème est de construire une triangulation respectant ces *contraintes* (*i.e.* les segments contraints doivent apparaître dans la triangulation), possédant idéalement les propriétés de la triangulation de DELAUNAY. Malheureusement, la triangulation de DELAUNAY étant unique, si les contraintes de l'ensemble $\mathcal{C}$ n'y apparaissent pas toutes, cette tâche est vouée à l'échec. On peut cependant se rapprocher du résultat voulu, en adaptant la propriété de la sphère vide au cas contraint. Pour un triangle, possédant une arête contrainte, on désigne comme *visible* le demi plan contenant le triangle délimité par la droite support de la contrainte.

Propriété 1.3 *Étant donné un échantillon de points $\mathcal{S}$ et un ensemble de contraintes $\mathcal{C}$, un triangle $\sigma_{\mathcal{T}}$ vérifie la* version contrainte de la propriété de la sphère vide *si la portion* visible *de l'intérieur de la sphère circonscrite à $\mathcal{T}$ est vide de points de $\mathcal{S}$ et d'extrémités des segments de $\mathcal{C}$.*

On appelle alors *triangulation de* DELAUNAY *contrainte* une triangulation pour laquelle tous les triangles vérifient cette propriété de la sphère vide adaptée au cas contraint.

1.1.6 *La triangulation de Delaunay conforme*

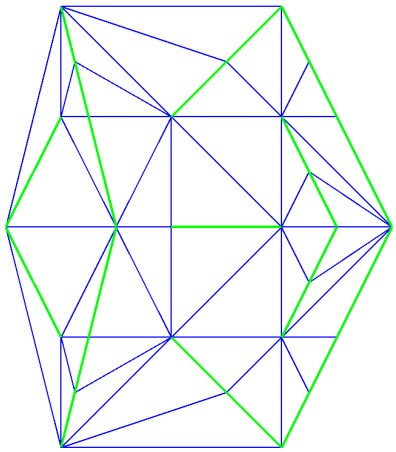


FIG. 1.6 – Illustration : une triangulation de DELAUNAY conforme vis à vis des arêtes vertes.

Cette autre variante propose une autre approche pour tenter de répondre au problème précédant, consistant à fournir une triangulation possédant les «bonnes» propriétés d'une triangulation de DELAUNAY, tout en imposant la présence d'un ensemble d'arêtes $\mathcal{C}$. Là encore, nous étudierons uniquement le cas de la dimension 2, qui sera seul utilisé dans la suite du manuscrit. Ici, plutôt que de modifier la propriété de la sphère vide afin de tolérer les contraintes, on va chercher à construire une vraie triangulation de DELAUNAY, en cassant les contraintes en autant d'arêtes de DELAUNAY que nécessaire.

Ainsi, pour être plus correct, nous allons présenter une méthode permettant de conformer (*i.e.* rendre de DELAUNAY) une triangulation dans le plan en respectant un ensemble de

contraintes donné $\mathcal{C}$ [Boi88, She96]. Tout d'abord, on insère les points de l'échantillon $\mathcal{S}$, ainsi que les extrémités des contraintes de $\mathcal{C}$ dans une triangulation de DELAUNAY. Dans le cas de contraintes disjointes et sans intersections, considérons $\mathcal{L}$, l'ensemble des contraintes de $\mathcal{C}$, dont le cercle diamétral n'est pas vide de points (*i.e.* qui n'apparaissent pas naturellement comme arête de la triangulation de DELAUNAY). Il suffit d'insérer dans la triangulation de DELAUNAY le milieu de chaque arête e de $\mathcal{L}$, et de remplacer e dans $\mathcal{L}$ par les moitiés d'arête issues de e si leur cercle diamétral n'est pas vide de points, et ce jusqu'à ce que $\mathcal{L}$ soit vide. Cette méthode termine [She96] et, bien que le nombre théorique de points rajoutés soit très grand, il se révèle tout à fait acceptable en pratique.

Par ailleurs, il existe une solution pour traiter le cas de contraintes s'intersectant (surtout quand elles forment des petits angles) [She00], qui consiste à protéger ces points d'intersections problématiques, en leur appliquant un traitement spécial. Sans entrer dans le détail de la solution évoquée pour résoudre ce problème, on pourra cependant retenir le résultat : il est possible de conformer une triangulation contrainte y compris quand les contraintes s'intersectent.

§1.2 *Le problème de la reconstruction*

1.2.1 *Du problème de la reconstruction*

Le problème de la reconstruction que nous allons aborder dans ce manuscrit dépasse de loin les enjeux que nous allons évoquer par la suite. En fait, la formulation que nous allons adopter ici n'est qu'une particularisation d'un problème fondamental étudié de longue date dans des domaines différents. Philosophiquement, on pourrait le formuler ainsi : comment représenter correctement un phénomène global simplement à partir d'observations partielles ? Cette problématique prend évidemment un sens propre suivant les différents cadres où elle intervient. Elle peut, à son tour, susciter bien d'autres questions plus métaphysiques encore. Contentons nous ici d'illustrer différentes interprétations possibles de ce problème dans un cadre scientifique afin d'en saisir les caractéristiques communes.

De nombreux domaines sont confrontés au problème consistant à retrouver les caractéristiques, la nature, d'un phénomène en ne disposant pour cela que de certaines réalisations concrètes de l'objet convoité. On peut, par exemple, songer au physicien qui tente de trouver les lois régissant un phénomène simplement à partir d'observations expérimentales. D'un point de vue plus pratique, les domaines d'applications, pour lesquels l'un des enjeux principaux est d'estimer les caractéristiques d'un objet qui n'est que partiellement maîtrisé, ne manquent pas. On citera, à titre d'exemples, la cryptologie, où on souhaiterait retrouver le message d'origine à partir du message chiffré, l'astrophysique ou à l'extrême inverse la physique des particules, où l'on doit se contenter de mesures indirectes sur les objets d'étude. Sans pousser plus loin ces considérations hasardeuses, on peut toutefois mettre en lumière, un fait qui doit être gardé à l'esprit pour aborder le problème qui va être le nôtre. Dans la plupart des cas où il est possible de modéliser correctement un phénomène prétendument «inconnu» à partir d'informations partielles, c'est essentiellement parce que l'on cherche, en fait, quelque chose de précis. C'est bien souvent un *a priori* (plus ou moins conscient) qui comble le déficit d'informations. Ainsi, nous avons tenté, au cours de ce travail, d'énoncer les différents présupposés sur lesquels reposent les approches que nous allons proposer.

1.2.2 La reconstruction de modèles géométriques

Sous une formulation plus théorique, le problème évoqué précédemment est aussi un objet d'étude important pour les mathématiques. Dans le cadre mathématique, il s'agit de reconstruire idéalement exactement, ou au moins d'approcher, un objet mathématique (par exemple une fonction) à partir d'un échantillon provenant de cet objet. Ce problème est ancien et ses développements, jusqu'à aujourd'hui, nombreux. Des formulations les plus simples[1] au fameux théorème de SHANNON et la théorie de l'information, le chemin n'est pas aisé à retracer. Nous ne nous y essayerons d'ailleurs pas dans le cadre de ce manuscrit. Il serait cependant impossible de présenter notre travail sans au moins évoquer l'imposant héritage légué par les mathématiques en ce qui concerne les problèmes d'interpolation, d'approximation ou de reconstruction.

Suivant l'évolution générale motivée par les nouvelles possibilités offertes par l'outil informatique, le problème de la reconstruction est aujourd'hui abordé sous un angle différent. Les résultats attendus se doivent dorénavant d'être constructifs plus qu'existentiels. La formulation particulière du problème qui va nous intéresser ici est la suivante :

Supposons disposer d'un ensemble de points mesurés sur un objet $\mathcal{O}$,
comment, alors, fournir un modèle géométrique qui approche la frontière de l'objet $\mathcal{O}$.

Il ne s'agit plus seulement de formuler la possibilité de reconstruire un objet mais de réaliser *efficacement* la reconstruction et d'établir un cadre pratique permettant de mieux cerner la limite entre des données partielles inintelligibles et l'objet d'origine dont elles proviennent. Sous cet aspect, de nombreux résultats mathématiques perdent de leur intérêt. Les contingences informatiques imposent, malheureusement, des contraintes spécifiques pour un calcul raisonnable. Ces contraintes ne sont cependant qu'un «petit» désagrément comparé à l'étendue des nouvelles applications possibles pour le problème de la reconstruction. En effet, de nombreux domaines, profitant eux aussi du développement de l'outil informatique, motivent aujourd'hui la recherche de méthodes pratiques pour notre problème. Sans entrer dans le détail puisque nous y reviendrons plusieurs fois au cours de notre présentation, on peut citer, à titre d'exemple, le domaine médical, le cinéma, l'industrie automobile, l'aéronautique, la géologie, chacun ayant leurs besoins spécifiques vis à vis de la reconstruction de modèles géométriques.

Avant de présenter notre contribution au problème de la reconstruction, il convient de détailler les différentes approches proposées dans la littérature pour répondre à ce problème. Nous avons choisi de séparer en deux parties cet état de l'art, afin de regrouper entre elles, d'une part, les méthodes à vocation générale et d'autre part, des méthodes dédiées à certains types d'échantillons de points.

§1.3 État de l'art des méthodes générales

1.3.1 Introduction

Précisons, tout d'abord, ce que l'on entend par *méthodes générales*. On convient dans la littérature de nommer ainsi les méthodes capables de traiter en entrée des ensembles de points qui auraient été permutés aléatoirement. Il ne s'agit pas de compliquer le problème pour le plaisir, mais plutôt de présenter le point de vue adopté par cette classe de méthodes. Il s'agit, en fait, de reconstruire des modèles sans rien présumer sur le procédé de mesure utilisé. Ainsi, on peut

[1] Quel est le chemin le plus court passant par deux points ?

considérer que les méthodes dites générales se proposent de dissocier la numérisation des objets de l'étape de reconstruction. Cela peut se justifier au regard de la grande diversité des procédés permettant d'acquérir des échantillons de points. Il serait, en effet, absurde de développer dans chaque cas des méthodes spécifiques si l'on peut reconstruire correctement les modèles grâce à une seule méthode. Pour répondre à cette formalisation générique du problème de la reconstruction, on peut distinguer deux grandes familles de méthodes :

* d'une part, les méthodes de nature discrète, fondées sur l'utilisation de certains *critères* permettant la sélection de certaines facettes dans la triangulation de DELAUNAY,
* d'autre part, les méthodes utilisant une certaine *fonction* qu'il s'agit suivant les cas d'annuler ou de minimiser afin de définir une surface interpolant les données à partir de laquelle on peut construire une représentation approchée.

1.3.2 *Les méthodes de sélection dans la triangulation de Delaunay*

Les premières approches envisagées pour répondre au problème de la reconstruction d'un ensemble de points désorganisés exploitent toutes deux certaines propriétés de la triangulation de DELAUNAY pour des points situés sur une surface. Ainsi, les *Alpha shapes* (que nous nommerons, par la suite Alpha-formes en français), énoncées par H. EDELSBRUNNER *et al.* [EKS83, EM94], et la *méthode de sculpture*, l'une des approches présentée par J.-D. BOISSONNAT dans [Boi84], sont les premières à établir un lien entre cette structure majeure de la Géométrie algorithmique, que représente la triangulation de DELAUNAY, avec le problème de la reconstruction. Sans nous attarder sur les Alpha-formes, que nous détaillerons au chapitre suivant (cf. section 2.4), on peut cependant en exprimer simplement l'idée : il s'agit de trier les simplexes de la triangulation de DELAUNAY suivant le rayon de leur plus petite sphère vide (cf. propriété 1.1), on présume alors que, pour un échantillon de points uniforme, la reconstruction est fournie par les simplexes de rayon inférieur à une certaine valeur de α. Pour la technique de sculpture, il s'agit de retirer un à un des tétraèdres à la triangulation de DELAUNAY afin de creuser l'enveloppe convexe jusqu'à atteindre la forme voulue, le retrait des tétraèdres obéissant à un certain critère géométrique sur leur aspect.

Autour des Alpha-formes

Dans la lignée des Alpha-formes, différentes méthodes ont été formulées jusqu'à aujourd'hui pour tenter de corriger les défauts de la version originale. Ainsi, les *Weighted alpha shapes*, toujours présentées par par H. EDELSBRUNNER [Ede92], proposent une généralisation des Alpha-formes au cas des triangulations régulières dans le but de pouvoir traiter des échantillons de points non-uniformes. F. BERNARDINI et C. L. BAJAJ [BB97] proposent une méthode permettant d'extraire une surface triangulée à partir du résultat obtenu grâce aux Alpha-formes, puisqu'il y a souvent des problèmes de cet ordre. Citons ici, une méthode se situant entre les Alpha-formes et la sculpture présentée par F. BERNARDINI *et al.* [BBCS99] qui se propose de creuser à la manière de la *sculpture* dans les Alpha-formes. On évoquera, enfin, la notion de *γ-graphe* introduite par R.C .VELTKAMPF [Vel92], décrivant une surface fermée passant par tous les points liée à la notion de graphe de GABRIEL.

Autour du Crust

En s'écartant de la famille de méthodes apparentées aux Alpha-formes, une autre méthode a été développée par N. AMENTA *et al.* [ABE98]. Il s'agit du *Crust*, qui propose un critère de

sélection dans la triangulation de DELAUNAY d'un type différent. Pour un échantillon de points en dimension 2, il s'agit cette fois de ne conserver les arêtes de la triangulation de DELAUNAY que si elles apparaissent toujours dans la triangulation après y avoir inséré, en plus, les sommets du diagramme de VORONOI. Une autre méthode dans le plan est présentée dans le même article : il s'agit du *β-Skeleton* qui propose de sélectionner les arêtes pour lesquelles une zone d'exclusion formée par deux cercles circonscrits à l'arête est vide. Parallèlement à ce travail, mais en conservant le formalisme introduit autour de la notion d'Alpha-formes, M. MELKEMI [Mel97] présente une idée similaire en utilisant les sommets du diagramme de VORONOI pour définir une reconstruction à différents niveaux de détails. Par ailleurs, le *Crust* a ouvert la voie à une nouvelle question concernant la qualité de l'échantillonnage requis pour permettre la reconstruction ; ainsi N. AMENTA *et al.* [ABE98] ont introduit un formalisme permettant d'évaluer un échantillon de points par rapport à l'objet sur lequel il est mesuré. Nous reviendrons bien sûr précisément sur ces différentes notions dans la suite du manuscrit (cf. chapitre 3). Les auteurs énoncent, de plus, les conditions de validité du *Crust* et du *β-Skeleton* au regard de la qualité de l'échantillon 2D fourni. Suivant cette démarche, plusieurs méthodes, toujours dans le plan, ont aussi établi des conditions minimales requises pour obtenir une reconstruction correcte. On citera les *formes r-régulières*, présentées par D. ATTALI [Att98] utilisant des notions empruntées à la Morphologie mathématique, ainsi que les résultats établis par F. BERNARDINI et C. L. BAJAJ [BB97] sur la condition d'échantillonnage requise par les Alpha-formes dans le plan. On peut aussi songer aux garanties apportées par T.K. DEY *et al.* [DMR00] pour leur méthode dans le plan, dans le cas où les conditions requises par leur approche ne seraient pas vérifiées. Enfin, J. GIESEN *et al.* [AGJ00], présente le *λ-complexe*, qui mêle entre eux différents aspects relatifs aux Alpha-formes et au *β-Skeleton*, et pour lequel sont présentés certains résultats de convergence.

De plus, le *Crust*, ainsi que les conditions d'échantillonnage définies dans le plan, ont évidemment été étendus à la dimension 3. À partir d'une première version formulée par N. AMENTA et M. BERN [AB99], permettant de déterminer parmi les sommets du diagramme de VORONOI ceux qui se trouvent proches du squelette de l'objet, on trouve plusieurs présentations de la méthode jusqu'à une formulation plus correcte et plus concise des critères employés par l'algorithme et des conditions d'échantillonnage requises dans l'espace dans l'article [ACDL00]. On peut aussi évoquer les développements de cette méthode dans le cas d'échantillons de points très volumineux par T.K. DEY *et al.* [TKDH01], utilisant les mêmes critères à partir du complémentaire d'un certain cône défini en chaque point mais simplement en découpant l'échantillon pour éviter d'avoir à construire une triangulation de DELAUNAY complète.

Les méthodes surfaciques

Par ailleurs, on peut identifier une autre famille de méthodes de nature surfacique, consistant à construire progressivement une surface sur l'ensemble des points. Une des premières approches pouvant s'interpréter de cette manière est présentée par J.-D. BOISSONNAT [Boi84], où est décrite une méthode sélectionnant des triangles successivement à partir d'une arête initiale en utilisant des critères locaux à partir d'une projection dans un plan mobile. Par ailleurs, on peut évoquer, la méthode présentée par B.K. CHOI *et al.* dans [CSYL88] qui propose de construire une surface triangulée à partir d'un triangle initial en se fondant sur l'hypothèse peu réaliste qu'il existe un point sur la surface à partir duquel l'ensemble de la surface est visible. Plus récemment, on peut songer à l'approche proposée par F. BERNARDINI *et al.* [BMR+99], où on ajoute les triangles un à un pourvu qu'il existe une boule vide de rayon fixe r circonscrite aux sommets du triangle. Dans cette lignée, on peut aussi songer à une méthode très rapide présentée par M. GOPI *et al.* [GKS00], qui étend une surface triangulée sur les points en calculant une triangulation plane

des points dans le plan de projection moyen. Enfin, on évoquera une dernière méthode récente beaucoup plus robuste que la précédente, développée par S. Petitjean et E. Boyer [PB01], qui consiste à étendre une surface dans le graphe de Gabriel à partir d'un triangle initial en utilisant à chaque étape le triangle possédant la plus petite sphère circonscrite. Par ailleurs, cet article présente un nouveau critère discret permettant d'évaluer la qualité d'un échantillon grâce à la notion d'interpolant régulier qui sera présenté plus en détail au chapitre 3.

1.3.3 *Les méthodes fonctionnelles*

Dans cette deuxième partie, nous parlerons des méthodes utilisant une certaine fonction mathématique pour guider la reconstruction. Il s'agira suivant les cas d'une fonction d'interpolation implicite dont on cherche le niveau zéro ou encore d'une énergie que l'on tentera de minimiser afin d'interpoler au mieux les données. Ces différentes approches sont plus aisées à manipuler théoriquement grâce à l'héritage conséquent des mathématiques classiques pour l'étude de tels objets. La meilleure compréhension de l'objet sur lequel s'appuie la reconstruction a évidemment bien des avantages, mais il a aussi évidemment un coût qui diffère suivant les approches adoptées. On peut se représenter les difficultés rencontrées par les méthodes de nature fonctionnelle à partir du constat suivant : les données sont de nature discrète et le résultat attendu aussi. L'utilisation de résultats mathématiques du continu nécessite alors une transcription initiale du problème du discret vers le continu, puis en fin de traitement un second passage cette fois de l'objet définissant la reconstruction à sa représentation par un modèle géométrique discrétisé.

Représentation d'une surface interpolante

Pour la classe de méthodes que nous allons présenter en premier, il s'agit de définir la reconstruction de manière implicite comme le niveau zéro d'une fonction d'interpolation définie grâce aux points de données. Pour l'ensemble de ces méthodes, on peut considérer que l'étape critique consiste à représenter la surface de niveau zéro définie implicitement. Il s'agit, en effet, de disposer d'une fonction d'interpolation facile et surtout rapide à calculer afin de pouvoir calculer efficacement des points où son évaluation est nulle.

La première approche adoptant ce point de vue est présentée par H. Hoppe *et al.* [HDD+92]. La fonction d'interpolation utilisée est définie, par morceaux, à partir de la distance signée au plan tangent en chaque point de l'échantillon. Une première étape consiste ainsi à évaluer ces plans tangents à partir des k plus proches voisins de chaque point, puis à propager une orientation cohérente sur l'ensemble de l'échantillon. Une représentation approchée du niveau zéro est alors obtenue grâce à l'algorithme du *marching cube*. Dans le même ordre d'idée, évoquons aussi la méthode proposée par B. Curless et M. Levoy [CL96] qui détaille une spécialisation de l'approche précédente au cas de la reconstruction à partir d'images volumétriques. L'utilisation de la structure particulière de ce type de données permet, en effet, de notables simplifications pour le calcul des fonctions élémentaires associées à chaque point, ici des *voxels*. On notera que cette méthode se place à la limite des méthodes dites générales que nous présentons ici.

Par ailleurs, différentes approches se situent à la frontière entre les méthodes de nature discrète et les méthodes fonctionnelles. On peut, par exemple, évoquer l'approche formulée par C.L. Bajaj *et al.* [BBX97] qui proposent une fonction d'interpolation lisse algébrique par morceaux. Cette fonction utilise le calcul d'une distance signée définie à partir d'une reconstruction obtenue grâce aux Alpha-formes. Elle requiert donc malheureusement les mêmes conditions d'échantillonnage uniforme.

Enfin, un nouvelle méthode proposée par J.-D. Boissonnat et F. Cazals [BC00] se situe elle aussi à la limite entre les approches discrètes et celles dites fonctionnelles. Elle mêle intimement les deux aspects, grâce à l'utilisation des *voisins naturels*, introduits à l'origine par R. Sibson pour l'interpolation de fonction définie sur un plan [Sib81] et utilisés par la suite dans des cadres légèrement différents [Far90, Wat92]. On la décrira ici simplement en quelques mots, puisque nous y reviendrons plus en détail au chapitre 4. Son originalité consiste à utiliser le système de coordonnées barycentriques défini grâce aux voisins naturels pour interpoler les fonctions élémentaires — distance au plan tangent — définies pour chaque point de l'échantillon. Ce système de coordonnées, que nous présenterons à la section 2.5, se calcule efficacement à partir de la triangulation de Delaunay des points et possède, de plus, des propriétés de dérivabilité remarquables. De plus, la représentation de la surface de niveau zéro de cette fonction d'interpolation peut être obtenue grâce à un nouvel algorithme de polygonalisation s'adaptant à la forme de l'objet représenté.

Déformation de contours

Les outils utilisés par la classe de méthodes que nous allons maintenant évoquer s'apparentent aux techniques classiques issues du traitement d'images. Elle consiste, en quelque sorte, à considérer une surface fermée élastique, qu'il s'agit de faire coller au plus près des données. Pour cela, on initialise, en général, le processus avec une surface fermée. Ensuite, on opère une déformation continue de la surface en cherchant à minimiser une certaine quantité, souvent interprétée comme une *énergie*, construite pour présenter un minimum local pour une reconstruction satisfaisante. Lorsque, la surface d'initialisation est suffisamment proche du minimum à atteindre, les résultats obtenus grâce à ce type d'approche sont, en général, corrects et peu sensibles au bruit sur les données. La formulation variationnelle par surface de niveaux, détaillée par J.A. Sethian [Set96] ou encore par H.-K. Zhao *et al.* [ZOMK00], offre, de plus, de bonnes performances. Cependant, plusieurs remarques viennent, tout de même, assombrir cette liste d'éloges. Tout d'abord, un des problèmes réside dans l'initialisation de la méthode, il n'est pas toujours aisé de générer des contours captant correctement la topologie du modèle dans le cas d'un nombre inconnu d'objets distincts possédant chacun une topologie complexe. Par ailleurs, la surface obtenue ne passe pas nécessairement par les points de l'échantillon. Le résultat est donc une approximation des données d'origine plus qu'une interpolation. Cette caractéristique permet, par exemple, d'obtenir une reconstruction satisfaisante dans le cas d'échantillons bruités. En contrepartie, si l'on dispose de données très précises, on peut alors regretter une distance non nulle entre la surface reconstruite et les points de l'échantillon. On observe ainsi, en quelque sorte, un lissage du bruit, mais aussi des détails, présents dans le modèle d'origine.

§1.4 *État de l'art des méthodes spécialisées*

1.4.1 *Introduction*

Nous allons maintenant présenter un tour d'horizon des méthodes spécialisées au cas de données organisées en sections. Cette forme d'échantillons est commune à différents procédés d'acquisition, dont les plus connus sont sûrement ceux en rapport avec l'imagerie médicale. Précisons, tout d'abord, l'organisation des données attendue par l'ensemble des méthodes que nous allons présenter. Dans ce cadre, on considère disposer d'un ensemble de sections successives de l'objet à reconstruire. Ainsi dans chaque section, on dispose, le plus souvent, de contours

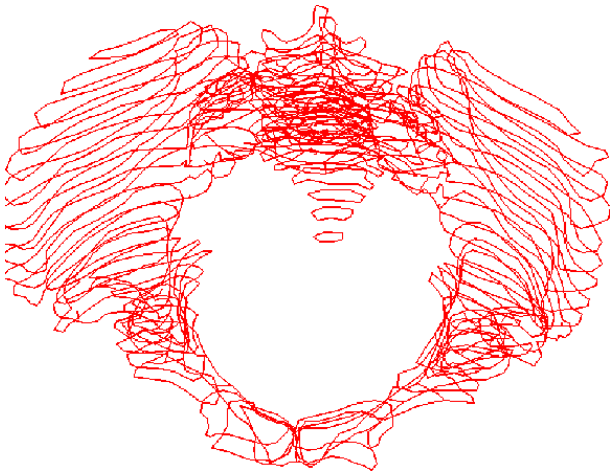


FIG. 1.7 – Illustration : des sections à interpoler.

délimitant l'intérieur de l'objet dans le plan de coupe. Par ailleurs, les plans de sections sont, dans la plupart des cas, parallèles entre eux. La raison du développement de méthodes spécifiques à ce type de jeux de données est que, bien souvent, les sections ne peuvent pas être traitées correctement par les méthodes à vocation générale. En effet, il est fréquent d'observer un écart tel entre les sections qu'il est nécessaire de forcer la connexion des contours contenus dans les sections consécutives. Ainsi, on peut identifier différents enjeux spécifiques pour les méthodes particularisées au traitement de données organisées en sections. On peut, tout d'abord, considérer que la mise en correspondance forcée entre les contours doit obéir à des critères précis qui rendent compte du comportement de la méthode. Par ailleurs, pour les approches qui permettent des mises en correspondance complexes (opérant des changement de topologie, par exemple), il est alors nécessaire de fournir une transition correcte entre les sections, capable de représenter les embranchements, l'apparition de trous etc. Pour répondre à ces différentes questions, on peut, là encore, distinguer deux grandes familles de méthodes parmi celles qui ont été proposées jusqu'ici :

* d'une part, une famille de méthode manipulant des objets de nature *surfacique*, où une fois la mise en correspondance établie, on cherche à construire entre les contours une surface triangulée ou encore à définir une surface paramétrée ;
* d'autre part, des méthodes, d'origine variées et relativement peu nombreuses jusqu'ici, fondées sur la représentation d'un certain *volume*.

1.4.2 *Les méthodes surfaciques*

Les méthodes que l'on classe dans cette catégorie possèdent pour la plupart des règles de mise en correspondance explicites. On peut considérer que c'est là leur point fort, elle permettent de contrôler exactement les contours qui seront mis en correspondance. Naturellement les critères de mise en correspondance ont fait l'objet d'études particulières, car ils conditionnent, pour bien des méthodes, entièrement le résultat de la reconstruction. Ainsi, on trouve, plusieurs études détaillées sur les conditions de mise en correspondance. On peut songer au critère formulé par M.J. ZYDA *et al.* [ZJH87], fondé sur le calcul d'un certain taux de recouvrement par projection, en considérant les boîtes englobantes des contours contenus dans deux sections consécutives. Évidemment, d'autres critères plus sophistiqués ont été énoncés depuis. On citera, entre autres,

l'étude développée par D. Meyers *et al.* [MSS92] proposant une mise en correspondance à partir de considérations sur des cylindres de base elliptique associés à chaque contour.

Évidemment la mise en correspondance des contours n'est pas tout, il s'agit, une fois cette opération réalisée, de reconstruire effectivement une surface entre les contours qui se correspondent. Pour effectuer cette opération, on peut distinguer deux types d'approches parmi les méthodes existantes : celles qui se lancent dans la reconstruction directe d'une surface triangulée s'appuyant sur les contours et celles qui utilisent une surface paramétrique pour obtenir une représentation plus lisse des transitions entre les sections.

Surfaces triangulées

Parmi les méthodes se proposant de fournir directement une reconstruction à partir des contours polygonaux décrits dans chaque section, on trouve principalement deux types d'approches. Les méthodes dont le résultat vérifie un critère d'optimalité de la surface interpolée entre les contours. On peut songer à la méthode de reconstruction proposée par H. Fuchs *et al.* [FKU77] qui consiste à réduire le problème à la recherche de chemin optimal dans un graphe représentant les différentes triangulations possibles entre les deux contours à connecter. Ce type de méthode se révèle cependant relativement coûteuse. Ainsi, d'autres méthodes moins exigeantes ont été développées, par exemple celle de D. Meyers [Mey94], se proposant d'approcher l'optimum au lieu de le calculer exactement. Le gain en terme de performance est considérable, mais des problèmes dans la reconstruction peuvent alors survenir.

Par ailleurs, ces méthodes ne sont capables de traiter la mise en correspondance que pour des contours simples. Elles ne prennent pas du tout en compte la pertinence topologique de la reconstruction. Il s'agit alors de proposer un traitement spécial dans le cas d'un changement de topologie entre les sections. Ce problème est connu dans la littérature comme étant l'étude des *embranchements*. Différentes approches pour opérer une transition acceptable entre les sections ont été proposées. Chacune a pour but d'insérer une ou plusieurs sections intermédiaires topologiquement compatibles (aux singularités près) avec les sections du dessus et du dessous. À titre d'exemple, on peut citer la méthode présentée dans l'article [MSS92] déjà évoqué. Une autre approche, formulée par J.M. Oliva *et al.* [OPC96], permet d'introduire plusieurs sections intermédiaires à partir des squelettes des contours des deux sections. Par ailleurs, on songera aussi aux travaux de G. Barequet et M. Sharir [BS96] qui proposent une méthode de reconstruction où la mise en correspondance et la construction d'une surface triangulée entre les contours appariés s'effectuent simultanément dans un plan où les deux sections considérées sont projetées le long de l'axe de mesure.

Pour conclure sur les différents aspects de ce type de méthodes, on peut citer le résultat de C. Gitlin *et al.* [GOS96], qui établit l'impossibilité de construire un maillage polyédrique à partir de deux contours polygonaux très différents. On notera que ce problème peut être résolu par l'ajout d'au plus deux points, comme le montre B. Geiger dans sa thèse [Gei93].

Surfaces paramétriques

La seconde catégorie de méthodes surfaciques utilise des surfaces paramétriques, couramment utilisée en CAO, pour définir la reconstruction. Un des intérêts de cette approche, comparée aux méthodes fondées sur l'utilisation des surfaces triangulées, réside dans la possibilité d'obtenir une reconstruction lisse. De plus, ces méthodes permettent une représentation compacte de la reconstruction que l'on peut raffiner suivant l'utilisation. Bien que proposant une description de la surface reconstruite entre les contours, cette catégorie de méthodes souffre des mêmes problèmes

que les méthodes précédemment évoquées en ce qui concerne la gestion des changements de topologie entre les sections. Tentons de donner un aperçu des idées proposées pour la reconstruction de sections grâce aux surfaces paramétriques. On peut évoquer le *Skinning*, qui consiste, en quelque sorte, à tendre une surface lisse entre deux contours à interpoler. L'idée est, tout d'abord, de définir une paramétrisation pour des contours simples à mettre en correspondance, puis de les rendre compatibles dans chaque section, ensuite on peut alors obtenir une surface paramétrique entre les sections à partir de la reconstruction obtenue dans chaque section par une courbe *B-spline*, reste alors à gérer correctement les embranchements grâce à des *carreaux* de surfaces paramétriques. À titre d'exemple, citons la méthode de *Skinning* présentée par F. Jaillet *et al.* [JSV97].

1.4.3 *Les méthodes volumiques*

En reprenant des idées proches de celles développées à la section 1.3.3, présentons maintenant les méthodes de nature volumique qui, pour certaines, utilisent aussi une fonction d'interpolation. Tout d'abord, on peut penser à l'article [Lev86] de D. Levin qui propose une méthode d'interpolation lisse de données organisées en sections, en interprétant le problème comme une généralisation du problème de l'interpolation d'un ensemble de données valuées. Cette présentation est, sans doute, une des premières formulations mêlant l'interpolation de fonction avec la notion de volume intérieur. Par ailleurs, l'auteur précise l'intérêt majeur que peut représenter cette approche du problème, à savoir être débarrassé de la gestion problématique des changements de topologie entre les sections.

Par ailleurs, les méthodes issues de l'Informatique graphique, tel le fameux algorithme du *Marching Cube* présenté par W. Lorensen *et al.* [LC87] dont l'objet, à l'origine, est de représenter des images volumétriques, peuvent aussi être considérées comme liées au cas des sections très denses. Suivant ce courant, différentes approches en Informatique graphique ont été développées pour traiter le cas des sections dans un cadre moins exigeant. On peut, par exemple, évoquer la méthode présentée par G.T. Herman *et al.* [GTHB92], qui, en quelque sorte, propose d'interpoler entre les sections une version discrète (*i.e.* sur chaque *voxel*) de la distance signée au bord de l'objet dans chaque section. Ces techniques plus tournées vers le rendu de volume qu'en rapport direct avec la reconstruction effective de modèles géométriques, ont cependant ouvert la voie à d'autres méthodes de nature volumique.

Ainsi, sous un angle différent, on peut songer à la méthode présentée par J.-D. Boissonnat [Boi88] qui introduit la triangulation de Delaunay pour guider la reconstruction offrant ainsi une certaine souplesse dans la gestion des changements de topologie. On citera aussi l'approche originale, développée par B. Geiger [Gei93], qui présente la particularité de proposer une approche discrète du problème tout en manipulant un volume intérieur. En quelques mots, l'idée est de construire deux triangulations de Delaunay planaires pour les points de deux sections consécutives (on conforme, au besoin, les arêtes des contours pour que ceux-ci apparaissent dans la triangulation), puis de former entre ces triangulations 2D des tétraèdres grâce à leur diagramme de Voronoi, que l'on trie enfin comme intérieur ou extérieur à l'objet. La surface reconstruite étant alors constituée des faces incidentes en même temps à un tétraèdre intérieur et un tétraèdre extérieur. On peut aussi citer l'approche de S. W. Cheng et T. K. Dey [CD99] qui développe des idées similaires tout en ne manipulant que des triangulations de Delaunay dans le plan.

Plus récemment, la méthode développée par B. Curless *et al.* [CL96], bien que plutôt dédiée aux images volumétriques reprend l'idée, applicable au cas de sections, de définir une fonction d'interpolation grâce à une fonction distance aux contours. Dans le même ordre d'idée, on peut songer à la méthode présentée par S. Nullans [Nul98] qui propose différentes interpolations

entre les sections pour les fonctions distances signées aux différents contours définies dans les plans de sections.

Pour conclure, on notera cependant que, jusqu'ici, seule la méthode développée par B GEIGER[2], parmi ces approches présentant une interprétation volumique naturelle, apporte, en pratique, des résultats réellement convaincants et efficaces.

[2]Méthode implantée par le logiciel NUAGES.

Chapitre 2

Contributions à CGAL

Sommaire

§2.1 *Présentation de* CGAL

2.1.1 *Motivations pour une librairie de Géométrie algorithmique*

La Géométrie algorithmique est une discipline qui fut trop longtemps considérée comme théorique. En effet, historiquement, son développement s'est amorcé autour du milieu des années soixante dix alors que, bien souvent, les moyens informatiques ne permettaient pas la mise en œuvre des solutions proposées. Ainsi, ne trouvant pas de réalisation immédiate, de nombreux travaux négligeaient les problèmes de *dégénérescence* et de *précision numérique*, en adoptant les hypothèses de la position générale et de l'utilisation d'une arithmétique exacte sur des nombres réels. Cette attitude, renforcée par une sophistication toujours plus grande des solutions proposées, pouvait alors rebuter les tentatives d'implantation ou d'utilisation, en dépit du besoin de solutions efficaces pour la résolution de problèmes géométriques.

Grâce aux progrès réalisés par les matériels informatiques, la situation est aujourd'hui toute autre et, surtout, des alternatives concrètes ont été élaborées pour échapper aux hypothèses irréalistes des débuts. Ainsi, le travail de ces années, où l'implantation pratique des algorithmes géométriques semblait inaccessible, trouve maintenant des applications réelles et performantes. De surcroît, ces nouvelles possibilités ont naturellement accru le nombre et les besoins des utilisateurs potentiels. Les problèmes de géométrie interviennent, en effet, dans des domaines extrêmement variés tels que la conception assistée par ordinateur (CAO), la robotique, les systèmes d'information géographique (SIG), la vision, les jeux vidéos, la reconstruction, la modélisation moléculaire, ou encore la conception de circuit intégrés, pour ne citer que les plus connus. Tous ces champs d'applications, aujourd'hui en plein essor, exigent pour leurs problèmes géométriques des réponses pratiques et performantes.

Dans cette nouvelle étape de son développement, l'enjeu pour la Géométrie algorithmique était de fournir une implantation rationnelle et rigoureuse des différentes structures classiques de la Géométrie algorithmique et des algorithmes qui interagissent avec elles. En effet, autant pour la pérennité de la recherche dans ce domaine que pour la diffusion des solutions dans l'industrie, il est devenu impératif d'offrir un cadre de développement pratique permettant la mise en place rapide d'algorithmes dont l'implantation *ex nihilo* pouvait rebuter. Entre une collection de petits programmes indépendants à la manière des *Graphics Gems* et le modèle d'une bibliothèque monolithique telle LEDA se voulant base unique de développement, l'enjeu principal consistait à fournir, enfin, une bibliothèque *ouverte* d'utilisation immédiate, mais dont les algorithmes pourraient aussi être utilisés ponctuellement dans un autre cadre. C'est dans ce contexte que la librairie d'algorithmes géométriques CGAL (*Computational Geometry Algorithms Library*) vit le jour.

2.1.2 *Un bref historique*

La librairie CGAL a été développée conjointement par huit équipes scientifiques[3] regroupant de nombreux spécialistes de la communauté de la Géométrie algorithmique dans le cadre d'un projet ESPRIT IV LTR financé par l'Union européenne. L'idée principale était de proposer pour la géométrie une solution comparable à la librairie standard C++ STL (*Standard Template Library*) pour l'Algorithmique générale. Après le référencement des premières implantations pratiques

[3] Ultrecht university (Pays-Bas), ETH Zürich (Suisse), Free University Berlin (Allemagne), Martin-Luther University Halle (Allemagne), INRIA Sophia-Antipolis (France), Max-Planck-Institute for Computer Science and University Saarbrücken (Allemagne), RISC Linz (Autriche) et Tel-aviv University (Israël).

et efficaces [Ame97], les premières discussions furent entreprises dès 1996, donnant lieu à la définition de plusieurs enjeux qu'une telle librairie aurait à atteindre. Ainsi, en parallèle ou presque, furent introduites la définition de l'architecture générale de CGAL [Vel97], les solutions apportées pour répondre aux problèmes de précision et de robustesse [Sch98], ou encore un modèle d'implantation des structures combinatoires typiques de la Géométrie algorithmique (polyèdre [Ket98], triangulation [BDTY00]...), pour aboutir en avril 1998 à la première version de la librairie CGAL (version 1.0).

Depuis, tout en gardant une ligne directrice constante, la librairie s'est énormément enrichie tant au niveau du contenu que de la qualité. La fiabilité de l'implantation des algorithmes de base est éprouvée par une utilisation poussée dans des cadres très variés. En outre, de grands progrès ont été réalisés en ce qui concerne les performances et, en particulier, dans le traitement de la robustesse [Pio99] amenant ainsi CGAL à un niveau des plus compétitifs. On peut considérer que le projet initial est atteint et que l'on dispose aujourd'hui d'un outil de développement performant pour la Géométrie algorithmique[4]. Suivant les évolutions techniques de la librairie, le statut de CGAL a lui aussi gagné en maturité (la version actuelle est 2.3 [CGA01]). Ainsi, au delà du monde académique, il est aujourd'hui sérieusement envisagé des applications industrielles soutenues par la possible création d'une *start-up* qui encadrerait la maintenance et les évolutions de CGAL dans le futur (cf. `www.cgal.org`).

2.1.3 *Les concepts fondamentaux de la librairie* CGAL

Présentons tout d'abord les choix généraux et fondateurs de la librairie CGAL. Outre des enjeux communs à chaque implantation tels la *correction* des algorithmes, l'*efficacité* ou encore la *facilité d'utilisation*, d'autres aspects plus spécifiques ont été retenus dans la conception de la librairie.

Modularité

L'un des premiers enjeux fut de hiérarchiser, de structurer les algorithmes géométriques. En effet, une librairie se doit de présenter une organisation claire des différentes méthodes proposées. Pour CGAL, la nature même des objets géométriques et le besoin de les manipuler explicitement ont conduit à une conception totalement *orientée objet*. L'organisation de la librairie consiste alors à classer ces différents objets selon leur nature (objets de taille fixe ou variable, simple ou composite...) et à définir les liens qui les unissent. Outre une plus grande lisibilité pour l'utilisateur, cette rationalisation permet de définir différents *modules* et des *interactions* entre ces modules. Une fois les interfaces entre les différents modules établies par des *spécifications* précises, le développement de chaque partie peut alors être confiés aux personnes les plus qualifiées et la maintenance du tout s'en trouve facilitée.

Adaptabilité

D'autre part, pour rendre compte du caractère très général de la plupart des algorithmes géométriques, on a voulu préserver l'*adaptabilité* des méthodes de la librairie. On souhaite, de plus, que les algorithmes puissent être utilisés dans toute leur généralité, par exemple, quelque soit le type de nombre utilisé. Mais plus encore, l'implantation se devait de permettre la modification des points ou des objets de base manipulés, de la métrique utilisée, si l'algorithme théorique n'en dépendait pas. Cet objectif n'est pas uniquement esthétique : éviter la duplication du code facilite la maintenance et clarifie souvent les implantations. De plus, la fiabilité de la librairie se trouve accrue, de par l'utilisation dans des configurations diverses et variées des mêmes codes sources.

[4] L'essentiel des résultats décrits dans ce manuscrit n'auraient certainement pas pu être établi sans CGAL.

Ouverture

De plus, pour la pérennité de la librairie, un soin particulier a été consacré à préserver les possibilités de collaborations avec d'autres librairies et évidemment à favoriser l'ajout d'extensions au fur et à mesure. C'est ainsi que, dès sa conception, CGAL a, dans la mesure du possible, cherché à collaborer avec des librairies concurrentes, telle LEDA, à utiliser différentes librairies de calcul exact, tel GMP ou CLN. Bref, CGAL se veut une librairie *ouverte*.

§2.2 *La réalisation de* CGAL

2.2.1 *Le choix du langage*

Pratiquement pour atteindre ces différents objectifs, le langage retenu a été le C++ , pour des raisons tant théoriques que pratiques. D'une part, les structures géométriques (points, cercles, polygones...) s'y interprètent naturellement[5] en tant qu'*objets* et les algorithmes géométriques qui agissent dessus comme des *méthodes* de ces objets. Aussi en C++ , l'*adaptabilité* trouve une réalisation pratique et efficace grâce au mécanisme de *templatisation* et au paradigme de la *programmation générique* développé à partir de la STL. Évidemment, le C++ possède l'avantage d'être performant, largement diffusé (dans le milieu industriel comme dans le monde académique) et de pouvoir être interfacé avec d'autres langages d'usage courant tels le Fortran ou le C.

2.2.2 *Généricité*

Pour réaliser l'*adaptabilité* souhaitée pour la librairie CGAL, on peut considérer que le modèle suivi a été la librairie standard C++ STL (*Standard Template Library*). Il s'agit, en effet, de proposer enfin les solutions apportées par la Géométrie algorithmique sur le modèle des solutions présentées pour l'implantation des structures de l'Algorithmique classique (liste, vecteur, queue de priorité...) dans la STL. Le paradigme de la programmation générique est obtenu par une paramétrisation des classes (mécanisme de *templatisation*, présenté dans [Str97]). Voilà somme toute, l'exemple dont CGAL s'inspire. Ainsi, le concept d'*itérateur* est repris et même développé par deux nouvelles notions les *poignées* et les *circulateurs*. Les *poignées* permettent de référencer les éléments d'un ensemble sans définir aucune succession entre ces éléments ; les *circulateurs* permettent eux de matérialiser des cycles dans les structures géométriques (le bord d'un polygone, par exemple).

Notons, par ailleurs, qu'en dehors de cet aspect technique modélisant l'*adaptabilité* souhaitée pour CGAL, une *généricité* concrète des implantations nécessite un effort particulier en ce qui concerne les *spécifications* des interfaces des différentes classes de la librairie, implémentées ou à venir. La *documentation*, accompagnant naturellement la librairie, n'a donc pas pour simple but de renseigner l'utilisateur sur les fonctionnalités offertes, mais de lui permettre d'utiliser telle ou telle partie de la librairie avec ses propres objets, ses propres algorithmes, pour peu qu'il respecte l'interface établie dans CGAL. Prendre soin de spécifier à l'avance des fonctionnalités non encore disponibles permet, en outre, de coordonner le développement en parallèle des différents modules constituant la librairie.

[5] Peut-être encore plus que pour d'autres domaines...

2.2.3 Robustesse

Un problème essentiel, inhérent à quasiment toute tentative d'implantation d'algorithmes géométriques, est lié à l'imprécision numérique des calculs de prédicats. Ce problème est d'autant plus réel, que bien souvent la description des algorithmes proposés est faite pour des données en position générale avec un modèle de calcul exact. Il est évidemment crucial pour CGAL de proposer des algorithmes fiables (ou encore *robustes*). Ainsi, les implantations traitent les configurations dégénérées et leur résultat est garanti pour un type arithmétique exact. Cependant, pour nombre d'applications, l'utilisation du calcul exact se révèle trop gourmande en temps et en mémoire. Pour obtenir une réponse plus économique en terme de performances au problème de la *robustesse*, CGAL propose une alternative. Notons, pour cela, que l'exactitude de tous les calculs n'est en fait pas nécessaire, il suffit que les *prédicats* géométriques des algorithmes soient évalués exactement. L'identification de ces prédicats permet d'adopter une approche *filtrée* [Pio99], qui consiste à calculer les prédicats tout d'abord avec un type de nombre approché qui suffit bien souvent à fournir une réponse au prédicat et à ne lancer un calcul exact que si cette réponse est incertaine. Cette approche permet de conserver une implantation claire et générique pour les algorithmes et autorise de très bonnes performances.

§2.3 Développer un module pour CGAL

2.3.1 Généralités

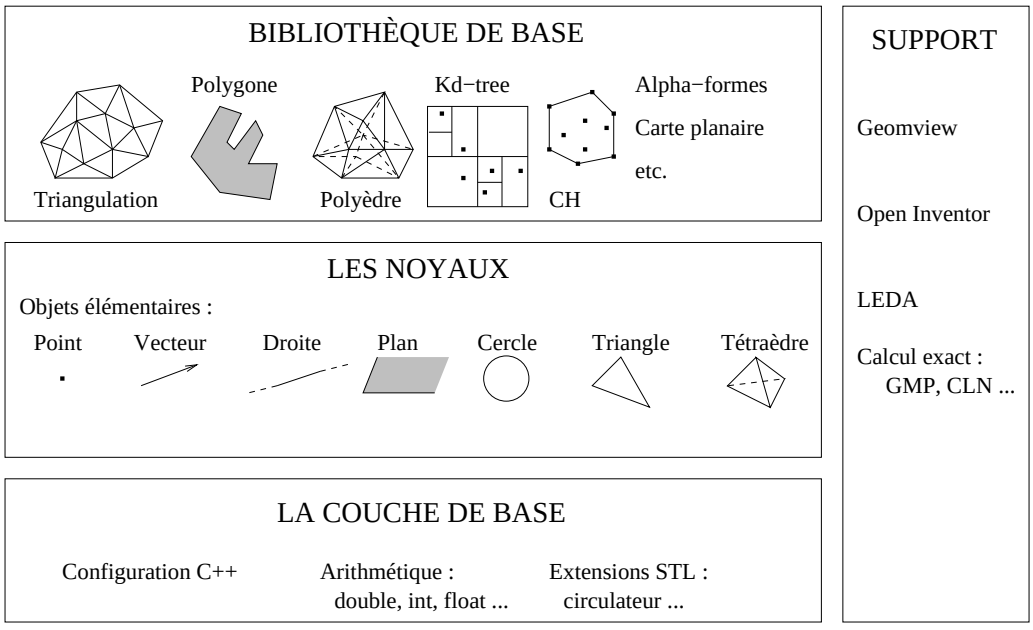


FIG. 2.1 – Illustration : les différentes couches de CGAL.

Maintenant qu'ont été introduits les différents concepts fondamentaux de la librairie CGAL, nous allons présenter différents aspects pratiques essentiels au développement de *modules* pour la librairie. Le but à atteindre est évidemment de suivre au mieux les objectifs généraux de la librairie. À cette fin, il est important de connaître, outre la structure globale de CGAL, les règles

adoptées par l'ensemble des développeurs. Pour commencer, donnons une brève description de l'organisation des différentes couches de la librairie (figure 2.1).

Organisation de la librairie

La couche de base

On regroupe ici les adaptations souvent nécessaires afin de pouvoir disposer sur chaque plate-forme d'un compilateur C++ *standard* respectant les normes en vigueur le plus possible [Str97]. Ceci concerne essentiellement la configuration des modes d'arrondis pour le calcul flottant, ou encore l'adaptation de la librairie standard *STL*. Cette partie devant idéalement disparaître, elle est implantée, en pratique, comme une partie de la bibliothèque de support, qui regroupe des fonctionnalités annexes, bien que nécessaires à l'utilisation de la librairie.

Différents noyaux

Un *noyau* contient différentes classes représentant les objets géométriques *élémentaires*, *i.e.* dont le nombre de paramètres est déterminé, ainsi que les opérations de base que l'on peut leur appliquer (prédicats, constructions). On peut citer, entre autres objets de ce type, points, vecteurs, directions d'une droite, mais aussi triangles ou encore cercles... Par opposition, un polygone, possédant un nombre arbitraire de sommets, ne fait pas partie de cette catégorie d'objets dits élémentaires. En fait, pour des raisons d'efficacité, CGAL dispose de plusieurs noyaux spécialisés dans le cas du plan, de l'espace ou encore de la dimension d. Par ailleurs, les noyaux existent chacun pour des coordonnées cartésiennes ou homogènes, implantant le *reference counting*[6] ou non.

La bibliothèque de base

Cette partie de CGAL regroupe des classes représentant des *objets complexes*, par opposition aux objets élémentaires du noyaux, ainsi que les algorithmes considérés comme classiques en Géométrie algorithmique agissant sur eux. On peut songer pour ces objets, par exemple, aux polygones comme précédemment évoqué, mais aussi aux diverses triangulations, de DELAUNAY, régulières, contraintes. Quant aux méthodes agissant sur ces objets, on citera le calcul de l'enveloppe convexe, ou encore les Alpha-formes.

La bibliothèque de support

En tant que librairie ouverte, CGAL se doit de pouvoir interagir avec d'autres librairies. À cette fin sont regroupées ici de multiples interfaces avec des librairies de calcul exact telles LEDA, GMPZ, ou CLN, mais aussi avec des outils de visualisation tels OPEN INVENTOR, GEOMVIEW, POSTSCRIPT. Cette partie de la librairie n'a pas de fonctionnalité géométrique, elle contient juste des utilitaires pour l'ensemble des autres classes.

Les modules d'extension

Cette partie de la librairie n'est pas destinée à être fournie d'office avec CGAL. On nomme ainsi les différentes contributions ayant un caractère moins général, et qui n'ont pas (encore) leur place dans la bibliothèque de base, ou encore des applications spécifiques utilisant la librairie CGAL.

[6] *reference counting* : plusieurs occurrences d'un même objet ne sont stockés en mémoire qu'une seule fois.

Protocole de développement

Par ailleurs, afin de développer un module destiné à intégrer la librairie CGAL, il est primordial d'accorder une attention particulière aux phases de spécification et de test de validation, ainsi qu'à la compatibilité avec les différentes plateformes supportées.

Pour l'étape de spécification, l'enjeu principal est de fournir une interface fonctionnelle suivant les conventions du standard et des conventions plus spécifiques à CGAL. En effet, par essence, les objets manipulés en Géométrie algorithmique sont définis pour une certaine géométrie représentée, pour les objets de CGAL, par le concept de *Traits* (qui coïncide idéalement avec le concept de noyau). Ils possèdent, de plus, une certaine structure combinatoire, représentée dans la librairie par le concept de *Data_structure* [Ket99]. Grâce à l'utilisation de l'implantation du paradigme de programmation générique en C++ par le mécanisme de *template*, chaque classe de CGAL peut être paramétrée par une *Traits* et une *Data_structure*. Une fois, les spécifications achevées, elles doivent être approuvées par le comité éditorial (cf. `www.cgal.org`). Ce travail de spécification, une fois achevé, trouve usage en tant que manuel de référence, que l'on se doit d'accompagner par un manuel utilisateur qui contiendra, outre une description qualitative des fonctionnalités apportées par le module, différents exemples d'utilisation.

Outre l'implantation des différentes classes spécifiées, CGAL impose l'écriture d'une série de tests, s'appuyant sur le principe du contrat, grâce à un mécanisme de pré-conditions et de post-conditions. En bref, les algorithmes de la librairie tentent d'offrir une garantie du type : «si les données fournies en entrée vérifient telle condition, alors l'algorithme fournit une sortie ayant telle propriété». Il est donc nécessaire d'écrire pour chaque classe une *test_suite* ayant pour rôle de vérifier chaque clause de ce contrat. La soumission des modules complets s'effectue grâce à un serveur de gestion de sources partagées, CVS. Enfin, des procédures de tests automatiques sur chacune des plateformes supportées pour l'ensemble de la librairie sont mises en œuvre périodiquement pour s'assurer de la cohérence de l'ensemble.

2.3.2 Au sujet des implantations réalisées

Ma participation à CGAL a débuté en 1998 tout d'abord en tant qu'utilisateur puis comme responsable d'un module appliqué à la reconstruction. Mon travail a consisté à spécifier puis à développer et maintenir différents algorithmes classiques d'aide à la reconstruction : les *Alpha-formes* en dimension 2 et 3, et le calcul des *voisins et coefficients naturels*. Nous détaillerons dans la suite de ce chapitre les différents choix opérés et les fonctionnalités offertes par cette partie de CGAL sous ma responsabilité. Les méthodes développées reposent essentiellement sur la triangulation de DELAUNAY et sa généralisation la triangulation *régulière*, dont les définitions formelles ont été données en 1.1.2. Nous allons développer ici les aspects d'implantation propre à CGAL, qui expliquent en partie les choix opérés dans la spécification des modules présentés ci-après. On pourra par exemple se référer à [BDTY00] pour la description de l'algorithme utilisé, et à [BCD+99] pour l'utilisation du module de Triangulation.

Le calcul des différentes triangulations disponibles dans CGAL a été organisé sur un modèle précis s'inspirant de celui développé pour la représentation de polyèdres [Ket99]. Tout d'abord, une classe particulière, la *Geometric_traits*, est chargée de fournir une représentation des points contenus dans chaque sommet de la triangulation ainsi que tous les prédicats nécessaires au calcul. Cette classe est le plus souvent instanciée par l'un des noyaux de CGAL, mais pour des triangulations «exotiques», nécessitant des prédicats à l'usage rare, il s'agit parfois d'un noyau enrichi spécifique à ce calcul particulier (par exemple, pour les triangulations régulières).

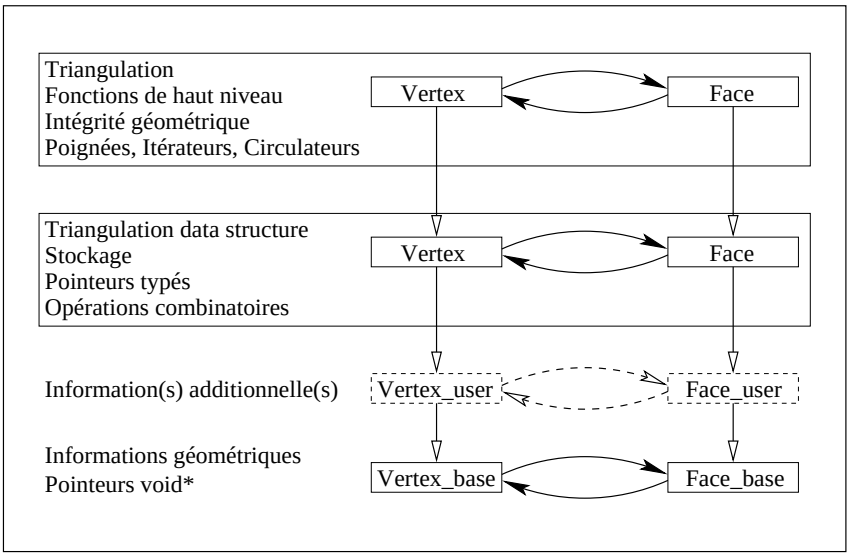


FIG. 2.2 – Illustration : implantation des triangulations dans CGAL.

Par ailleurs, on peut distinguer *trois niveaux* (figure 2.2) dans l'organisation du calcul des triangulations :

* Un niveau représentant les simplexes manipulés. Le choix a été fait de ne représenter explicitement que les sommets et les simplexes de dimension maximale, et d'en déduire les simplexes intermédiaires. Ainsi, en dimension 2, une arête sera représentée comme un couple constitué d'un triangle la contenant et de l'indice du sommet opposé à cette arête dans ce triangle.
* Un niveau combinatoire, *Triangulation_data_structure*, chargé de maintenir le graphe représentant la triangulation. Il s'agit, à ce niveau, en dehors de toutes considérations géométriques, de gérer les mises à jour des différents liens entre les cellules lors de l'ajout (ou du retrait) d'un point dans une triangulation. De plus, cette représentation se doit de fournir des réponses aux requêtes de nature combinatoire, comme l'ensemble des cellules incidentes à un sommet par exemple.
* Enfin, le niveau géométrique chargé de représenter la triangulation concrètement pour l'utilisateur. Il contient tous les éléments permettant de renseigner le niveau combinatoire, comme la localisation d'un point dans la triangulation, l'ensemble des cellules *en conflit* (pour la propriété de la sphère vide par exemple) qu'il conviendra de modifier du fait d'une insertion.

Ces choix opérés pour l'implantation des triangulations dans CGAL ont bien des avantages. Entre autres, par rapport à d'autres représentations possibles, tels les *dcells* par exemple, il propose un stockage beaucoup plus économe en mémoire : dans le plan, environ 13 pointeurs par sommets pour CGAL, contre 27 pour une structure de données de type *dcell* [BDTY00]. Bien sûr, il y a des contreparties, la représentation des simplexes de dimension intermédiaire étant implicite, l'accès à une arête diffère techniquement en dimension 2 et en dimension 3. Alors que comme précédemment évoqué, dans le plan une arête est représentée par un couple constitué d'un triangle incident et de l'indice associé au sommet opposé à l'arête dans ce triangle, dans l'espace une arête est représentée par un triplet composé d'un tétraèdre incident et des deux indices associés aux extrémités de l'arête dans le tétraèdre. En fait, dans CGAL, c'est la représentation

des $d-1$-faces qui reste identique : ainsi arête en dimension 2 et facette en dimension 3 ont la même représentation. Cette différence de représentation des arêtes en dimension 2 et 3 nous a imposé, par exemple, d'implanter aussi deux versions des classes chargées du calcul des Alpha-formes dans le plan et dans l'espace, malgré le caractère récursif par rapport à la dimension de l'algorithme théorique.

§2.4 *Alpha-formes*

2.4.1 *Présentation*

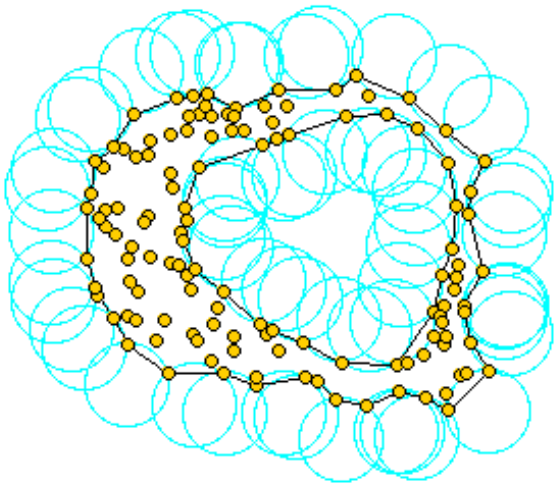


FIG. 2.3 – Illustration : Alpha-formes.

Avant de donner une définition formelle de la notion d'*Alpha-formes*, présentons en, tout d'abord, une description intuitive. Supposons que nous disposons d'un ensemble de points mesurés sur un objet. Les Alpha-formes proposent une manière simple de se représenter la forme de la frontière de cet objet. En effet, dans le plan, on peut illustrer la réponse apportée par cette approche de la manière suivante, en observant la figure 2.3. Si l'on considère qu'en chaque point un petit clou est planté et que l'on fait rouler une bobine de fil de rayon α autour de ces clous, qui laisse un morceau de fil tendu entre deux clous qu'elle a touché simultanément, on obtient alors la frontière de l'Alpha-forme associée à α. Ainsi, avec un α très grand, la bobine ne peut toucher que les points de l'enveloppe convexe. *A contrario*, pour un α très petit, notre bobine peut tourner autour de chaque point sans rencontrer aucun autre point et l'Alpha-forme est alors réduite à l'ensemble des points sans aucun lien entre eux. Ainsi, les différentes Alpha-formes constituent une famille de formes allant des points eux-mêmes à leur enveloppe convexe. On peut évidemment reprendre cette description intuitive en dimension 3, en imaginant des points fixes dans l'espace sur lesquels on fait rouler un boule de rayon α.

2.4.2 *Définitions*

La définition formelle des Alpha-formes a été présentée par H. Edelsbrunner et E. P. Mücke dans leur article [EM94]. La formalisation de l'idée intuitive consistant à «creuser» l'enveloppe convexe d'un échantillon de points à l'aide d'une boule fait intervenir la triangulation de Delaunay de ces points (cf. 1.1.2). En effet, cette boule est naturellement vide et les simplexes formant la frontière des Alpha-formes sont des simplexes de Delaunay. Détaillons plus précisément les liens entre une triangulation de Delaunay et la famille d'Alpha-formes que l'on peut lui associer tels qu'ils sont présentés dans [EM94].

Alpha-formes

Décrivons, tout d'abord, la version la plus naturelle des Alpha-formes telle que nous venons de l'évoquer de manière intuitive. À partir de cette version simple, on pourra alors définir une version *pondérée* formellement identique mais construite à partir d'une métrique différente (cf. 1.1.4). Pour préciser les idées précédemment évoquées, on se placera en dimension 3 et on considérera, pour simplifier l'exposé, un ensemble de points, $\mathcal{S}$ en *position générale* (*i.e.* sans aucune configuration dégénérée au sens de la triangulation de Delaunay). Introduisons quelques notions utiles, préalables à l'expression de la famille d'Alpha-formes associée à $\mathcal{S}$.

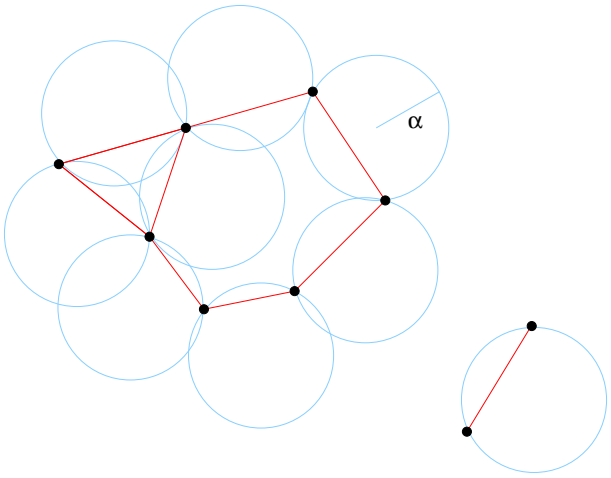


Fig. 2.4 – Illustration : α-boules et simplexes α-exposés.

Définition 2.1 Les simplexes exposés :

* *On nommera α-boule, une boule ouverte de rayon α ; une telle boule $\mathscr{B}$ est dite* vide[7]*, si $\mathscr{B} \cap \mathcal{S} = \emptyset$. Par abus de langage, on parlera aussi dans ce cas de sphère de rayon α vide de points.*
* *Rappelons que chaque sous-ensemble constitué de $k+1$ points, $0 \leq k \leq 3$, définit un k-simplexe (de dimension k, les points étant en position générale).*
* *On dira qu'un k-simplexe est* α-exposé *s'il existe une α-boule vide dont la frontière passe par les sommets du simplexe. On note alors l'ensemble des simplexes de dimension k* α-exposés *$\mathcal{F}_{k,\alpha}$.*

[7] Il ne s'agit, ni plus ni moins, que de la propriété de la sphère vide.

Pour éclairer la définition précédente, observons la figure 2.4 : on peut y voir un échantillon de points dans le plan pour lequel on a représenté les arêtes α-*exposées*, ainsi que l'une des mboxα-boules qui peut leur être associée. Par commodité, on notera α-*forme* l'Alpha-forme associée au paramètre α. Ainsi, on définit la famille des Alpha-formes associée à un échantillon de points $\mathcal{S}$ grâce à la définition suivante.

Définition 2.2 *Pour un échantillon de points $\mathcal{S}$, on appelle* α-forme *le polytope composé des triangles $\mathcal{F}_{2,\alpha}$, des arêtes $\mathcal{F}_{1,\alpha}$ et évidemment des sommets*[8].

Notons que l'on parle ici d'un polytope au sens le plus général, puisqu'une α-forme, $\mathfrak{S}_\alpha$, n'est *a priori* qu'un complexe composé de simplexes de dimensions variées sans propriétés particulières. Pour être un peu plus précis, observons l'ensemble des triangles α-exposés. Pour chacun de ces simplexes, il existe deux α-boules passant par ses trois sommets. Si ces deux boules sont vides, le triangle sera considéré comme étant une *singularité* : en effet, il appartient à l'α-forme, mais pas à sa frontière. Dans l'autre cas, une seule des α-boules est vide, le triangle α-exposé appartient alors à la frontière et on considère que le centre de la boule non-vide indique le côté intérieur à l'α-forme. On peut alors, aisément classer les différentes composantes connexes de $\mathbb{R}^3 - \partial\mathfrak{S}_\alpha$ comme intérieures ou extérieures.

Si l'on se réfère à la figure 2.4 : en dimension 2, l'α-forme constituée, selon la définition, des arêtes α-exposées ainsi que des sommets est représentée en rouge ; le triangle à gauche de la figure est intérieur, le rayon de son cercle circonscrit étant plus petit que α.

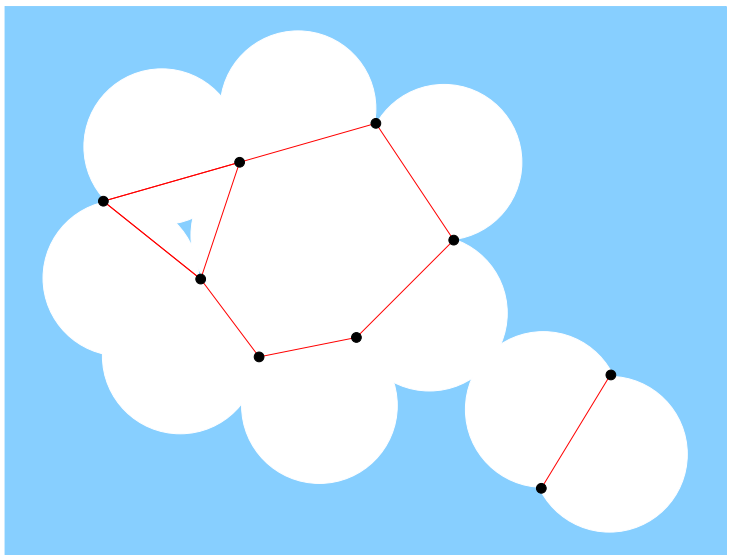


FIG. 2.5 – Illustration : α-enveloppe.

D'autres objets liés aux Alpha-formes peuvent être définis, ils permettent, en fait, de visualiser une α-forme sous un jour différent. On se limitera ici à les évoquer, sans détailler leurs propriétés qui découlent naturellement de celles de la famille des Alpha-formes. Pour un α donné, on peut considérer :

l'α-enveloppe $\mathfrak{H}_\alpha$ qui est le complémentaire de la réunion de tous les α-disques vides. Remarquons que, pour $\mathcal{S}$ un échantillon de points, $\mathfrak{H}_\infty$ est l'enveloppe convexe de $\mathcal{S}$, que l'on a

[8] Les sommets sont toujours exposés.

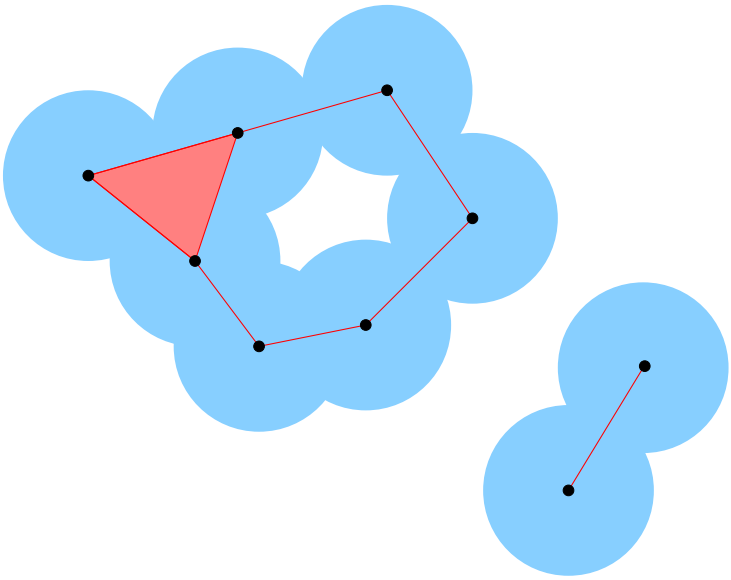


FIG. 2.6 – Illustration : α-diagramme.

$\mathcal{S} = \mathfrak{H}_\alpha$ pour α assez petit et que $\mathfrak{H}_{\alpha_1} \subseteq \mathfrak{H}_{\alpha_2}$ pour $\alpha_1 < \alpha_2$. En dimension 2, on peut se représenter l'α-enveloppe en se reportant à la figure 2.5 òu elle est représentée en bleu dans des conditions identiques à celles de la figure 2.4.

l'α-diagramme $\mathfrak{U}_\alpha$ est la réunion des α-boules centrées sur des points de l'échantillon. Sa frontière est ainsi constituée de portions de sphères, d'arcs de cercle et de sommets, qui sont respectivement les 2-, 1- et 0-faces de l'α-diagramme. La figure 2.6 représente en bleu l'α-diagramme associé à l'α-forme de la figure 2.4.

l'α-complexe $\mathfrak{C}_\alpha$ est le sous-complexe simplicial de la triangulation de DELAUNAY qui contient tous les simplexes dont la plus petite boule vide circonscrite est de rayon inférieur à α. Remarquons que la frontière de l'union de tous les simplexes formant l'α-complexe est exactement l'α-forme. De plus, l'intérieur de l'α-complexe correspond à l'intérieur de l'α-forme. Sur notre exemple dans le plan (figure 2.6), l'α-complexe est ainsi constitué des sommets, des arêtes de l'α-forme et du triangle rouge.

Alpha-formes pondérées

Nous allons maintenant présenter une généralisation de la notion d'Alpha-formes. De même que la *triangulation régulière* est une généralisation de la triangulation de DELAUNAY (cas régulier avec des poids nuls), les Alpha-formes d'un échantillon de points se généralisent en Alpha-formes *pondérées*. Pour cela, il suffit de reprendre les définitions précédentes et de les interpréter dans le cas de points munis de *poids* non nuls et grâce à la notion de *puissance* entre points pondérés. L'intérêt d'introduire la notion d'Alpha-formes pondérées est de pouvoir donner une importance différente à chaque point. Ainsi on peut espérer traiter des échantillons de points non-uniformes en attribuant de plus grands poids dans les zones les moins denses.

Précisons les liens entre Alpha-formes pondérées et triangulation régulière. On considère un ensemble de points pondérés $\mathcal{S}_w$. En se référant à la définition d'une triangulation régulière (cf. 1.1.4), la notion d'*orthogonalité* permet de définir, dans l'espace, la sphère orthogonale à

quatre points pondérés qui joue le rôle de la sphère circonscrite à quatre points dans le cas de DELAUNAY. Poussons un peu le parallèle entre les notions de triangulation de DELAUNAY et de triangulation régulière et appliquons-le aux Alpha-formes. On veut évidemment que la notion de Alpha-formes pondérées coïncide avec celle d'Alpha-formes dans le cas où les poids sont nuls. Rappelons la définition de la *puissance* de deux points pondérés, $P_w = (p, w_p)$ et $Q_w = (q, w_w)$, $\pi(P_w, Q_w) = \|\overrightarrow{pq}\|^2 - w_p - w_q$.

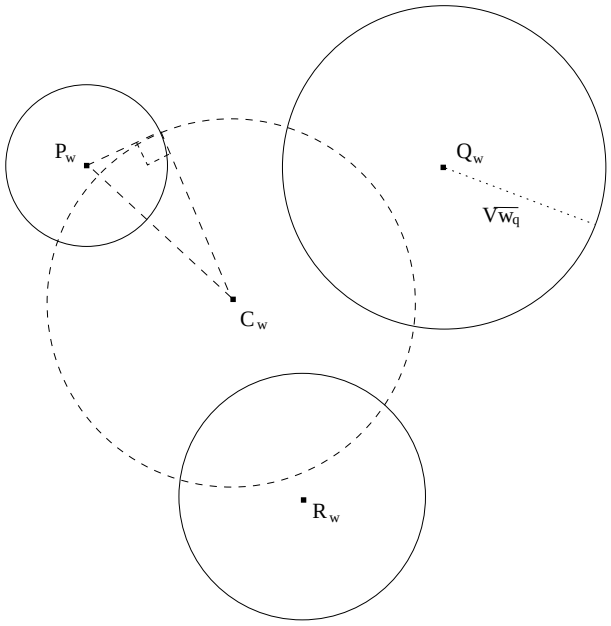


FIG. 2.7 – Illustration : sphère orthogonale à trois points pondérés P_w, Q_w et R_w dans le plan.

La *puissance* pour une triangulation régulière peut être considérée comme l'équivalent de la distance euclidienne dans le cas d'une triangulation de DELAUNAY. Pour clarifier l'exposé et aider à mieux se représenter les différentes propriétés, il est important d'avoir à l'esprit qu'un point pondéré peut être vu comme une sphère centrée en ce point et dont le carré du rayon équivaut au poids de notre point. De même, une sphère peut être interprétée comme un point pondéré. Ainsi en dimension 3, on peut alors considérer la sphère orthogonale à quatre points pondérés comme le point pondéré de puissance nulle par rapport aux quatre points considérés. Dans le plan, on aura, de même, une sphère orthogonale à trois points pondérés comme illustré par la figure 2.7.

Ainsi, avec ce vocabulaire, on a une transcription de la *propriété de la sphère vide* :
une sphère de centre c de rayon r sera dite vide, si vue comme le point pondéré $C_w = (c, w_c)$ avec $w_c = r^2$, aucun point de $\mathcal{S}_w$ n'est de puissance strictement négative par rapport à C_w (figure 2.8).

Considérons un simplexe de dimension k, dont l'ensemble des sommets est $\mathcal{T}$. On dira que ce simplexe est α-*exposé* s'il existe une sphère *vide* de rayon α orthogonale aux sommets de $\mathcal{T}$. On peut aussi écrire ceci plus formellement : il existe un point pondéré C_w de poids $w_C = \alpha^2$ tel que pour tout point de l'échantillon $P_w \in \mathcal{S}_w$, soit $P_w \in \mathcal{T}$ et $\pi(P_w, C_w) = 0$, soit $P_w \notin \mathcal{T}$ et $\pi(P_w, C_w) > 0$. On note alors l'ensemble des simplexes de dimension k α-*exposés* $\mathcal{F}^w_{k,\alpha}$.

Ainsi, on peut maintenant présenter la notion d'Alpha-formes pondérées, comme la famille des différentes α-*formes* pondérées. Pour un échantillon de points pondérés $\mathcal{S}_w$, on appelle α-*forme* pondérée le polytope composé des triangles $\mathcal{F}^w_{2,\alpha}$, des arêtes $\mathcal{F}^w_{1,\alpha}$ et des sommets $\mathcal{F}^w_{0,\alpha}$, qui

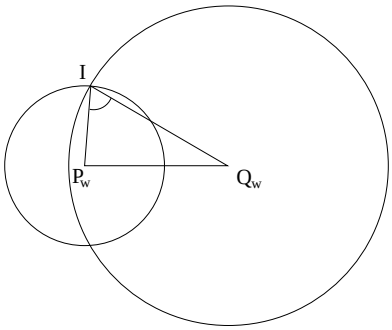


FIG. 2.8 – Illustration : P_w et Q_w deux points pondérés de puissance négative dans le plan ; l'angle au sommet I est alors inférieur à $\frac{\pi}{2}$.

sont les sommets non cachés dans la triangulation régulière.

Les notions d'α-*enveloppe*, d'α-*diagramme* et d'α-*complexe* ont, elles aussi, tout naturellement leur version pondérée. Les liens existants entre ces différents objets restent les mêmes que dans la version classique, nous ne nous étendrons cependant pas davantage sur ce sujet.

2.4.3 *Propriétés et calcul des Alpha-formes*

Propriétés d'une famille d'Alpha-formes

Nous allons maintenant détailler les propriétés essentielles à la compréhension du comportement d'une famille d'Alpha-formes. Cette connaissance nous permettra alors de définir un algorithme de calcul pour une famille d'Alpha-formes. Revenons donc aux différents aspects liant Alpha-formes et la triangulation sous-jacente, DELAUNAY dans le cas classique, *régulière* pour le cas pondéré. On se contentera dans cette présentation de détailler les différentes propriétés en renvoyant le lecteur vers l'article [EM94] pour les preuves. Il convient, en premier lieu, d'énoncer une propriété importante qui caractérise les α-formes comme incluses dans la triangulation sous-jacente.

Propriété 2.1

* *Pour chaque simplexe de la triangulation sous-jacente, il existe un plus petit α pour lequel il devient exposé, donc un simplexe de l'α-forme.*
* *Réciproquement, étant donné un ensemble de points, la frontière de toute Alpha-forme est formée de simplexes de la triangulation sous-jacente.*

Cette propriété nous permet donc de considérer la construction d'une α-forme comme un problème de sélection de simplexe dans la triangulation sous-jacente. La figure 2.9 illustre, par exemple, la sélection des arêtes constituant une Alpha-forme simple parmi les arêtes de la triangulation de DELAUNAY des points. Il s'agit maintenant, de classifier les simplexes de la triangulation sous-jacente pour connaître précisément leur rôle pour une α-forme. En cela, par dualité, le diagramme, de VORONOI dans le cas classique, ou de *puissance* dans le cas régulier, nous fournit une autre propriété intéressante pour la description des Alpha-formes.

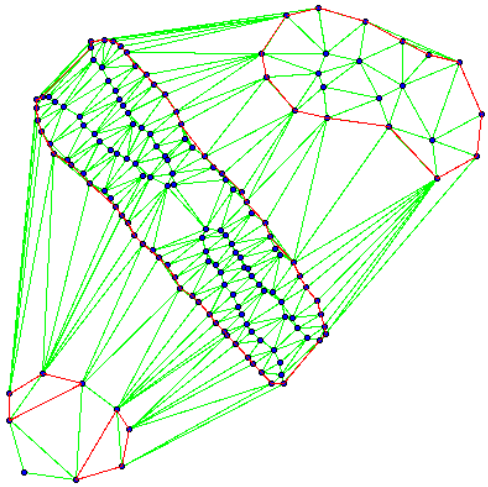


FIG. 2.9 – Illustration : sélection des arêtes composant une Alpha-forme simple dans la triangulation de DELAUNAY.

Propriété 2.2 *Soit $\mathcal{T}$ un ensemble de $1 \leq k \leq 4$ points. Notons $\mathfrak{V}(\mathcal{T})$ la face de dimension $(4-k)$ associée à $\mathcal{T}$ dans le diagramme de* VORONOI/puissance. *Alors, par convexité de $\mathfrak{V}(\mathcal{T})$, il existe un unique intervalle I tel que le $(k-1)$-simplexe $\sigma_{\mathcal{T}}$ soit α-exposé si et seulement si $\alpha \in I$.*

Une autre chose capitale est de remarquer que, pour un ensemble fini de points, les Alpha-formes forment une famille discrète (naïvement les simplexes de la triangulation sous-jacente, qui sont en nombre fini, peuvent soit appartenir, soit ne pas appartenir à l'Alpha-forme) et que deux Alpha-formes $\mathfrak{S}_{\alpha_1}$ et $\mathfrak{S}_{\alpha_2}$ diffèrent si et seulement s'il existe au moins un simplexe dans la triangulation dont la plus petite sphère vide circonscrite/orthogonale a un rayon compris entre α_1 et α_2.

Propriété 2.3 *Une Alpha-forme est donc caractérisée par un certain intervalle de valeur $[\alpha_i, \alpha_{i+1}]$. La famille des Alpha-formes est donc en correspondance bijective avec une famille d'intervalles contigus dont la réunion est $\mathbb{R}^+$. On nommera cette famille le* spectre *de l'Alpha-forme.*

Pour mieux comprendre comment construire ces intervalles caractérisant la famille d'Alpha-formes, revenons sur la notion d'α-complexe qui permet de clarifier l'interprétation. Remarquons, tout d'abord qu'un simplexe de la triangulation sous-jacente appartient à l'α-complexe si et seulement si α appartient à un certain intervalle $[\rho, +\infty[$, où ρ est le rayon de la plus petite sphère vide circonscrite/orthogonale au simplexe. Plus précisément, à l'intérieur même de cet intervalle, on peut encore distinguer différentes propriétés.

Définition 2.3 *On dira d'un k-simplexe, $1 \leq k \leq 3$, qu'il est* attaché *si sa plus petite sphère circonscrite/orthogonale n'est pas vide (on dit aussi que ce simplexe n'est pas de* GABRIEL*).*

Dans le cas *attaché*, le simplexe n'est incorporé à l'α-complexe que comme composant d'un simplexe de dimension supérieure. Au contraire, pour un simplexe *non attaché*, il est possible pour

un certain intervalle de α qu'il appartienne à l'α-complexe sans être incident à aucun simplexe de dimension supérieure de l'α-complexe, il forme alors une *singularité*. On dira alors de lui qu'il est *singulier*. Avec ces notations, nous pouvons détailler comment se découpe l'intervalle associé à un simplexe de la triangulation sous-jacente.

On peut découper cet intervalle en trois zones : un premier sous-intervalle, allant de l'infini à une certaine valeur, où le simplexe est *intérieur* (pas sur la frontière de l'α-forme), un autre où il est sur la frontière et incident à un simplexe de dimension supérieure dans l'α-complexe, on dit alors de lui qu'il est *régulier* et enfin un dernier sous intervalle de borne inférieure ρ, où il appartient à l'α-complexe, mais crée une singularité, on le considère alors *singulier*. Remarquons qu'un simplexe de l'enveloppe convexe n'est jamais intérieur (sous-intervalle correspondant vide) et qu'un simplexe *attaché* n'est jamais *singulier* (*idem*). De plus, on remarquera que l'on peut décider si un simplexe est attaché ou non, en temps constant amorti (pour chaque simplexe de dimension intermédiaire, il suffit de parcourir les simplexes incidents de dimension supérieure).

Ces considérations permettent ainsi de décrire la famille d'Alpha-formes associée à une triangulation sous-jacente en calculant pour chaque simplexe son intervalle caractéristique et son découpage le cas échéant. En effet, alors que le calcul d'une α-forme peut être opéré de la manière naïve, des opérations nécessitant la connaissance de la famille entière des Alpha-formes, comme la recherche d'un bon α pour reconstruire un objet, peuvent être effectuées avec profit sur son *spectre*.

Algorithme de calcul d'une famille d'Alpha-formes

Nous avons maintenant tous les éléments pour décrire de manière efficace un algorithme de calcul de la famille des Alpha-formes pour un ensemble de points $\mathcal{S}$ donné dans $\mathbb{R}^3$. Pour pouvoir traiter le cas classique et le cas pondéré avec le même algorithme, on parlera simplement de triangulation sous-jacente pour désigner soit la triangulation de DELAUNAY, soit la triangulation régulière.

Pour chaque simplexe $\sigma_{\mathcal{T}}$ de la triangulation sous-jacente, dont l'ensemble des sommets est noté $\mathcal{T}$, on calcule les trois grandeurs nécessaires (quand cela à un sens) pour décrire précisément les trois zones de son intervalle d'appartenance.

* $\rho_{\mathcal{T}}$ est égal au carré du rayon de la plus petite sphère circonscrite/orthogonale au simplexe $\sigma_{\mathcal{T}}$, si cette sphère est vide, et sera considéré comme non défini dans le cas contraire ;
* $\mu_{\mathcal{T}\_}$ le carré du plus petit rayon d'une sphère circonscrite/orthogonale à un simplexe de dimension supérieure incident à $\sigma_{\mathcal{T}}$;
* $\mu_{\mathcal{T}}{}^{+}$ le carré du plus grand rayon d'une sphère circonscrite/orthogonale à un simplexe de dimension supérieure incident à $\sigma_{\mathcal{T}}$ (notons que $\mu_{\mathcal{T}}{}^{+}$ peut être infini).

Remarquons que $\mu_{\mathcal{T}\_}$ et $\mu_{\mathcal{T}}{}^{+}$ ne sont pas définies pour les simplexes de dimension maximale. De plus, suivant que le simplexe est sur l'enveloppe convexe ou non, suivant qu'il est *attaché* ou non, on aura différentes configurations que l'on peut résumer dans le tableau 2.1. On connaît ainsi exactement le rôle de chaque simplexe de notre triangulation dans une Alpha-forme associée à un α particulier. De plus, on a alors aussi accès au *spectre* de la famille des Alpha-formes, d'après la propriété 2.3. Il suffit de trier les simplexes en ordre croissant par la valeur caractérisant le passage de l'extérieur à l'intérieur pour obtenir les valeurs α_i décrivant le spectre de la famille des Alpha-formes.

En fait, on peut définir suivant ce qui nous intéresse deux variantes d'Alpha-formes. La version normale (ou *singulière*), où on autorise des facettes ou des arêtes *singulières*, et une version dite *régulière*, où on exclut ces singularités. Dans le premier cas, on considérera que l'intervalle caractéristique d'un simplexe $\sigma_{\mathcal{T}}$ est la réunion des intervalles des colonnes titrées

	singulier	régulier	intérieur
tétraèdre			$[\rho_{\mathcal{T}}, +\infty[$
facette, arête			
- hors enveloppe convexe, non-attachée	$[\rho_{\mathcal{T}}, {\mu_{\mathcal{T}}}_-]$	$[{\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}^+]$	$[{\mu_{\mathcal{T}}}^+, +\infty[$
- hors enveloppe convexe, attachée		$[{\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}^+]$	$[{\mu_{\mathcal{T}}}^+, +\infty[$
- sur l'enveloppe convexe, non-attachée	$[\rho_{\mathcal{T}}, {\mu_{\mathcal{T}}}_-]$	$[{\mu_{\mathcal{T}}}_-, +\infty[$	
- sur l'enveloppe convexe, attachée		$[{\mu_{\mathcal{T}}}_-, +\infty[$	
sommet			
- hors enveloppe convexe	$[0, {\mu_{\mathcal{T}}}_-]$	$[{\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}^+]$	$[{\mu_{\mathcal{T}}}^+, +\infty[$
- sur l'enveloppe convexe	$[0, {\mu_{\mathcal{T}}}_-]$	$[{\mu_{\mathcal{T}}}_-, +\infty[$	

TAB. 2.1 – Classification des simplexes pour les Alpha-formes en dimension 3.

singulier et *régulier* dans le tableau de classification 2.1. Dans le second cas, on considère que seul l'intervalle de la colonne notée *régulier* caractérise l'entrée et la sortie dans l'Alpha-forme. Ainsi, on peut maintenant décrire l'algorithme de calcul du spectre d'une famille d'Alpha-formes, $\{\mathfrak{S}_\alpha\}_{\alpha\in[0,+\infty[}$. Pour représenter à la fois le spectre de la famille d'Alpha-formes dans sa version singulière et dans sa version régulière, on utilisera un ensemble, noté $\mathcal{L}$, dont les éléments seront des quadruplets $(\sigma_{\mathcal{T}}, \rho_{\mathcal{T}}, {\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}_+)$. En effet, par définition, on a $\rho_{\mathcal{T}} < {\mu_{\mathcal{T}}}_- < {\mu_{\mathcal{T}}}_+$. Ainsi, la valeur caractérisant l'appartenance d'un simplexe $\sigma_{\mathcal{T}}$ à une Alpha-forme est $\rho_{\mathcal{T}}$ pour les simplexes de dimension maximale et ${\mu_{\mathcal{T}}}_+$ pour les autres, on la nommera *valeur caractéristique* de $\sigma_{\mathcal{T}}$[9].

Algorithme de calcul d'une famille d'Alpha-formes

Require: $\mathcal{S}$, un ensemble de points.
Ensure: $\mathcal{L}$, un ensemble représentant la famille des Alpha-formes de $\mathcal{S}$.
  calculer la triangulation sous-jacente $\mathfrak{T}$ de l'ensemble de points, $\mathcal{S}$;
  for all simplexe $\sigma_{\mathcal{T}} \in \mathfrak{T}$ **do**
    classifier $\sigma_{\mathcal{T}}$ suivant le tableau 2.1 ;
    calculer le triplet de valeurs $(\rho_{\mathcal{T}}, {\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}_+)$ (mettre *indéfini* le cas échéant) ;
    $\mathcal{L} \Leftarrow (\sigma_{\mathcal{T}}, \rho_{\mathcal{T}}, {\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}_+)$;
  end for
  trier $\mathcal{L}$ en ordre croissant selon la *valeur caractéristique* pour chaque élément ;
  return $\mathcal{L}$;

Grâce à cet ensemble $\mathcal{L}$, on peut alors répondre à de multiples requêtes. En particulier, pour un α donné, on peut fournir l'Alpha-forme correspondante en parcourant $\mathcal{L}$ dans l'ordre jusqu'à l'élément de $\mathcal{L}$ de *valeur caractéristique* supérieure ou égale à α. Tous les simplexes des éléments précédents $(\sigma_{\mathcal{T}}, \rho_{\mathcal{T}}, {\mu_{\mathcal{T}}}_-, {\mu_{\mathcal{T}}}_+)$ sont extérieurs, réguliers ou singuliers. Il suffit de regarder où se situe α : s'il est inférieur à $\rho_{\mathcal{T}}$, le simplexe est extérieur ; sinon on regarde s'il est inférieur à ${\mu_{\mathcal{T}}}_-$, auquel cas il est singulier, dans le cas contraire il est régulier.

Le calcul de la triangulation sous-jacente est réalisé en temps $O(n \log n + n^{\lceil d/2 \rceil})$, n étant le nombre de points de l'échantillon et d la dimension (cf. [BY95]). La suite de l'algorithme consiste, en fait, en un tri des différents simplexes de la triangulation. Elle peut donc être réalisé en temps $O(m \log m)$, où m est le nombre total de simplexes de la triangulation. La requête sur une Alpha-

[9] C'est la valeur à partir de laquelle un simplexe devient intérieur.

forme souhaitée nécessite, elle, un temps logarithmique dans le nombre total d'Alpha-formes.

Prédicats et constructions nécessaires

Nous allons maintenant détailler les prédicats et constructions utilisés pour le calcul d'une famille d'Alpha-formes. On se place dans le cas *pondéré*, sachant que les prédicats dans le cas classique s'en déduisent naturellement en prenant les poids égaux à zéro. Outre les prédicats utilisés pour le calcul de la triangulation de DELAUNAY dans le cas classique, *régulière* dans le cas pondéré, on a besoin pour les Alpha-formes d'un test permettant de discriminer les simplexes *attachés* des autres. Pour cela, il suffit de tester pour chaque simplexe de dimension non maximale si sa plus petite sphère circonscrite/orthogonale est vide. D'autre part, nous avons besoin de savoir calculer le rayon de la plus petite sphère circonscrite/orthogonale à un simplexe.

Ainsi, pour ces calculs spécifiques aux Alpha-formes dans l'espace, il est nécessaire de caractériser cette plus petite sphère circonscrite/orthogonale à un simplexe, dans chacune des dimensions 1, 2 et 3. Considérons un simplexe $\sigma_{\mathcal{T}}$ de dimension $k = 1 \cdots 3$, dont l'ensemble des sommets est $\mathcal{T} = \{(p_i, w_i)\}_{i=1\cdots k}$. Dans le cas des Alpha-formes classiques, il suffira de prendre tous les poids w_i nuls. Notons c le centre de la plus petite sphère circonscrite/orthogonale à $\sigma_{\mathcal{T}}$ et w_c le carré de son rayon.

* Quand $k = 1$, on considère $\sigma_{\mathcal{T}}$ l'arête d'extrémités $\mathcal{T} = \{(p_0, w_0), (p_1, w_1)\}$. Par un argument de symétrie, on peut assurer que c est aligné avec p_0 et p_1. On a donc le système :

$$\begin{cases} \overrightarrow{p_0c} = k\overrightarrow{p_0p_1} \\ \|\overrightarrow{p_0c}\|^2 - w_0 - w_c = 0 \\ \|\overrightarrow{p_1c}\|^2 - w_1 - w_c = 0 \end{cases}$$

  Ainsi on a $k = \frac{\|\overrightarrow{p_0p_1}\|^2 - w_1 + w_0}{2\|\overrightarrow{p_0p_1}\|^2}$ et $w_c = \left(\frac{\|\overrightarrow{p_0p_1}\|^2 - w_1 + w_0}{2\|\overrightarrow{p_0p_1}\|}\right)^2 - w_0$.
  On peut aussi décrire le test d'*attachement* pour cette arête, qui consiste à s'assurer que, pour chaque triangle incident (p_0, p_1, q), on vérifie bien $\pi(c, q) > 0$, pour déclarer l'arête non-attachée. On remarquera que :

$$\pi(c, q) = \|\overrightarrow{p_0q}\|^2 - 2k\overrightarrow{p_0p_1}.\overrightarrow{p_0q} + w_0 - w_q$$

* Quand $k = 2$, on observe $\sigma_{\mathcal{T}}$ le triangle de sommets $\mathcal{T} = \{(p_0, w_0), (p_1, w_1), (p_2, w_2)\}$. Par un argument de symétrie, on peut assurer que c est coplanaire avec p_0, p_1 et p_2. On a donc le système :

$$\begin{cases} \overrightarrow{p_0c} = \lambda\overrightarrow{p_0p_1} + \mu\overrightarrow{p_0p_2} \\ \|\overrightarrow{p_0c}\|^2 - w_0 - w_c = 0 \\ \|\overrightarrow{p_1c}\|^2 - w_1 - w_c = 0 \\ \|\overrightarrow{p_2c}\|^2 - w_2 - w_c = 0 \end{cases}$$

  On a alors la résolution du système donnée par :

$$\begin{aligned} \lambda &= \frac{(\|\overrightarrow{p_0p_1}\|^2 - w_1 + w_0)\|\overrightarrow{p_0p_2}\|^2 - (\|\overrightarrow{p_0p_2}\|^2 - w_2 + w_0)\overrightarrow{p_0p_1}.\overrightarrow{p_0p_2}}{2(\|\overrightarrow{p_0p_1}\|^2\|\overrightarrow{p_0p_2}\|^2 - (\overrightarrow{p_0p_1}.\overrightarrow{p_0p_2})^2)} \\ \mu &= \frac{(\|\overrightarrow{p_0p_2}\|^2 - w_2 + w_0)\|\overrightarrow{p_0p_1}\|^2 - (\|\overrightarrow{p_0p_1}\|^2 - w_1 + w_0)\overrightarrow{p_0p_1}.\overrightarrow{p_0p_2}}{2(\|\overrightarrow{p_0p_1}\|^2\|\overrightarrow{p_0p_2}\|^2 - (\overrightarrow{p_0p_1}.\overrightarrow{p_0p_2})^2)} \\ w_c &= \lambda^2\|\overrightarrow{p_0p_1}\|^2 + \mu^2\|\overrightarrow{p_0p_2}\|^2 + 2\lambda\mu\overrightarrow{p_0p_1}.\overrightarrow{p_0p_2} - w_0 \end{aligned}$$

On peut aussi écrire le test d'*attachement* pour ce triangle, qui consiste à s'assurer que, pour chaque tétraèdre incident (p_0, p_1, p_2, q), on vérifie $\pi(c, q) > 0$, pour déclarer le triangle non-attaché. On remarquera que :

$$\pi(c,q) = \|\overrightarrow{p_0q}\|^2 - 2\lambda\overrightarrow{p_0p_1}.\overrightarrow{p_0q} - 2\mu\overrightarrow{p_0p_2}.\overrightarrow{p_0q} + w_0 - w_q$$

* Quand $k = 3$, on a $\sigma_{\mathcal{T}}$ le tétraèdre de sommets $\mathcal{T} = \{(p_0, w_0), (p_1, w_1), (p_2, w_2), (p_3, w_3)\}$. On a donc le système :

$$\begin{cases} \|\overrightarrow{p_0c}\|^2 - w_0 - w_c = 0 \\ \|\overrightarrow{p_1c}\|^2 - w_1 - w_c = 0 \\ \|\overrightarrow{p_2c}\|^2 - w_2 - w_c = 0 \\ \|\overrightarrow{p_3c}\|^2 - w_3 - w_c = 0 \end{cases} \iff \begin{cases} w_c = \|\overrightarrow{p_0c}\|^2 - w_0 \\ 2\overrightarrow{p_0p_1}.\overrightarrow{p_0c} = \|\overrightarrow{p_0p_1}\|^2 - w_1 + w_0 \\ 2\overrightarrow{p_0p_2}.\overrightarrow{p_0c} = \|\overrightarrow{p_0p_2}\|^2 - w_2 + w_0 \\ 2\overrightarrow{p_0p_3}.\overrightarrow{p_0c} = \|\overrightarrow{p_0p_3}\|^2 - w_3 + w_0 \end{cases}$$

On note $W_i = \|\overrightarrow{p_0p_i}\|^2 - w_i + w_0$ et (x_i, y_i, z_i) les coordonnées du vecteur $\overrightarrow{p_0p_i}$ pour $i = 1 \cdots 3$. Alors, en utilisant les formules de CRAMER, on résout le système précédent, grâce aux sous-déterminants suivants :

$$\Delta = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix},\ \Delta_1 = \begin{vmatrix} W_1 & y_1 & z_1 \\ W_2 & y_2 & z_2 \\ W_3 & y_3 & z_3 \end{vmatrix},\ \Delta_2 = \begin{vmatrix} x_1 & W_1 & z_1 \\ x_2 & W_2 & z_2 \\ x_3 & W_3 & z_3 \end{vmatrix},\ \Delta_3 = \begin{vmatrix} x_1 & y_1 & W_1 \\ x_2 & y_2 & W_2 \\ x_3 & y_3 & W_3 \end{vmatrix}.$$

Ainsi, on a $c = (\frac{\Delta_1}{2\Delta}, \frac{\Delta_2}{2\Delta}, \frac{\Delta_3}{2\Delta}) + p_0$ et $w_c = \frac{\Delta_1^2 + \Delta_2^2 + \Delta_3^2}{4\Delta^2} - w_0$.

On peut facilement déduire de ce qui précède l'équivalent en dimension 2, ou même une généralisation aux dimensions supérieures [Ede92] (que nous ne détaillerons pas ici). De plus, il est à noter que l'expression de ces prédicats n'est valable que pour des points en position générale. Évidemment, la mise en pratique du calcul des Alpha-formes nécessite de traiter ces configurations spéciales, ce qui est, naturellement, le cas dans l'implantation proposée au sein de CGAL.

2.4.4 *Spécifications*

Le choix dans l'implantation des triangulations de ne représenter explicitement que les sommets et les simplexes de dimension maximale n'a pas permis d'offrir, en pratique, une implantation capable de traiter les dimensions 2 et 3. On notera, en effet, qu'un triangle pour une triangulation en dimension 2 est représenté explicitement par CGAL, alors que sa représentation pour une triangulation en dimension 3 est implicite et correspond à un couple composé d'un tétraèdre incident et de l'indice du sommet opposé au triangle dans ce tétraèdre. Il y a donc deux implantations spécialisées dans le plan et dans l'espace pour le calcul des Alpha-formes. La figure 2.10 présente les liens qui unissent les différentes classes du module de calcul des Alpha-formes en 3D avec les classes calculant la triangulation sous-jacente choisie. Pour plus de détails, on pourra se référer à la documentation CGAL fournie en annexes A pour la dimension 2 et B pour la dimension 3.

En fait, on peut considérer les Alpha-formes comme des triangulations enrichies de données précalculées. Ainsi, une famille d'Alpha-formes peut être représentée par une classe dérivant de sa triangulation sous-jacente, contenant en plus le spectre de la famille. De plus, pour représenter l'intervalle caractéristique de chaque simplexe vis à vis de la famille d'Alpha-formes, un sommet d'une Alpha-forme est représenté par une classe dérivée d'un sommet de triangulation mais

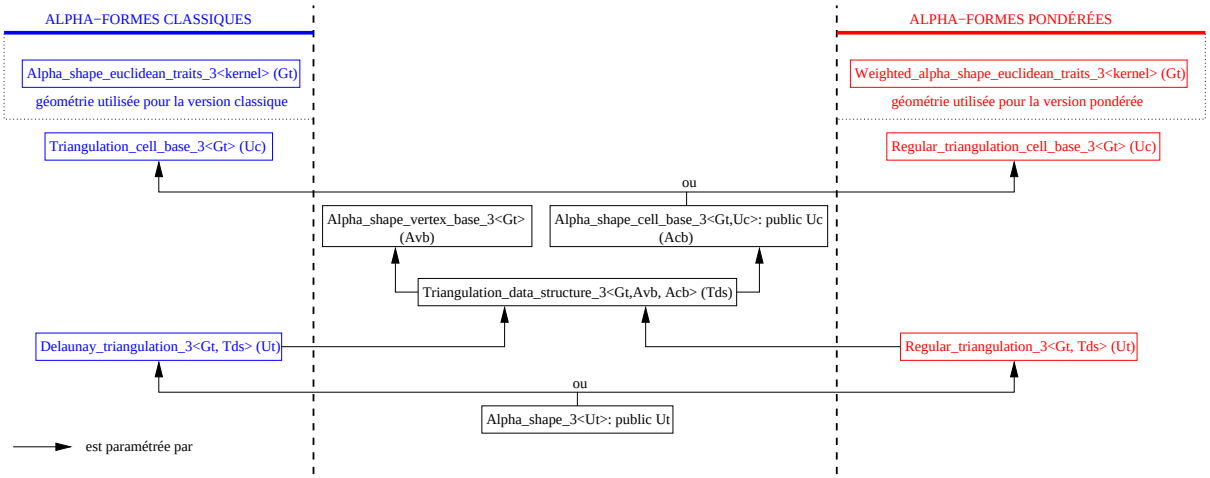


FIG. 2.10 – Illustration : interactions des différentes classes du module de calcul d'Alpha-formes avec ceux de la triangulation sous-jacente.

contenant en plus son intervalle caractéristique. De même, la cellule de dimension maximale est dérivée d'une cellule de la triangulation sous-jacente, mais en plus de son propre intervalle caractéristique, elle contient des méthodes d'accès aux intervalles des simplexes qui lui sont incidents.

2.4.5 *Détails d'implantation*

La version des Alpha-formes présentée par CGAL est, pour le moment, statique. En effet, la mise à jour de l'ensemble des informations nécessaires à l'établissement du spectre de la famille est une opération coûteuse. Et il convient de garder à l'esprit que l'intérêt du calcul de l'ensemble de la famille des Alpha-formes n'est réel que si on a beaucoup de requêtes sur des α-formes particulières à effectuer.

Cependant, afin de permettre la réalisation d'une version incrémentale future si elle s'avérait utile pour certaines applications, les structures employées, pour représenter le spectre de la famille d'Alpha-formes, sont dynamiques ; concrètement, il s'agit d'une implantation de queue de priorité fournie par la *STL* (*i.e. multimap*).

2.4.6 *Fonctionnalités*

La version classique

Les exemples que nous allons détailler, à titre d'illustrations, sont présentés, arbitrairement, pour les Alpha-formes en dimension 3, les fonctionnalités des classes implantant les Alpha-formes dans le plan et dans l'espace étant identiques pour ce que nous allons évoquer[10]. Le lecteur pourra se reporter aux annexes A et B pour plus de détails.

Dans sa version classique, le calcul des Alpha-formes s'opère évidemment avec une triangulation sous-jacente de DELAUNAY *Dt*. Cette triangulation (cf. [CGA01]) doit être paramétrée

[10] En fait, les différences sont essentiellement techniques, car héritées des choix opérés pour l'implantation des triangulations en 2D et en 3D.

par une *Geometric_traits* et une *Triangulation_data_structure*. Pour la *Geometric_traits*, il est nécessaire de fournir une classe contenant les prédicats et constructions spécifiques au calcul des Alpha-formes (cf. 2.4.3). On pourra utiliser *CGAL::Alpha_shape_euclidean_traits_3*, notée *Gt*, qui devra être paramétrée, à son tour, par un noyau de CGAL. Pour la *Triangulation_data_structure*, il convient de choisir des sommets et des cellules enrichies pour le calcul des Alpha-formes, par exemple *CGAL::Alpha_shape_vertex_base_3<Gt>* and *CGAL::Alpha_shape_cell_base_3<Gt,Tc>*, où *Tc* est une cellule valide pour une triangulation. On peut maintenant donner un exemple d'instanciation pour le calcul d'une famille d'Alpha-formes en dimension 3 :

```
typedef CGAL::Cartesian<double>                  K;
typedef CGAL::Alpha_shape_euclidean_traits_3<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_3<Gt>     Av;

typedef CGAL::Triangulation_cell_base_3<Gt>     Tc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Tc>    Ac;

typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Delaunay_triangulation_3<Gt,Tds>         Dt;
typedef CGAL::Alpha_shape_3<Dt>                        Alpha_shape_3;
```

La version pondérée

Pour les Alpha-formes pondérées, la triangulation sous-jacente, *Dt*, sera une triangulation régulière, implantée dans CGAL par la classe *CGAL::Regular_triangulation_3*. Il conviendra alors d'utiliser la cellule requise par ce type de triangulation, *CGAL::Regular_triangulation_cell_base_3<Gt>*, capable de contenir des sommets non apparents, pour instancier la classe *CGAL::Alpha_shape_cell_base_3*. Voici un exemple d'instanciation d'une classe calculant les Alpha-formes pondérées :

```
typedef CGAL::Cartesian<double>                            K;
typedef CGAL::Weighted_alpha_shape_euclidean_traits_3<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_3<Gt>         Av;

typedef CGAL::Regular_triangulation_cell_base_3<Gt> Rc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Rc>        Ac;

typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Regular_triangulation_3<Gt,Tds>          Rt;
typedef CGAL::Alpha_shape_3<Rt>                        Alpha_shape_3;
```

2.4.7 *Exemples d'utilisation*

Remarquons que le calcul des Alpha-formes bénéfice pleinement de toutes spécialisations possibles pour le calcul des triangulations. Ainsi, pour de gros échantillons de points (disons plus de 10.000 points), on pourra utiliser la version hiérarchique de la triangulation de Delaunay. Cet exemple illustre bien les avantages que CGAL tire de l'utilisation de la programmation

générique. Suivant les besoins, on peut à loisir choisir l'algorithme de calcul le plus efficace pour la triangulation de DELAUNAY et l'employer pour le calcul des Alpha-formes, simplement en modifiant quelques lignes de l'en-tête du programme. Détaillons l'instanciation d'une classe spécialisée pour le calcul des Alpha-formes classiques sur de gros échantillons :

```
typedef CGAL::Cartesian<double>                 K;
typedef CGAL::Alpha_shape_euclidean_traits_3<K> Gt;

// Definition du sommet approprie pour utiliser la hierarchie
typedef CGAL::Alpha_shape_vertex_base_3<Gt>              Avb;
typedef CGAL::Triangulation_hierarchy_vertex_base_3<Avb> Av;

typedef CGAL::Triangulation_cell_base_3<Gt>  Tc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Tc> Ac;

typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Delaunay_triangulation_3<Gt,Tds>         Dt;

// Instanciation de la triangulation de Delaunay hierarchique
typedef CGAL::Triangulation_hierarchy_3<Dt>            Ht;
typedef CGAL::Alpha_shape_3<Ht>                        Alpha_shape_3;
```

De même, dans le cas de données en coordonnées flottantes présentant des configurations proches de cas dégénérés, on pourra avec profit utiliser des prédicats filtrés grâce à une arithmétique spécialisée. Pour ceci, il suffit juste d'utiliser un noyau correctement paramétré, sans rien modifier au reste de la déclaration du type *Alpha_shape_3*, ni évidemment à l'utilisation que l'on peut en faire.

```
// Instanciation d'un noyau utilisant des predicats filtres
typedef CGAL::Filtered_exact<double, leda_real> NT;
typedef CGAL::Cartesian<NT>                     K;

// Simple utilisation des Alpha-formes avec ce noyau
typedef CGAL::Alpha_shape_euclidean_traits_3<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_3<Gt>  Av;

typedef CGAL::Triangulation_cell_base_3<Gt>  Tc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Tc> Ac;

typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Delaunay_triangulation_3<Gt,Tds>         Dt;
typedef CGAL::Alpha_shape_3<Dt>                        Alpha_shape_3;
```

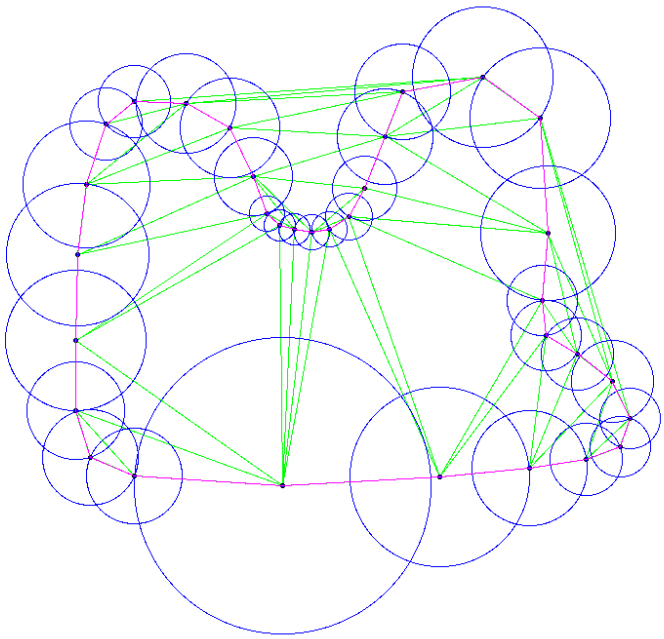


FIG. 2.11 – Illustration : initialisation des poids au plus proche voisin pour un échantillon 2D non-uniforme.

2.4.8 *Applications*

Heuristique d'initialisation des poids dans le plan

Présentons, tout d'abord, une application pour les Alpha-formes pondérées dans le plan, la reconstruction de courbes. Dans le cas d'échantillons de points uniformes suffisamment denses, on peut trouver une valeur pour α qui correspond à une reconstruction satisfaisante [BB97]. Les conditions d'échantillonnage requises sont malheureusement assez contraignantes, il faut, en effet, disposer d'un échantillon où le pas est celui exigé par la zone la plus difficile à reconstruire. Sans rentrer dans plus de détails, ceci s'avère rarement être le cas en pratique et, si tel est le cas, d'autres méthodes (on peut songer au *Crust* [ABE98] ou à la version 2D de la méthode développée au chapitre 3, entre autres) sont alors capables de fournir une réponse équivalente plus rapidement. Pour pouvoir traiter le cas d'échantillons non uniformes, on dispose, comme précédemment évoqué, des Alpha-formes pondérées. Les échantillons tels qu'ils sont disponibles traditionnellement sont constitués de points seuls. Le problème est alors de trouver une méthode permettant d'affecter à chaque point un poids dans le but d'aider à la reconstruction. Cet enjeu capital dans l'utilisation des Alpha-formes pondérées sur des jeux de données réels n'est pas du tout évoqué dans la littérature. Tentons d'y apporter une réponse pratique.

Dans le cas où l'objet à reconstruire est une courbe lisse, fermée et sans singularité, on sait que tout point p de l'échantillon devra être relié à exactement deux voisins dans la reconstruction. Pour tout échantillon acceptable, une idée simple et naturelle est alors que le plus proche voisin de p est nécessairement l'un de ces deux points. Exploitons ce fait pour proposer une heuristique d'initialisation des poids. On attribue à chaque point un poids égal au carré de la distance à son plus proche voisin. Pour ce faire, on pourra évidemment utiliser avec profit une triangulation

de DELAUNAY. Les poids ainsi attribués, on peut alors faire varier le paramètre α pour tenter d'obtenir une bonne reconstruction. Cette approche apporte de bonnes réponses en pratique (figure 2.11).

Notons, toutefois, qu'il existe des cas où cette heuristique ne suffira pas pour obtenir un résultat acceptable. Il suffit pour cela de construire un échantillon $\mathcal{S}_\varepsilon$ en dédoublant les points d'un échantillon non uniforme, $\mathcal{S}$, le long de la courbe d'origine, les doubles étant à une distance de ε des points originaux (ε très petit). Le poids attribué à chaque point, par l'heuristique appliquée à $\mathcal{S}_\varepsilon$, est alors exactement ε^2 et l'utilisation d'Alpha-formes pondérées sur l'échantillon de points dédoublés exactement équivalente à l'utilisation d'Alpha-formes simples sur l'échantillon de départ $\mathcal{S}$.

Cet exemple d'heuristique, bien que relativement efficace en pratique, montre que le problème de l'initialisation des poids est difficile même en dimension 2. Il est encore plus difficile à aborder dans l'espace puisque un point possède un nombre non déterminé de voisins sur une surface, même lisse, sans singularité et sans bord.

Reconstruction pour des échantillons uniformes

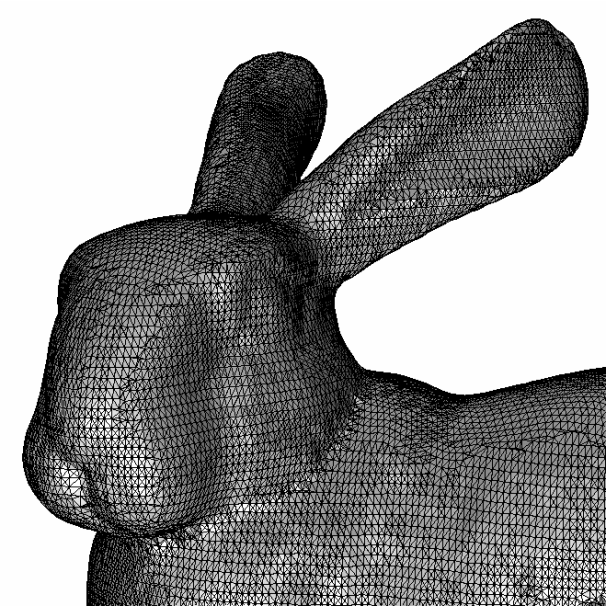


FIG. 2.12 – Résultat : un exemple d'échantillon de points uniforme, la lapin de Stanford.

Comme évoqué ci-avant, les conditions d'échantillonnage requises pour obtenir un réponse satisfaisante grâce aux Alpha-formes dans le plan sont drastiques. Observons, tout d'abord, le comportement des Alpha-formes sur un échantillon quasi idéal (*i.e.* très dense, uniforme, cf. figure 2.12) dans l'espace, le *lapin* de l'université de Stanford (grand classique dans l'illustration des méthodes de reconstruction). On trouvera, figure 2.15, une illustration du comportement de la famille des Alpha-formes (version singulière) quand α décroît sur $\mathbb{R}^+$. En version régulière, les Alpha-formes ne permettent pas de trouver de réponse satisfaisante ; en effet, des trous apparaissent sur le corps du lapin avant que les oreilles soient séparées.

Par contre, il semble, à première vue, qu'en mode singulier, on puisse sélectionner un α optimal (si on en prend un légèrement plus petit, des trous apparaissent) permettant d'obtenir

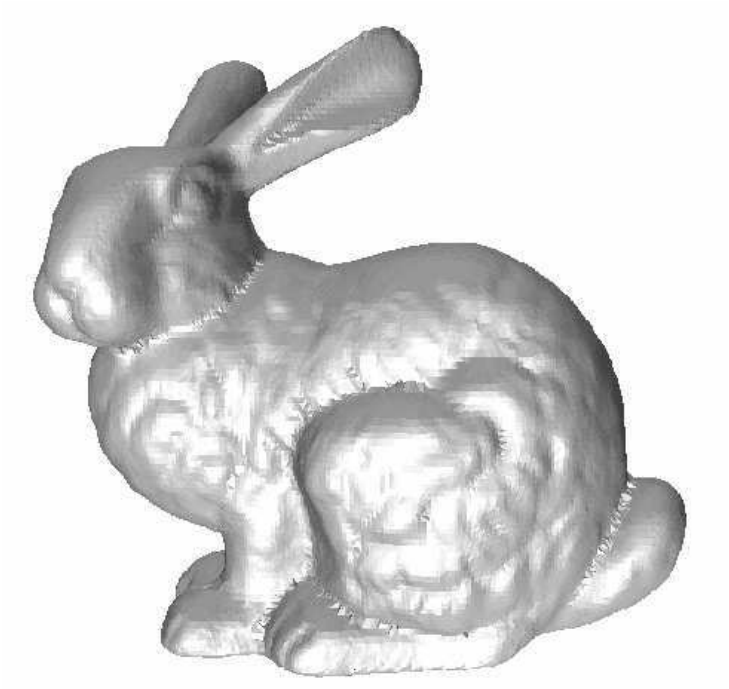


FIG. 2.13 – Résultat : reconstruction du lapin par Alpha-formes en mode général, pour un α optimal.

une représentation correcte du lapin (figure 2.13). Malheureusement, un examen plus approfondi, révèle que, pour ce α, l'objet reconstruit n'est pas partout une «vraie» surface : elle possède, en fait, des singularités. De l'affichage de la version régulière (qui ne conserve que des volumes) pour ce même α (figure 2.14), on peut déduire que les facettes n'ayant pas disparu par rapport à la version singulière cachent en fait de petits volumes[11].

Nous voilà, pour la première fois confrontés aux *slivers* (notion que l'on retrouvera «malheureusement»[12] dans le chapitre 3). Ces tétraèdres, dont les sommets sont quasi coplonaires, quasi cocycliques, et dont les quatre faces relient des points voisins sur la surface, représentent une difficulté supplémentaire spécifique à la dimension 3. Faute de reconstruction totalement correcte, les Alpha-formes permettent ici de montrer l'existence des *slivers*. Remarquons, de plus, que les quatre facettes d'un *sliver* ont bien leur plus petite sphère circonscrite vide de points du même ordre de grandeur, par définition des Alpha-formes).

[11] L'affichage des arêtes exhibe des tétraèdres plats matérialisés par des quadrilatères dont les diagonales se croisent.

[12] Sans eux, le problème de la reconstruction serait tellement plus simple...

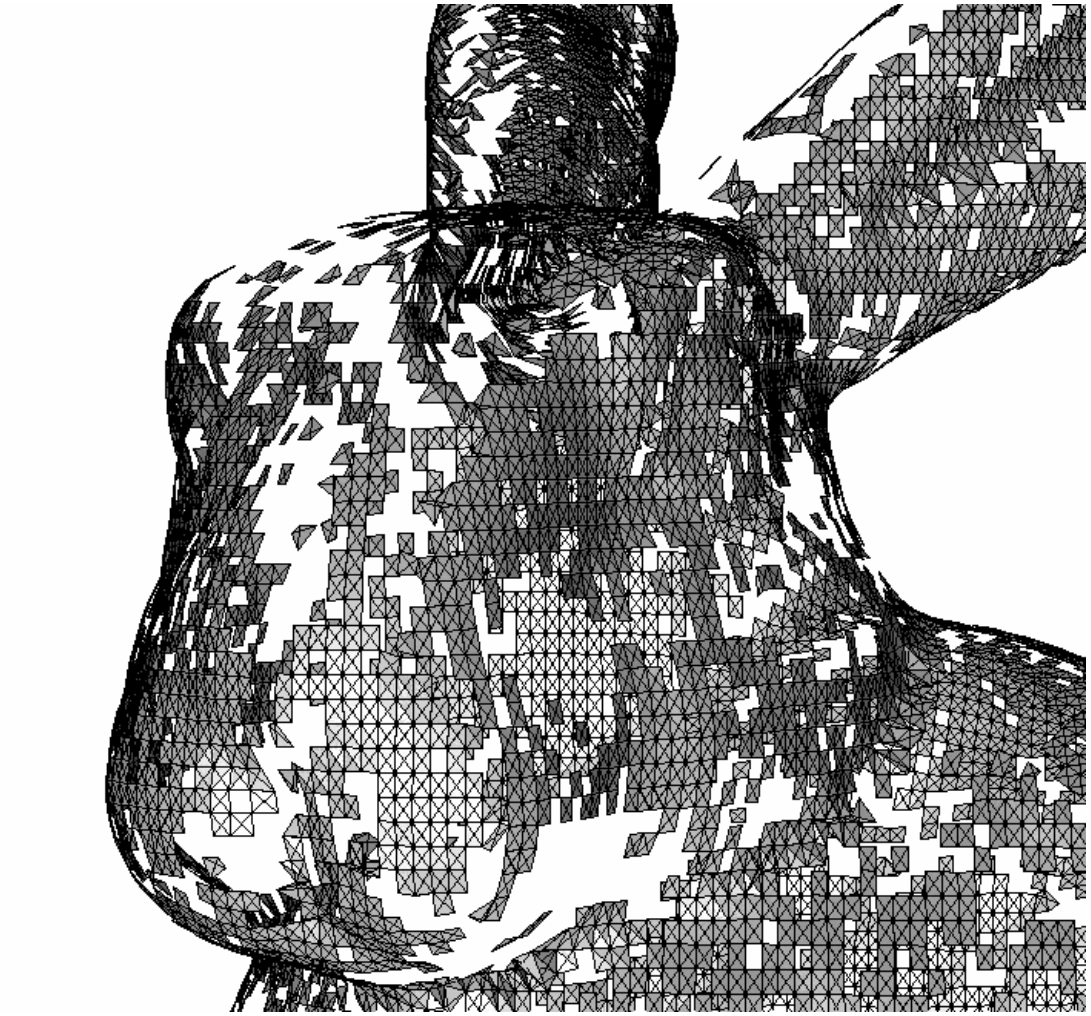


FIG. 2.14 – Résultat : mode régulier pour la reconstruction du lapin par Alpha-formes présentée figure 2.13, mise en évidence des *slivers*.

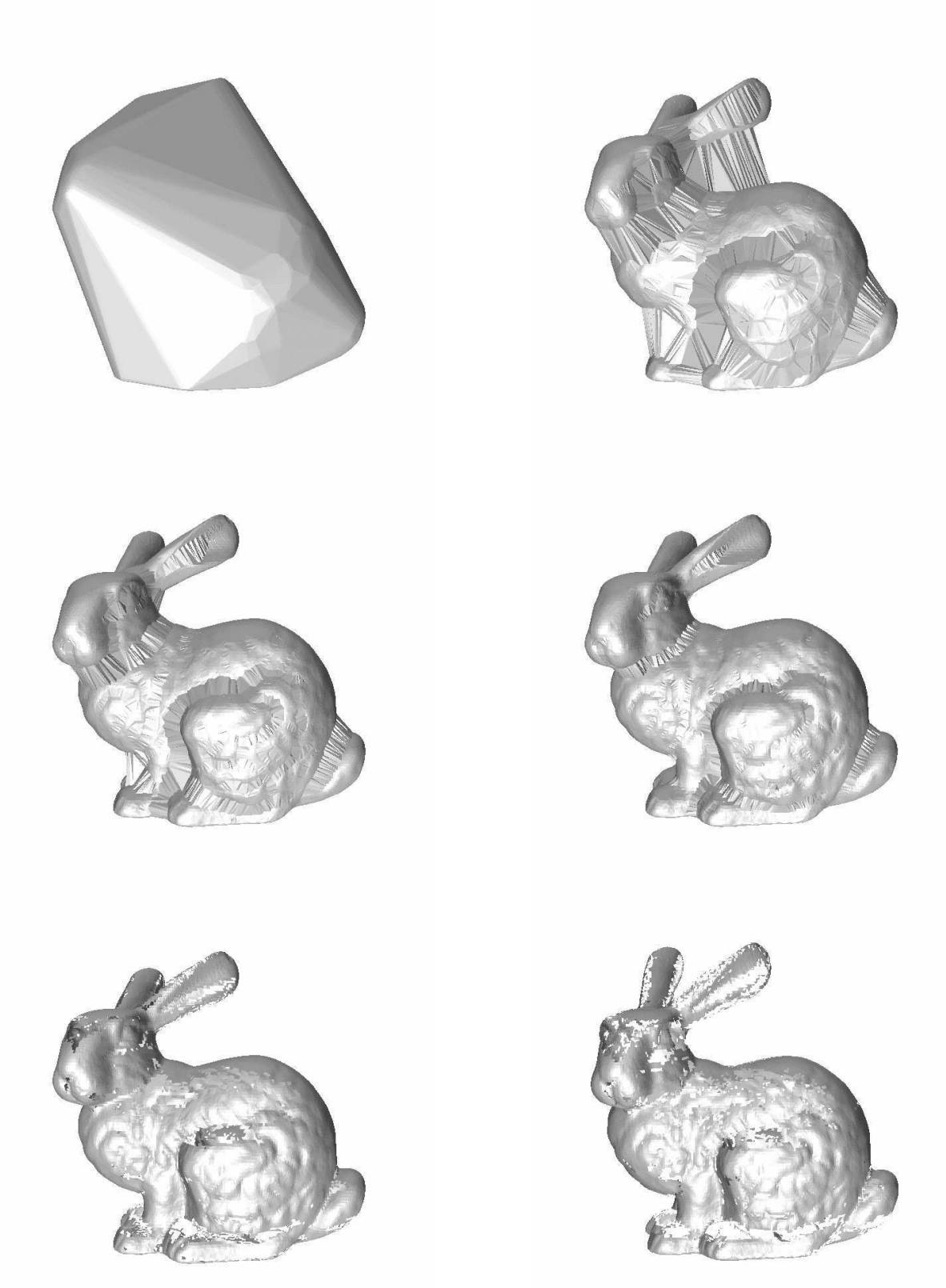


FIG. 2.15 – Illustration : évolution des Alpha-formes pour α décroissant.

§2.5 Les voisins naturels

2.5.1 Introduction

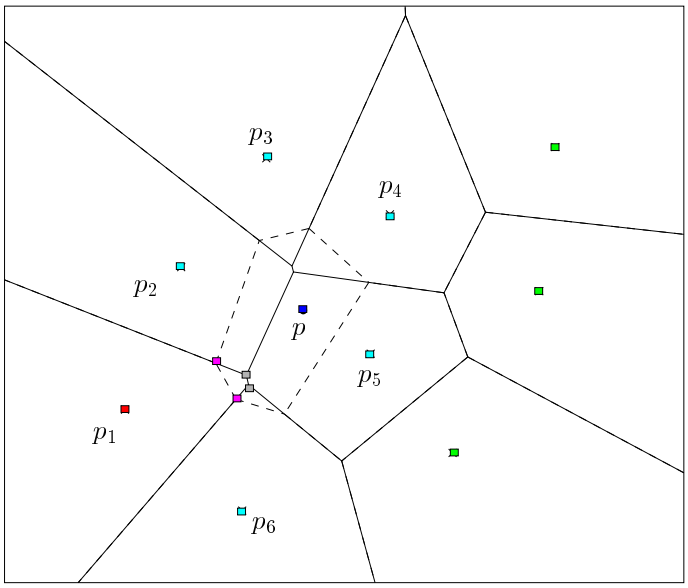


FIG. 2.16 – Illustration : $\{p_i\}_{i=1\cdots5}$, l'ensemble des voisins naturels du point p.

Les *voisins naturels* et les *coordonnées naturelles* qui leur sont associées ont été introduits par R. SIBSON, originellement dans le but de définir une interpolation de données échantillonnées sur une fonction à deux variables [Sib81]. S'agissant d'un système de coordonnées barycentrique, on peut donc interpoler exactement des données échantillonnées sur une fonction linéaire. Ce qui a retenu notre attention sur ce schéma d'interpolation parmi d'autres est que l'évaluation de la fonction d'interpolation en un point ne fait intervenir qu'une partie des données. Cet aspect est primordial pour une application informatique efficace par rapport à des systèmes de coordonnées qui feraient intervenir tous les points de l'échantillon. D'autres schémas possèdent des propriétés similaires, telles les coordonnées de LAPLACE, ainsi que d'autres systèmes définis dans [HS00]. Cependant, comparé à ses concurrents, il offre des propriétés de continuité, de dérivabilité, particulièrement intéressantes, tout en permettant un algorithme de calcul raisonnable. En effet, étant intimement liée au diagramme de VORONOI, la mise en œuvre du calcul se révèle efficace dans le plan [Alf89, Wat92, Wat94] et aussi dans l'espace [BC00].

2.5.2 Définition

Considérons un ensemble de points $\mathcal{S}$. Notons $\mathfrak{V}(\mathcal{S})$, le diagramme de VORONOI de ces points. On définit alors pour tout point de l'espace ses *voisins naturels* et leurs coordonnées associées ainsi :

Définition 2.4 *Pour tout point p intérieur à l'enveloppe convexe de $\mathcal{S}$, on nomme* voisins naturels *les points de $\mathcal{S}$ auxquels p serait relié par une arête dans le* dual *de $\mathfrak{V}$. Notons $\mathfrak{V}(p)$, la cellule de p si on l'insérait dans $\mathfrak{V}$. Pour chacun de ses voisins naturels p_i, on peut définir un*

coefficient, que l'on nommera coordonnée naturelle*, égal à l'aire de l'intersection entre $\mathfrak{V}(p)$ avec la cellule de p_i, normalisée par l'aire de $\mathfrak{V}(p)$ toute entière (figure 2.16).*

$$\lambda_{\mathrm{i}} = \mathrm{aire}(\mathfrak{V}(p_i) \cap \mathfrak{V}(p))/\mathrm{aire}(\mathfrak{V}(p))$$

2.5.3 Propriétés

Les coordonnées naturelles présentent plusieurs propriétés remarquables. Présentons les, tout d'abord informellement, afin de mieux saisir leur intérêt dans le cadre de l'interpolation et plus précisément pour une application informatique au problème de la reconstruction. Tout d'abord, ce schéma répond à un enjeu d'efficacité informatique de par *la localité* : en d'autres termes, seuls des points «autour» du point considéré soient pris en compte. De plus, plus mathématiquement, les coordonnées naturelles sont $\mathscr{C}^1$ sauf aux points de l'échantillon. Enfin, la propriété du *barycentre* : soit $\{p_i\}_{i=1\cdots n_p}$ les voisins naturels de p, alors p est le barycentre des points p_i affectés de leurs coordonnées naturelles.

Pour être plus correct, formalisons un peu ce que nous venons d'évoquer. Si on considère $\mathcal{S}$, un ensemble de n points $\{p_i\}_{i=1\cdots n}$, on peut alors définir la famille de fonctions suivante : pour i variant de 1 à n, λ_{i} est la fonction égale à la coordonnée naturelle associée à p_i si p_i est voisin naturel du point d'évaluation et à zéro sinon. Alors le système de coordonnées naturelles défini par cette famille de fonctions associées au points de $\mathcal{S}$ vérifient les propriétés suivantes :

Propriété 2.4

(i) La localité *: pour tout point p intérieur à l'enveloppe convexe de $\mathcal{S}$, les fonctions λ_{i} prennent toutes la valeur zéro, exceptées les fonctions associées aux voisins naturels de p.*

(ii) Chaque fonction λ_{i} est $\mathscr{C}^0$ partout et $\mathscr{C}^1$ sauf en p_i ; notons que $\lambda_{\mathrm{i}}(p_j) = \delta_{ij}$, où δ_{ij} est le symbole de Kronecker.

(iii) Pour tout point p intérieur à l'enveloppe convexe de $\mathcal{S}$, on a l'égalité barycentrique : $p = \sum_{p_i \in \mathcal{S}} \lambda_{\mathrm{i}}(p).p_i$.

2.5.4 Algorithme de calcul

Le calcul des voisins naturels peut être effectué sans difficulté grâce à la triangulation de Delaunay des points de $\mathcal{S}$. En effet, triangulation de Delaunay et diagramme de Voronoi étant *duaux* l'un de l'autre, on peut transcrire la définition 2.4 pour ne l'exprimer que grâce à la triangulation de Delaunay. Pour le calcul dans le plan, on peut se référer à [Alf89, Wat92, Wat94], dans l'espace à [BC00].

En ce qui nous concerne, c'est le calcul dans le plan que nous allons utiliser. Soit $\mathcal{S}$ un échantillon de points, $\mathfrak{D}$ la triangulation de Delaunay de $\mathcal{S}$ et p un point dont on veut calculer les voisins naturels et les coefficients qui leur sont associés :

* les voisins naturels de p sont les points du bord de la région composée des triangles en conflit avec le point p ;
* considérons p_i, un des voisins naturels de p, il s'agit donc de calculer le coefficient naturel qui lui est associé. Rappelons la définition de cette quantité :

$$\lambda_{\mathrm{i}}(p) = \mathrm{aire}(\mathfrak{V}(p_i) \cap \mathfrak{V}(p))/\mathrm{aire}(\mathfrak{V}(p))$$

On décrira simplement l'algorithme de calcul de l'aire de $\mathfrak{V}(p_i) \cap \mathfrak{V}(p)$, la portion de la cellule de p_i englobée par la cellule de p, si on l'insérait, dans le diagramme de VORONOI. La normalisation étant ensuite triviale, puisque l'on a :

$$\text{aire}(\mathfrak{V}(p)) = \sum_i \text{aire}(\mathfrak{V}(p_i) \cap \mathfrak{V}(p))$$

Précisons quelques notations utilisées dans la description de l'algorithme qui va suivre, pour une arête e, on désignera par e^0 et par e^1 ses extrémités. De plus, on notera $\text{centre}_{\text{triangle}}$ la fonction calculant le centre du cercle circonscrit à un triangle et $\text{aire}_{\text{triangle}}$ la fonction calculant l'aire d'un triangle.

Algorithme de calcul d'un coefficient naturel avant normalisation

Require: p un point requête et p_i un de ses voisins naturels.
Ensure: retourne $\alpha_i = \text{aire}(\mathfrak{V}(p_i) \cap \mathfrak{V}(p))$.
  soit e_i et e_f les deux arêtes adjacentes à p_i délimitant la région en conflit avec p.
  $\alpha_i \Leftarrow 0$;
  $v_1 \Leftarrow \text{centre}_{\text{triangle}}(e_i^0, e_i^1, p)$
  soit c un itérateur sur les facettes en conflit avec p adjacente à p_i ;
  $c \Leftarrow f_i$, la facette incidente à e_i en conflit avec p ;
  $v_2 \Leftarrow \text{dual}(f_i)$;
  repeat
    c++ ;
    $v_3 \Leftarrow \text{dual}(c)$;
    $\alpha_i \Leftarrow \alpha_i + \text{aire}_{\text{triangle}}(v_1, v_2, v_3)$;
    $v_2 \Leftarrow v_3$;
  until c non incidente à e_f
  $v_3 \Leftarrow \text{centre}_{\text{triangle}}(e_f^0, e_f^1, p)$;
  $\alpha_i \Leftarrow \alpha_i + \text{aire}_{\text{triangle}}(v_1, v_2, v_3)$;
  return α_i ;

2.5.5 *Spécifications*

La mise en œuvre du calcul des voisins naturels ne présente pas de difficultés majeures. Il consiste juste à ajouter une fonctionnalité à la triangulation de DELAUNAY. En pratique, le choix a donc été fait de faire de la fonction de calcul du système de coordonnées naturelles une fonction membre *natural_coordinates* de la classe *CGAL::Delaunay_triangulation_2*. Par ailleurs on a définit une nouvelle face de base, chargée de conserver le centre de son cercle circonscrit, qui peut être utilisée à la place de la face standard. Ainsi, avec une face de base standard, le centre du cercle circonscrit sera recalculé à chaque requête, alors que la nouvelle face permet une évaluation paresseuse (*i.e.* le calcul n'est fait que s'il est demandé et conservé pour d'éventuelles requêtes futures).

Pratiquement, la méthode employée pour permettre les deux options, est transparente pour l'utilisateur. On a utilisé deux versions marquées pour le calcul du centre du cercle circonscrit à une face dont l'utilisation n'est commandée que par le fait d'avoir paramétré la triangulation de DELAUNAY par l'une ou l'autre des faces de base disponibles.

Par ailleurs, il a été nécessaire d'implanter un conteneur, permettant de retourner le système de coordonnées naturelles associé à un point, *CGAL::Barycentric_coordinates<Vh, Nt>*, où *Vh*

sont respectivement une *poignée* (*Handle*) sur un sommet et un type de nombre. Ainsi le prototype de la fonction membre de la triangulation de DELAUNAY chargé de réaliser le calcul du système de coordonnées naturelles, *Nc*, associé à un point p est :

bool natural_coordinates(const Point& p, Barycentric_coordinates& Nc) ;

Le *booléen* renvoyé est mis à *vrai* quand le point p est intérieur à l'enveloppe convexe des points insérés dans la triangulation de DELAUNAY et à *faux* dans le cas contraire. On pourra, ainsi, opérer un traitement adéquat dans le cas particulier où l'on cherche à prolonger le schéma d'interpolation hors de l'enveloppe convexe des points. Pour plus de détails, on pourra se référer à la documentation de CGAL.

2.5.6 *Fonctionnalités*

Détaillons donc des exemples d'instanciations pour les deux options offertes par l'implantation du calcul du système de coordonnées naturelles associées à un échantillon de points. Tout d'abord, si la mémoire est limitée ou que l'échantillon est modifié sans cesse, il est plus efficace d'utiliser une face de base classique. Voici un exemple d'instanciation puis d'utilisation, où on affiche à l'écran les coordonnées associées à un point :

```
typedef CGAL::Cartesian<double>  K;

typedef CGAL::Triangulation_vertex_base_2<K> Vb;

// Choix du type de face utilise suivant la version souhaitee (ici recalcul)
typedef CGAL::Triangulation_face_base_2<K>   Fb; // avec recalcul
// typedef CGAL::Voronoi_face_base_2<K>      Fb; // avec evaluation paresseuse

typedef CGAL::Triangulation_default_data_structure_2<K,Vb,Fb> Tds;
typedef CGAL::Delaunay_triangulation_2<K,Tds>                 Dt;

typedef Dt::Barycentric_coordinates Bc;          // type de retour
typedef Bc::Coordinate_iterator     Bc_it;

void natural_coordinates_pretty_print(const Dt& dt, const K::Point& p)
{
  Bc Nc;
  std::cout << "pour le point requete :" << p << std::endl;
  if (dt.natural_coordinates(p, Nc))
    {
      Bc_it Nc_it;
      Point O(0,0), G(0,0);
      for(Nc_it = Nc.begin(); Nc_it != Nc.end(); ++Nc_it)
        {
          Dt::Vertex_handle vh(Nc_it->first);
          std::cout << "la coordonnee associee au point "
                    << vh->point() << " est : "
                    << Nc_it->second/Nc.get_norm_coeff()
                    << std::endl;
```

```
          // Verification de la propriete du barycentre
          G = G + Nc_it->second/Nc.get_norm_coeff()*(vh->point()-O);
        }
      std::cout << "le barycentre calcule est " << G << std::endl;
    }
  else
    std::cout << "il est en dehors de l'enveloppe convexe." << std::endl;
}
```

Dans la cas où on opère de nombreuses requêtes sur un échantillon de points que l'on ne modifie pas ou que très rarement, il sera judicieux d'utiliser la version avec évaluation paresseuse des centres de cercle circonscrit. On utilisera alors simplement la face spéciale *CGAL::Voronoi_face_base_2* au lieu de la face classique. On notera que l'appel de la fonction de calcul des coordonnées naturelles est le même que dans le cas précédant.

Par ailleurs, remarquons que, là encore, on profite pleinement de toutes les optimisations possibles pour les triangulations de DELAUNAY. Ainsi, pour les gros échantillons de points (plus de 10.000 points), on pourra aussi utiliser la version hiérarchique de la triangulation de DELAUNAY. En outre, on utilisera avec profit un noyau filtré afin de traiter efficacement des échantillons de points présentant des configurations litigieuses que seul le calcul exact permet de distinguer. Présentons, à titre d'exemple, un instanciation de la triangulation de DELAUNAY dans le cas où on voudrait utiliser une arithmétique filtrée en même temps que la version hiérarchique de DELAUNAY et l'évaluation paresseuse des cercles circonscrits :

```
// Instanciation d'un noyau utilisant des predicats filtres
typedef CGAL::Filtered_exact<double, leda_real> NT;
typedef CGAL::Cartesian<NT>                     K;

// Definition du sommet approprie pour utiliser la hierarchie
typedef CGAL::Triangulation_vertex_base_2<K>            Vb;
typedef CGAL::Triangulation_hierarchy_vertex_base_2<Vb> Hv;

// Choix du type de face utilise
typedef CGAL::Voronoi_face_base_2<K>   Fb; // avec evaluation paresseuse

// Triangulation de Delaunay interne a la hierachie
typedef CGAL::Triangulation_default_data_structure_2<K,Hv,Fb> Tds;
typedef CGAL::Delaunay_triangulation_2<K,Tds>                 Idt;

// Instanciation de la triangulation de Delaunay hierarchique
typedef CGAL::Triangulation_hierarchy_2<Idt>                  Dt;

typedef Dt::Barycentric_coordinates Bc; // type de retour
typedef Bc::Coordinate_iterator     Bc_it;
```

Par ailleurs, on trouvera différentes applications reposant sur le calcul des voisins naturels dans le chapitre traitant de la reconstruction à partir de données organisées en sections (cf. chapitre 4). C'est pourquoi, nous ne nous attarderons pas plus ici sur les applications possibles des voisins naturels.

2.5.7 Étude des performances

Surcoût du calcul des coefficients naturels

Détaillons le coût du calcul des coefficients naturels par rapport à une simple insertion dans la triangulation de DELAUNAY. En effet, dans les deux cas, il s'agit d'effectuer des opérations sur les voisins naturels d'un point, ces opérations consistant en une simple mise à jour de la connectivité pour l'insertion et en des calculs d'aire pour le calcul des coefficients naturels. Le tableau 2.2 présente les résultats de cette étude comparative. On peut y lire le coût du calcul du système de coordonnées naturelles d'un point par rapport à son insertion dans une triangulation de DELAUNAY hiérarchique et robuste (utilisation de prédicats filtrés). Notons qu'il s'agit donc d'une implantation réaliste, qui sera, par exemple, utilisée dans l'implantation des algorithmes développés dans le chapitre 4.

Modèle	nombre de points	insertion seule (t_1)	insertion (t_1) + *natural_coordinates* (t_2)	(t_2/t_1)
in_square	1000	0,03	0,08	1,67
on_square	1000	0,04	0,1	1,5
in_disc	1000	0,03	0,08	1,67
on_circle	1000	0,38	0,74	0,95
in_square	5000	0,18	0,46	1,56
on_square	5000	0,3	0,68	1,27
in_disc	5000	0,18	0,47	1,61
on_circle	5000	1,22	2,41	0,98
in_square	10000	0,44	1,03	1,34
on_square	10000	0,97	2,04	1,1
in_disc	10000	0,4	1,01	1,52
on_circle	10000	2,69	5,01	0,86
in_square	50000	2,38	5,53	1,32
on_square	50000	13,89	29,47	1,12
in_disc	50000	2,28	5,43	1,38
on_circle	50000	37,13	81,99	1,21
in_square	100000	5,1	11,35	1,23
on_square	100000	48,25	100,67	1,09
in_disc	100000	4,95	11,47	1,32
on_circle	100000	144,9	318,33	1,2

TAB. 2.2 – Étude du coût du calcul des coordonnées naturelles par rapport à l'insertion dans une triangulation de DELAUNAY *hiérarchique* (temps de calcul CPU exprimés en secondes).

Pour représenter les différentes configurations possibles, on a choisi différents types de jeux de données. Le cas de points *random* dans un carré (*on_ square*) représente le cas optimal pour le calcul de DELAUNAY (configurations dégénérées improbables). Le cas des courbes, qui est intéressant pour la reconstruction, est représenté par le cas de points sur un cercle (*on_ circle*), un des pires pour DELAUNAY (toutes les configurations sont dégénérées), et le cas de points sur un carré (*on_ square*) plus favorable. Enfin, le cas de points dans un disque (*in_ disc*) représente un cas moyen pour DELAUNAY, où il peut y avoir des configurations dégénérées. Les temps CPU

fournis sont exprimés en secondes, ils correspondent à la version 2.3 de CGAL sur une machine équipée d'un processeur INTEL PENTIUM i686 1GHz et disposant de 1Go de RAM. Notons, de plus, que, du fait du calcul des voisins naturels, une boite englobante a été ajoutée.

L'analyse de ce tableau 2.2 révèle tout d'abord que le surcoût dû aux calculs des coordonnées naturelles n'excède pas $1,7$ et se révèle très avantageux pour les cas qui modélisent une courbe. Notons de plus, que lorsque le nombre de points augmente, le coût de la localisation devient plus grand, et qu'ainsi le surcoût des calculs numériques des voisins naturels se fait moins sentir, il tombe alors en dessous de $1,4$ avec la triangulation hiérarchique. Par ailleurs, on pourrait légitimement s'étonner de voir apparaître un surcoût inférieur à 1 pour le modèle *on_circle* jusqu'à 10.000 points. Pour mieux comprendre, une autre série de tests à été réalisée dans les mêmes conditions, mais avec une version non hiérarchique de la triangulation de DELAUNAY (cf. tableau 2.3).

Modèle	nombre de points	insertion seule (t_1)	insertion (t_1) + *natural_coordinates* (t_2)	(t_2/t_1)
in_square	1000	0,03	0,09	2
on_square	1000	0,08	0,2	1,5
in_disc	1000	0,03	0,09	2
on_circle	1000	0,37	0,78	1,1
in_square	5000	0,34	0,85	1,5
on_square	5000	1,32	2,94	1,23
in_disc	5000	0,32	0,81	1,53
on_circle	5000	1,77	4,09	1,31
in_square	10000	1,01	2,33	1,31
on_square	10000	9,96	22,31	1,24
in_disc	10000	0,97	2,28	1,35
on_circle	10000	4,42	10,96	1,48

TAB. 2.3 – Étude du coût du calcul des coordonnées naturelles par rapport à l'insertion dans une triangulation de DELAUNAY*simple* (temps de calcul CPU exprimés en secondes).

À la lumière du tableau 2.3, on s'aperçoit que la comparaison n'est pas si simple. En fait, le calcul des voisins naturels effectue des opérations semblables à celles réalisées pour une insertion dans une triangulation de DELAUNAY classique, par contre les opérations de mise à jour d'une triangulation de DELAUNAY hiérarchique, qui permettent une localisation plus rapide, sont plus complexes. Ainsi, la comparaison avec la version classique (cf. tableau 2.3) nous montre que le calcul des voisins naturels correspond bien à un surcoût. Ainsi, le tableau 2.2 démontre simplement que la version hiérarchique de la triangulation de DELAUNAY ne prend véritablement sa pleine mesure qu'une fois les 10.000 points dépassés[13]. En dessous de ce seuil, le calcul des voisins naturels est, dans certains cas, moins coûteux que la mise à jour de la hiérarchie de triangulations.

Par ailleurs, on peut comparer ces résultats à ceux décrits dans [BC00] qui concernent le cas de la dimension 3, où on voit apparaître un surcoût entre 5 et 10 pour le calcul des coordonnées naturelles dans l'espace grâce une étude semblable. Voilà donc un argument pour préférer utiliser les coordonnées naturelles en dimension 2 plutôt que dans l'espace pour notre méthode de reconstruction à partir de sections. On reviendra, ainsi, plus en détail dans le chapitre 4 sur la valorisation des calculs dans le plan.

[13] Comme annoncé dans la documentation CGAL, d'ailleurs.

Chapitre 3

Reconstruction à partir de nuages de points

Sommaire

§3.1 *Introduction*

3.1.1 *Les enjeux des méthodes dites générales*

Reconstruire des objets à partir de points mesurés sur leur surface, le problème semble posé. Cependant, est-ce vraiment la manière la plus intéressante de le formuler ? L'objectif utopique de ce chapitre pourrait être de présenter une méthode miraculeuse permettant de reconstruire toutes sortes d'objets à partir de n'importe quel type d'échantillon... Doux rêve... Non, on se contentera ici, de cerner des enjeux importants, déterminés à partir de jeux de données réelles, auxquels on s'attachera à fournir des réponses les plus satisfaisantes possibles. Voilà en quelques mots, la philosophie qui aura été la nôtre pour ce travail.

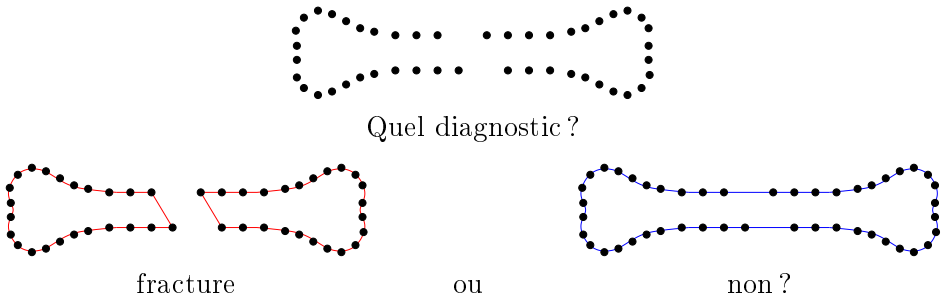


FIG. 3.1 – Illustration : un ensemble des points sans information supplémentaire.

Dans ce qui va suivre, nous considèrerons des échantillons de points non-uniformes ne possédant pas de structure, *i.e. désorganisés*. De tels échantillons seront encore appelés *nuages de points*. Le fait de ne considérer aucune structure sur les données revient à séparer, en deux tâches bien distinctes, la numérisation de données et la reconstruction proprement dite. Il est pourtant fondamental de comprendre que, sans autre considération, ceci serait voué à l'échec. Bien évidemment, on ne peut raisonnablement espérer reconstruire des objets à partir d'échantillons trop peu denses, sans présumer d'aucune information ni sur la nature de l'objet, ni sur la manière dont l'échantillon aura été mesuré. La figure 3.1 permet d'illustrer l'apport d'une connaissance complémentaire sur la nature des données : si l'on sait qu'il n'y a qu'un objet dans la scène le problème est résolu — l'os n'est pas brisé — alors que, sans cette information, la question est ouverte.

En fait, cette approche générale du problème trouve un intérêt, essentiellement, de par les progrès réalisés par les procédés de numérisation des objets. En effet, les échantillons sont dorénavant souvent suffisamment denses et le bruit sur les mesures effectuées suffisamment faible, pour ne pas avoir à considérer d'information complémentaire. Paradoxalement, c'est en fait là que se cache l'information *a priori* considérée par cette classe de méthodes. La possibilité de reconstruire un objet par une méthode générale est, en effet, intimement liée au fait que l'échantillonnage de cet objet soit «bon», à moins que l'on présume d'autre chose à un moment ou à un autre, ce qui spécialise alors inévitablement la méthode. En quelque sorte, disposer d'un «bon» échantillon de points est le pré-requis minimum au succès de toute méthode, dite générale. On supposera donc que l'échantillon a été mesuré sur un objet réel et qu'il est au moins «assez» dense.

Pour apprécier intuitivement, ce que l'on entend par «bon» échantillonnage, on peut considérer qu'il est nécessaire que, pour un point de l'échantillon, ses voisins sur la surface de l'objet

à reconstruire se doivent d'être aussi les points qui sont le plus proche de lui dans l'échantillon[14]. Malheureusement, cette condition nécessaire n'est pas facile à assurer. Actuellement, aucun des procédés de numérisation couramment utilisés n'est à même de satisfaire de tels critères. On sait juste numériser des données, parfois très précisément, mais les garanties sur la qualité de l'échantillonnage sont quasiment inexistantes. L'alternative consiste donc, dans tous les cas[15] où cela est possible, à sur-échantillonner les objets, dans l'espoir de numériser de «bons» échantillons. Cette tendance a pour conséquence de fournir des échantillons de plus en plus denses, des volumes de données toujours plus grands.

C'est dans ce cadre où les échantillons seront volumineux, les objets à reconstruire de nature variée, que les méthodes de reconstruction générales trouvent leur application. Outre la qualité de la reconstruction fournie, ce sont ces aspects de rapidité, de fiabilité, d'espace mémoire occupé, qui doivent être pris en compte pour évaluer leurs performances.

3.1.2 *Des nuages de points*

De nombreux domaines sont à même de fournir de tels échantillons. On peut citer l'imagerie médicale à des fins d'aide au diagnostic, de simulation ou d'assistance en chirurgie, ou encore le graphisme informatique, où, de plus en plus souvent, on cherche à incorporer des éléments réels dans des scènes, pour les jeux vidéos, les effets spéciaux au cinéma. On peut aussi penser aux tentatives de visite virtuelle du musée du Louvre, ou encore au projet Michelangelo [LPC+00] ayant pour objectif de scanner plusieurs statues de Michel Ange. L'industrie, pour des applications de *reverse engineering* par exemple, produit, de plus en plus couramment, des échantillons de données de tailles impressionnantes. La physique, la chimie, la géologie, les mathématiques ont aussi leurs propres besoins dans le domaine de la reconstruction d'objet.

Les techniques de numérisation sont, elles aussi, nombreuses et variées. On citera, entre autres, les scanners laser, la stéréo-vision, les images 3D — acquises par échographie, tomographie, imagerie par résonance magnétique (IRM), microscopie électronique (EM) — ou encore l'analyse d'ondes sismiques.

Cette profusion de domaines d'application, mêlée à la diversité des procédés d'acquisition de données, rend d'autant plus intéressante l'idée d'une méthode capable de reconstruire des objets de natures très différentes (surface fermée, à bord, à trous, topologie complexe) pour peu que l'échantillon soit «bon». De plus, d'autres applications découlent de la nature générale de ces méthodes. La compression de modèles géométriques, par exemple, peut être réalisée ainsi ; si l'on sait, en effet, reconstruire un modèle à partir de ses sommets, on peut alors se passer de la connectivité.

3.1.3 *Positionnement parmi les travaux antérieurs*

Essayons, ici, de décrire qualitativement les idées directrices qui sous-tendent l'approche que nous allons proposer. On pourra se référer à la partie 1.3 pour une étude générale des différentes contributions déjà apportées à la résolution de ce problème. Parmi les différentes idées qui émergent de cet état de l'art, certaines ont retenu plus particulièrement notre attention.

Tout d'abord, la méthode que nous allons présenter fait partie des méthodes de sélection dans la triangulation de Delaunay. D'une part, parce que, comme nos prédécesseurs, nous considérons que les conditions requises sur l'échantillon pour qu'une reconstruction acceptable

[14] Ainsi, si l'on dispose d'un «bon» échantillon de points pour la figure 3.1, on peut conclure que l'os est fracturé.

[15] Pour certaines applications, l'extraction des données est limitée pour des raisons de coût, en géologie par exemple, ou de risque, afin d'éviter une surexposition à des rayonnements pour les patients en imagerie médicale...

existe dans la triangulation de DELAUNAY représentent un seuil minimum en dessous duquel on peut douter qu'une méthode générale soit appropriée. D'autre part, les différentes propriétés de cette triangulation et surtout de son dual le diagramme de VORONOI, sont intimement liées à notre problème de reconstruction. En effet, en quelque sorte, du moins localement, reconstruire la surface autour d'un point de l'échantillon revient à sélectionner parmi les *plus proches voisins* de ce point, ceux qui sont aussi ses voisins sur l'objet originel.

Reprenons, pour fixer les idées, le fil directeur utilisé par les *Alpha-formes* [EM94]. Examinons la répartition des facettes présentes dans la triangulation de DELAUNAY, si on les classe, en ordre croissant, selon le rayon de leur plus petit cercle circonscrit vide de points. L'idée soutenue dans la notion d'Alpha-formes (cf. 2.4) est que les facettes de la reconstruction et celles à exclure seraient séparées par un saut de valeurs dans cette classification, il suffirait alors de trouver ce saut pour déterminer le résultat. Comme décrit dans [BB97], ceci est théoriquement possible ; cependant, les conditions d'échantillonnage imposées sont alors totalement irréalistes (surtout la condition d'uniformité[16]). En pratique, l'application de cette idée globalement sur les données n'est pas probante pour la plupart des échantillons, les facettes à conserver et certaines autres grossièrement mauvaises s'entremêlent. En fait, nous allons utiliser cette classification par le rayon du plus petit cercle circonscrit vide de points, mais *localement*, pour l'ensemble des facettes adjacentes à une arête e, dont les deux extrémités sont voisines sur l'objet réel. Ce critère n'est pas suffisant, et on devra le conjuguer à des contraintes topologiques, pour aboutir à notre reconstruction par *déploiement de surface*.

Notre méthode de reconstruction s'apparente ainsi à celle développée par S. PETITJEAN et E. BOYER dans leur article [PB01]. En effet, dans les deux cas, il s'agit de déployer une surface triangle après triangle en suivant un certain critère permettant de choisir correctement ces triangles et d'étendre ainsi cette surface idéalement sur l'ensemble de l'échantillon pour obtenir une reconstruction. Cependant, dans la mise en pratique de cette idée, notre approche se différencie sensiblement de celle adoptée par les auteurs de [PB01]. Contrairement à eux, nous avons décidé de contrôler précisément le déploiement de la surface en imposant des contraintes topologiques pour l'ajout des triangles. De plus, notre critère de sélection des triangles candidats pour l'extension de la surface est un peu plus général, dans le sens où il reste valide dans des conditions d'échantillonnage où le leur est inopérant. Par ailleurs, nous avons introduit une notion de *prudence* au cours du déroulement de l'algorithme. Le déploiement de notre surface s'effectue toujours grâce au triangle qui possède les meilleures caractéristiques pour prolonger notre reconstruction.

Notre méthode sera évidemment à même de traiter des échantillons de points non-uniformes. Mais le principal sujet d'intérêt sera de ne présupposer qu'un minimum de choses sur l'objet à reconstruire. Ce minimum, pour des échantillons de points mesurés sur des objets *réels*, nous a semblé être l'*orientabilité* de la surface reconstruite. Pour des échantillons synthétiques ou des objets mathématiques (bande de MOEBIUS par exemple), on pourra évidemment aussi ne pas considérer l'orientabilité comme impérative. Cette surface pourra en outre, suivant les cas, être fermée, à bord ou encore à trous. On espère ainsi être capable de traiter quasiment tous les types d'échantillons issus d'objets réels, quelque soit leur type de topologie, qu'ils soient volumiques (surfaces orientables fermées) ou surfaciques.

[16] Une version pondérée des Alpha-formes existe pour répondre au cas d'échantillons non uniformes, cependant le problème est alors d'évaluer les poids à affecter à chaque point (cf. 2.4.8), problème non traité jusqu'ici...

§3.2 Notion de bon échantillon de points

3.2.1 Définition de l'axe médian d'un objet

Comme nous l'avons vu, l'idée même d'une méthode de reconstruction à caractère général ne peut être dissociée de certaines conditions sur les échantillons de points qu'elle aura à traiter. Les conditions pour un échantillonnage satisfaisant ont déjà été étudiées. La formulation de telles conditions, établie par N. Amenta *et al.* [ABE98, ACDL00], peut, aujourd'hui, être considérée comme classique. Nous adopterons donc ici ces notions qui permettent d'apprécier la qualité des échantillons de points. Pour cela, il faut établir quelques définitions qui éclaireront le problème. Tout d'abord, nous allons préciser, ce qui, en quelque sorte, constitue le *dual* de la surface délimitant l'objet à reconstruire et qu'on nommera, ensuite, l'*axe médian* de l'objet.

Définition 3.1 *Soit $\mathcal{V}$ une surface fermée, une boule est dite* maximale *pour $\mathcal{V}$, si elle satisfait les assertions (i) et (ii) :*

(i) son intérieur n'intersecte pas $\mathcal{V}$,

(ii) elle ne peut pas être englobée par une boule plus grande satisfaisant l'assertion (i).

Notons qu'il y a deux boules maximales passant par un point de $\mathcal{V}$, si l'on accepte des boules de rayon infini. Pour une compréhension qualitative des idées que l'on développera par la suite, il est important de remarquer que ces boules maximales sont, d'une certaine manière, les sphères vides de Delaunay dans un cas continu.

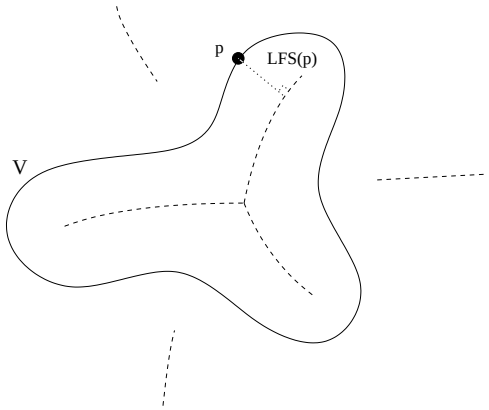


Fig. 3.2 – Illustration : l'axe médian.

Définition 3.2 *L'*axe médian*, ou* squelette généralisé *de $\mathcal{V}$, est le lieu des centres de toutes les boules maximales pour $\mathcal{V}$.*

De par la remarque faite précédemment sur les sphères maximales, on peut aussi considérer l'axe médian comme la version continue du diagramme de Voronoi (figure 3.2).

3.2.2 Définition d'un bon échantillon

Grâce à la notion d'axe médian d'une surface fermée $\mathcal{V}$, on peut maintenant formaliser certaines conditions essentielles pour espérer fournir une reconstruction acceptable. Commençons

par introduire une quantité mesurant la densité d'échantillonnage requise en un point donné de $\mathcal{V}$.

Définition 3.3 *Soit $\mathcal{V}$ une surface fermée et p un point de $\mathcal{V}$, on désignera par* LFS(p) (Local Feature Size) *la distance euclidienne de p à l'axe médian de $\mathcal{V}$.*

En effet, on peut déjà remarquer deux aspects de cette grandeur LFS, elle contrôle, non seulement, l'épaisseur de l'objet, mais aussi la courbure, au point considéré. Le fait qu'il faille plus de points aux endroits de forte courbure pour modéliser un objet est assez intuitif[17]. Par contre, la notion d'épaisseur de l'objet est propre au problème de la reconstruction. Pour saisir l'importance de l'épaisseur, on peut se référer une nouvelle fois à la figure 3.1 ; on remarquera alors que c'est un déficit d'échantillonnage par rapport à l'épaisseur de l'objet qui crée une ambiguïté.

Définition 3.4 *On nommera ε-échantillon d'une surface fermée $\mathcal{V}$, un échantillon de points $\mathcal{S} \subset \mathcal{V}$, tel que :*

$$\forall p \in \mathcal{V}, \exists q \in \mathcal{S} \quad \mathrm{d}(p,q) \leq \varepsilon \mathrm{LFS}(p)$$

Nous avons maintenant défini, du moins théoriquement, une mesure de qualité pour un échantillon. Cette mesure n'est pas uniforme, elle s'adapte à l'objet. Cependant, pour des applications pratiques, il semble difficile de demander, aux différents procédés de mesure de fournir des ε-échantillons à ε fixé. L'axe médian d'un objet réel n'est, en général, ni connu, ni facile à déterminer. Ces définitions sont donc difficilement exploitables. L'espoir en pratique de n'avoir à traiter que des ε-échantillons, surtout avec ε petit, semble donc bien mince, d'autant plus que certaines surfaces possèdent des singularités, telles des arêtes vives par exemple, et ne peuvent être ε-échantillonnées, quelque soit ε.

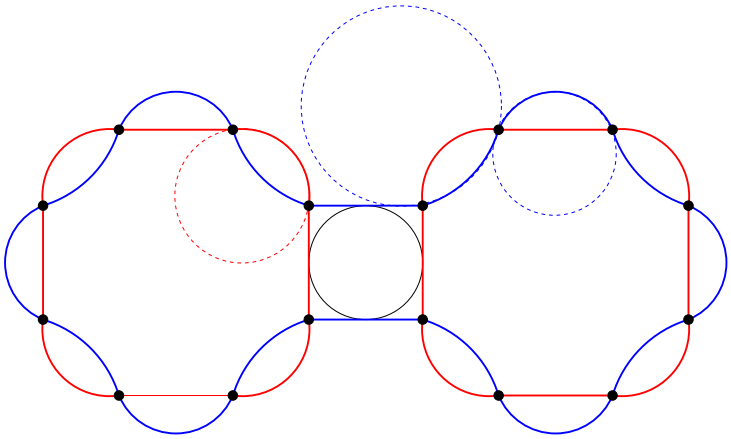


FIG. 3.3 – Illustration : exemple d'un même ε-échantillon, avec $\varepsilon = 1$, pour deux courbes distinctes.

Malgré tout, cette notion d'ε-échantillon définit un cadre théorique utile à la compréhension des limites intrinsèques des méthodes de reconstruction généralistes. En effet, comme le montre la figure 3.3 [ABE98], aucune méthode *généraliste* ne peut espérer répondre pour certains objets à partir d'un échantillon qui n'est pas un ε-échantillon, avec $\varepsilon \geq 1$. De plus, il existe même des

[17] On peut, par exemple, penser aux applications de CAO (Conception Assistée par Ordinateur).

méthodes dans le plan [ABE98] et dans l'espace [ACDL00] qui fournissent une reconstruction homéomorphe à l'objet d'origine pour un ε-échantillon avec ε inférieur à certaines constantes connues (mais évidemment petites). On peut ainsi considérer que cette notion d'ε-échantillons représente correctement ce que l'on peut espérer d'un échantillon pour un objet lisse, dans un cas idéal. Définissons donc la notion de «bon» échantillon, qui peut trouver une réalité pratique dans le cas d'échantillons «suffisamment denses».

Définition 3.5 *On appellera donc «bon» échantillon, un ε-échantillon avec $\varepsilon < 1$.*

3.2.3 *Au sujet des interpolants réguliers*

Une approche différente

Si l'on adopte une attitude plus pragmatique, une première remarque de bon sens est que ces considérations sur l'axe médian de l'objet à reconstruire sont certes intéressantes, mais que les garanties qu'elles prétendent apporter ne sont que théoriques et finalement assez éloignées des enjeux réels. En effet, si l'on cherche à reconstruire un objet, c'est certainement que l'on ne connaît pas sa géométrie[18] et encore moins son axe médian. Ainsi, prouver qu'une méthode reconstruit correctement un objet ε-échantillonné pour une certaine valeur de ε, montre que la méthode *peut* aboutir dans un cas idéal, mais qu'en est-il sur un échantillon réel qui n'est peut-être pas partout si «bon» que ça ? Concrètement, ces considérations n'aident en rien à la résolution du problème tel qu'il se pose : on dispose d'un objet (qui, éventuellement, peut posséder des arêtes vives[19]), que l'on a numérisé au mieux et on cherche à le reconstruire.

Existe-t-il alors d'autres façons d'évaluer la qualité d'un échantillon ? Une idée alternative à la notion d'ε-échantillon, développée par S. Petitjean et E. Boyer [PB01], peut être formulée ainsi : un échantillon $\mathcal{S}$ est de bonne qualité, s'il permet une reconstruction pour laquelle on peut considérer qu'il est un «bon» échantillon. Ce point de vue ne permet pas de garantir un lien quelconque avec l'objet d'origine certes, mais il est peut-être plus intéressant en pratique. La démarche, qu'il induit, diffère de celle que l'on peut avoir si on prétend disposer d'un ε-échantillon. Étant donné un échantillon de points, il s'agit, en effet, d'exhiber une reconstruction pour laquelle ces points constitueront un «bon» échantillon, l'idée étant d'apporter la réponse la plus cohérente possible au regard des données que l'on nous a fournies.

Une version discrète de l'axe médian

Plus précisément, reprenons, dans $\mathbb{R}^3$, les définitions énoncées dans [PB01]. On nommera *interpolant* une surface triangulée fermée qui passe par tous les points de l'échantillon. Considérons $\mathcal{O}$ un interpolant pour un échantillon de points $\mathcal{S}$. On désignera comme *adjacents* des sommets qui forment un simplexe (*i.e.* une arête ou un triangle) de $\mathcal{O}$.
On appellera *granularité locale* de $\mathcal{O}$ en p, notée $g_{\mathcal{O}}(p)$, le rayon de la plus petite sphère contenant tous les sommets de $\mathcal{O}$ adjacents à p. On peut se représenter la notion de granularité locale en dimension 2 sur la figure 3.4 comme le rayon du disque centré en p contenant les deux voisins de p sur l'interpolant $\mathcal{O}$.

Par ailleurs, on cherche à définir une version discrète pour l'axe médian de l'objet $\mathcal{O}$. Pour cela, reprenons la définition d'une *boule maximale* (cf. 3.1) appliquée à l'échantillon de points $\mathcal{S}$.

[18] Ou alors, on aime se poser des questions inutiles...
[19] Adieu alors, l'espoir de disposer d'un ε-échantillon !

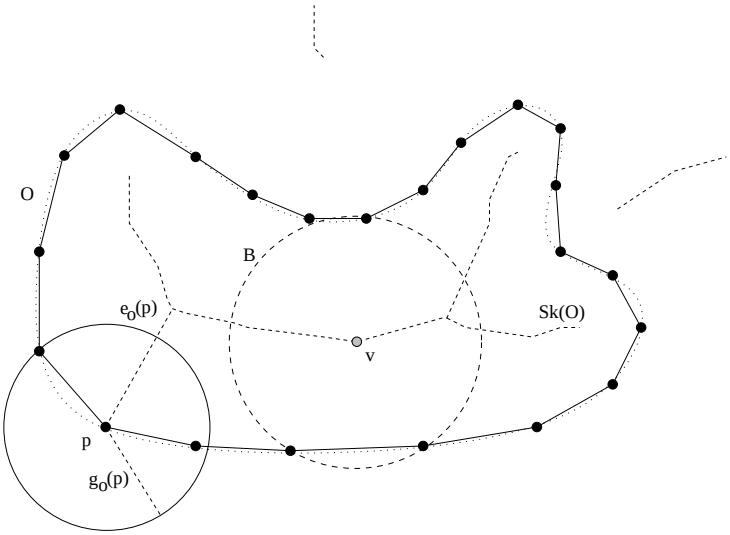


FIG. 3.4 – Illustration : version discrète de l'axe médian pour une courbe polygonale, granularité et épaisseur locale.

Il suffit pour cela de reformuler l'assertion (i) de la définition 3.1 pour le cas discret : une boule satisfait (i′) si elle vérifie la propriété de la sphère vide (cf 1.1). On dira de plus qu'une boule maximale est *libre* vis à vis de $\mathcal{O}$, si l'ensemble des points sur sa frontière ne constituent pas les sommets d'un même simplexe de $\mathcal{O}$. Dans le plan, en se référant à la figure 3.4, on notera, par exemple, que le cercle en pointillé de centre v est un cercle maximal libre vis à vis du contour polygonal $\mathcal{O}$ représentant l'interpolant.
On peut alors énoncer une définition discrète de l'axe médian :

Définition 3.6 *L'*axe médian *d'un interpolant $\mathcal{O}$ est le lieu des sphères maximales libres vis à vis de $\mathcal{O}$.*

Exprimé différemment, il s'agit de sélectionner parmi les faces et les arêtes du diagramme de VORONOI de $\mathcal{S}$ celles dont le dual n'apparait pas sur la surface triangulée $\mathcal{O}$. Sur la figure 3.4, on peut donc observer le squelette discret de l'interpolant $\mathcal{O}$ représentée par la courbe polygonale $Sk(\mathcal{O})$ en pointillé.
Définissons maintenant l'équivalent de LFS dans le cas discret, que S. PETITJEAN et E. BOYER nomment *épaisseur locale.*

Définition 3.7 *On nomme* épaisseur locale *de l'objet $\mathcal{O}$ en un point p une quantité notée $e_{\mathcal{O}}(p)$ qui est égale à :*

- *zéro si p ne contribue pas à l'axe médian dans chaque composante de $\mathbb{R}^3 \setminus \mathcal{O}$ ou si $\mathcal{O}$ intersecte l'une de ces contributions ;*
- *la distance entre P et l'axe médian de $\mathcal{O}$ dans les autres cas.*

Toujours en observant la figure 3.4, l'épaisseur locale au point p correspond alors à la distance de p au squelette $Sk(\mathcal{O})$.

Venons en, maintenant, à la définition d'un *interpolant régulier* pour un échantillon donné. Il s'agit d'établir si une reconstruction éventuelle est plausible vis à vis de l'échantillon sur lequel elle s'appuie. On cherche donc à vérifier si l'échantillon est un «bon» échantillon de l'interpolant.

Définition 3.8 *On dit d'un interpolant $\mathcal{O}$ qu'il est un* interpolant régulier *de l'échantillon $\mathcal{S}$, si la* granularité locale *de $\mathcal{O}$ en tout point de $\mathcal{S}$ est strictement inférieure à l'épaisseur locale de $\mathcal{O}$ en ce point.*

Enfin, on peut décrire l'équivalent de ce que l'on nomme «bon» échantillon d'une surface continue, en disant que l'échantillon $\mathcal{S}$ est de *«bonne qualité»* si $\mathcal{S}$ possède un interpolant régulier $\mathcal{O}$; on dira alors de $\mathcal{S}$ qu'il est *régulier*.

§3.3 Déploiement d'une surface sur un nuage de points

3.3.1 Propos introductifs

Pour pouvoir mettre en place les différents éléments de notre méthode, on considérera, le cas idéal d'une surface lisse $\mathcal{V}$ sur laquelle on aura numérisé un «bon» échantillon de points $\mathcal{S} \subset \mathcal{V}$. Notons $\mathfrak{D}$, la triangulation de DELAUNAY de $\mathcal{S}$. Nous détaillerons le cas de la dimension 3, sachant que la méthode proposée s'applique évidemment en dimension 2 avec de notables simplifications d'ailleurs. Le cas des dimensions supérieures est, quant à lui, beaucoup plus complexe. Les configurations géométriques des simplexes de DELAUNAY entre eux sont alors plus nombreuses et leur étude, sur laquelle notre méthode repose en partie, est évidemment plus ardue. Pour une facette f de $\mathfrak{D}$, on notera $\mathfrak{V}_f$ son arête duale.

La méthode décrite, a pour principe le déploiement d'une surface sur notre échantillon de points à partir d'un *germe*. Deux aspects peuvent être dissociés dans l'exposé de la méthode. Le premier aspect est topologique, il s'agit essentiellement d'interdire la création de singularités, de gérer les cas de recollement, de traiter les cas terminaux autour des arêtes vives de l'objet de départ le cas échéant. Il conviendra, en outre, de guider l'extension de notre reconstruction pour préserver, en permanence, l'orientabilité de notre surface dans le cas d'échantillons issus d'objets réels. L'autre aspect est la sélection des facettes candidates pour étendre notre surface et l'attribution d'une note, estimant la *pertinence* d'une facette à faire partie de notre reconstruction. On cherchera, ainsi, à déployer notre surface de la manière la plus *prudente* possible, par adjonction successive des facettes les plus *pertinentes*. Cette propagation prudente possède l'avantage, grâce à la surface en partie reconstruite, de nous aider à faire les bons choix dans les cas limites, dont le traitement est repoussé à la fin de l'exécution de l'algorithme. Cette notion de *prudence* est capitale : comme la méthode ne remet jamais en cause ce qui a déjà été reconstruit, il faut absolument éviter toute erreur. Ce constat peut être interprété de manière dialectique, les erreurs commises sont irrécupérables, mais tout ce qui est reconstruit correctement nous évite des écueils futurs.

3.3.2 Contraintes topologiques à l'extension d'une surface

Introduction

Le premier enjeu consiste à établir des règles afin de déployer notre surface sans créer aucune singularité. Dans le cas d'échantillons de points issus d'objets réels, on voudra, en outre, préserver au cours du processus, l'*orientabilité* de notre surface de reconstruction. Pour traiter des échantillons modélisant des objets mathématiques non orientables, de type bande de MOEBIUS, bouteille de KLEIN, on s'affranchira évidemment de cette contrainte supplémentaire. La volonté de préserver ces propriétés minimales au cours du déploiement de notre surface impose, malgré tout, des contraintes topologiques assez fortes pour le choix des facettes candidates à l'extension. Dans cette partie, on peut considérer que le déploiement de notre surface s'effectue grâce à un ensemble de facettes d'une triangulation quelconque, que l'on notera $\mathfrak{T}$. Le fait de disposer d'une triangulation de DELAUNAY n'intervient nullement ici. Notre surface initiale sera une facette dont le bord sera arbitrairement orienté. On reviendra, plus tard, sur la manière de choisir cette facette initiale.

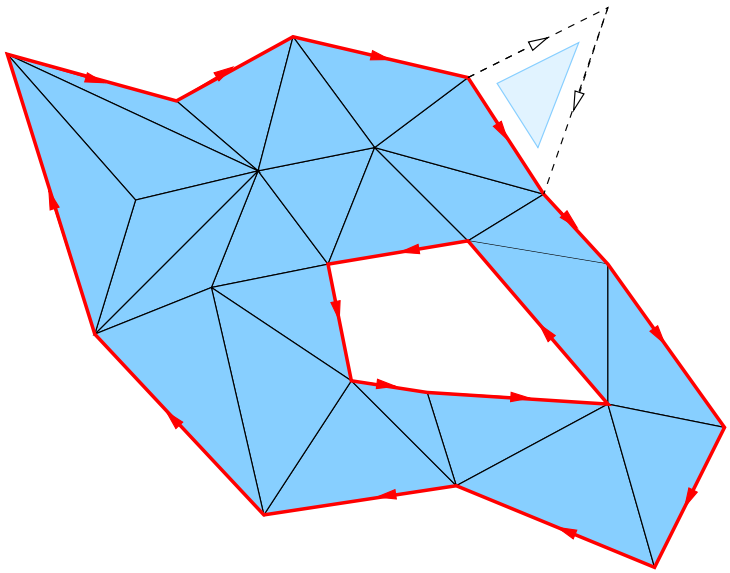


FIG. 3.5 – Illustration : pour des objets réels, déployer une surface orientable et maintenir son bord orienté.

Notations

À un instant donné au cours du déploiement, on suppose disposer d'une surface (orientable si souhaité), composée de facettes de $\mathfrak{T}$, dont on maintient le bord (orienté pour une surface orientable), noté $\mathcal{B}$. De plus, cette surface n'a aucune singularité. Les points qui la composent sont alors de deux types :

* soit le point considéré est *sur le bord* de la surface, c'est à dire sur le bord du disque topologique formé par les facettes de la surface adjacentes à ce point ;
* soit le point est *intérieur* à la surface et dans ce cas les facettes de la surface qui lui sont adjacentes forment un disque topologique autour de lui.

On dira, de plus, d'une arête de $\mathfrak{T}$ qu'elle est *intérieure* si elle appartient à deux facettes déjà sélectionnées pour notre reconstruction et qu'elle ne fait plus partie du bord $\mathcal{B}$ (figure 3.6).

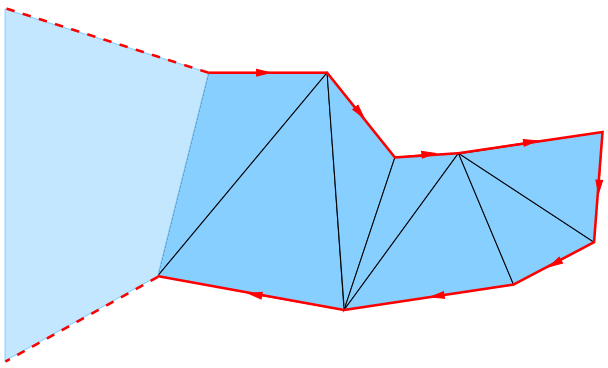
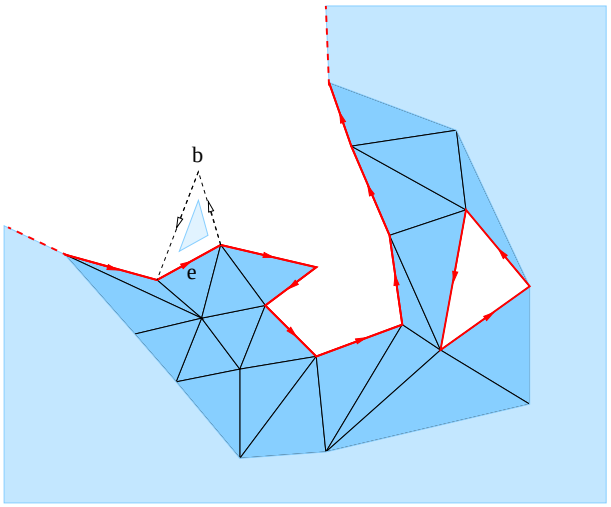


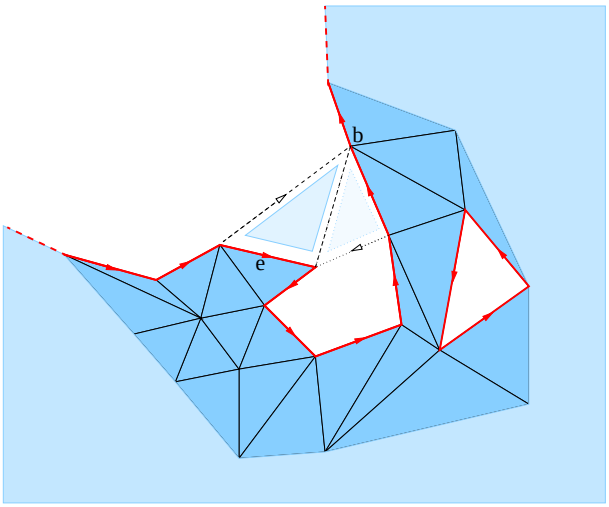
FIG. 3.6 – Illustration : cas d'une bande de triangles, d'où l'importance de savoir reconnaître les arêtes intérieures.

Énoncé des contraintes à l'extension

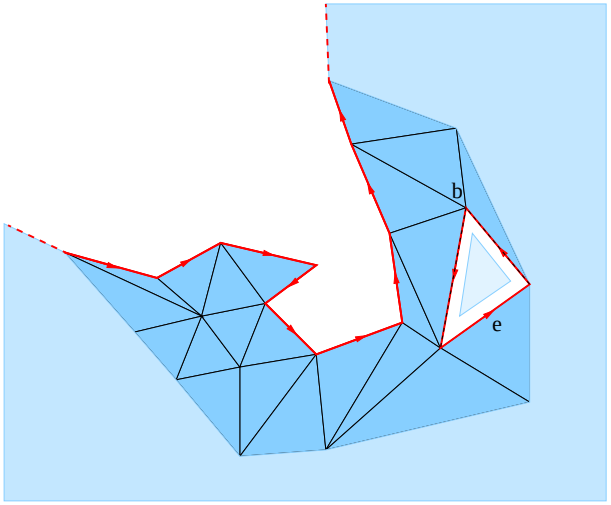
Détaillons maintenant les opérations autorisées permettant de maintenir une surface valide (*i.e.* sans singularités et orientable si on le souhaite) pendant son extension. Comme cette extension se fait triangle par triangle, il suffit d'étudier les différentes possibilités pour l'ajout d'un triangle à une surface triangulée en s'interdisant de créer des singularités. Pour cela, considérons une arête e du bord $\mathcal{B}$ à partir de laquelle, on veut ajouter un triangle à notre surface. Les ajouts valides de facettes à une surface correspondent, en fait, à des opérations topologiques bien précises : la *propagation*, la *fermeture* et le *recollement*. Pour les décrire, notons b le troisième sommet d'une facette candidate autour l'arête e.



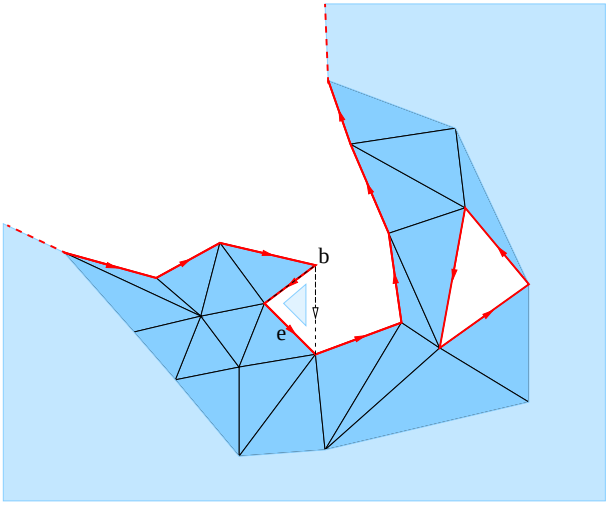
cas de propagation.



cas de recollement.



cas de fermeture (trou).



cas de fermeture (oreille).

FIG. 3.7 – Illustration : les différents cas autorisés pour l'extension de surface.

La propagation correspond au cas où b est un point de $\mathfrak{T}$ qui n'appartient pas encore à notre surface. C'est à dire l'ajout d'un nouveau sommet (ni intérieur, ni sur le bord) en même temps que d'une nouvelle facette à notre surface. Cette opération ne change pas le type d'homotopie.

La fermeture est l'opération correspondant au cas où le point b est successeur *ou* prédécesseur de l'arête e sur le bord orienté. Si b est successeur *et* prédécesseur de e, l'opération est, en fait, la fermeture du trou correspondant exactement à la facette candidate. Si le *ou* est exclusif, alors il s'agit du rajout d'une «oreille» ; notons que dans ce cas, on ne modifie pas l'homotopie de notre reconstruction. Le cas de fermeture étend la surface sans rajouter de point.

Le recollement est un cas plus difficile à traiter, celui où le point b est sur le bord de la surface, sans être un cas de fermeture. Il faut alors vérifier plusieurs choses, car l'ajout de cette facette modifie le type d'homotopie de notre reconstruction. Tout d'abord, pour éviter la création d'une singularité grossière (figure 3.6), la facette ne doit pas posséder d'*arêtes intérieures* à la surface[20].

De plus, il faut impérativement trouver, pour cette facette, une facette *jumelle*, candidate pour l'une des arêtes de $\mathcal{B}$ adjacentes à b, qui viendra s'appuyer sur notre facette en un cas de fermeture, afin de ne pas créer de pincement du bord de notre surface. D'autre part, si on veut préserver l'orientabilité de la surface, il faut prendre garde aux orientations des deux facettes jumelles qui se doivent d'être *opposées* sur leur arête commune.

Pour chaque arête e du bord $\mathcal{B}$ on peut ainsi déterminer un ensemble $\mathcal{E}_e$, éventuellement vide, constitué de facettes adjacentes à e, qui proposent une extension de notre surface en une surface sans singularités et, si on le souhaite, toujours orientable. Les facettes de $\mathcal{E}_e$, ou facettes *candidates à l'extension* à partir de e, correspondent chacune à un des cas autorisés pour l'extension : propagation, fermeture ou recollement.

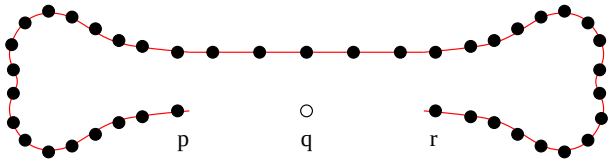


FIG. 3.8 – Illustration : un cas difficile *a priori* et accessible quand on connait une portion de la reconstruction.

Notons, que telles qu'elles sont définies, ces opérations interdisent au point b d'appartenir à l'intérieur de la surface. Cette propriété a une conséquence importante : ce qui est déjà reconstruit n'est plus accessible par la suite. En adoptant une loi de propagation *prudente*, plus on progresse dans la reconstruction, moins les possibilités d'erreur sont nombreuses : on peut alors résoudre des cas insolubles (ou très ambigus) *a priori* (figure 3.8) grâce à la portion de la surface déjà reconstruite. On remarquera, en outre, que les deux premières opérations, la propagation et la fermeture, préservent naturellement l'orientabilité de la surface, alors que pour la troisième, le recollement, on se doit d'y veiller explicitement.

[20] C'est le seul cas où il est nécessaire de le vérifier explicitement.

3.3.3 *Présentation de nos critères de sélection dans Delaunay*

Introduction aux idées sous-jacentes

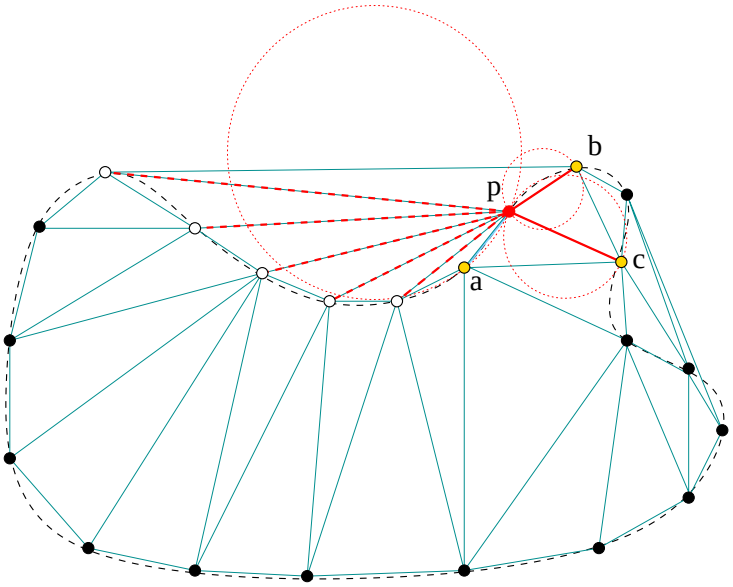


FIG. 3.9 – Illustration du cas 2D : répartition des arêtes et de leur plus petit cercle vide autour du point p.

L'idée principale qui sous-tend notre sélection de facettes dans DELAUNAY pour former la surface reconstruite, est de distinguer précisément les facettes de la triangulation traversées par l'axe médian des facettes dont les trois sommets sont voisins sur notre surface d'origine et qui pourraient donc, éventuellement, faire partie de notre reconstruction. On considère évidemment disposer d'un «bon» échantillon $\mathcal{S}$ (*i.e.* un ε-échantillon, avec $\varepsilon < 1$), pour pouvoir déterminer un critère distinctif qui fonctionne, au moins, dans le cas idéal. De plus, la triangulation utilisée ici sera la triangulation de DELAUNAY de l'échantillon $\mathcal{S}$, du fait de ses bonnes propriétés pour le problème de la reconstruction.

Soit f une facette déjà sélectionnée pour faire partie de notre reconstruction, on considère une de ses arêtes e autour de laquelle, il s'agit de trouver la facette qui sera la meilleure candidate pour continuer notre déploiement de surface. On considère l'ensemble $\mathcal{E}_e$ des facettes satisfaisant les contraintes topologiques (cf. 3.3.2), *i.e.* ensemble des facettes candidates pour l'extension de f à partir de e.

En reprenant l'idée maîtresse utilisée par les Alpha-formes mais cette fois localement autour de l'arête e, nous emploierons *le rayon de la plus petite sphère circonscrite vérifiant la propriété de la sphère vide* (propriété 1.1) comme grandeur sélective pour distinguer les facettes de $\mathcal{E}_e$. Par ailleurs, on pourra se reporter à la section 3.5.1, où sont détaillés des résultats théoriques qui légitiment *a posteriori* l'usage de ce critère de sélection.

Définition 3.9

* *Par abus de langage, on appellera* rayon de la facette g *et on notera* r_g*, le rayon de la plus petite sphère circonscrite à la facette* g*, vide de points de* $\mathcal{S}$[21].
* *On appelle* facette candidate à l'extension de f à partir de e*, la facette* g *possédant le plus petit* rayon r_g *parmi les facettes de* $\mathcal{E}_e$.

Tentons de donner une compréhension qualitative des raisons de notre choix. Pour cela plaçons-nous dans le cas un peu plus simple de la dimension 2. Il s'agit donc de reconstruire une courbe à partir d'un échantillon de points pris dans le plan. On déploie donc ici des arêtes uniquement par extension et fermeture de trou (les autres cas n'existant pas), le bord de la courbe est alors constitué des deux extrémités, notées p et q. Alors si on nomme *rayon* d'une arête g, le rayon de son plus petit cercle circonscrit vide de points, notre critère de sélection devient : on choisit comme candidate à l'extension d'une arête f autour de p l'arête possédant le plus petit rayon dans $\mathcal{E}_e$. Si on se réfère à la figure 3.9, on s'aperçoit que pour le point p en rouge, si l'arête $f = [a, p]$ est déjà sélectionnée, alors notre critère de sélection suffit à choisir $g = [p, b]$ pour continuer à déployer notre courbe. En effet, toutes les autres arêtes adjacentes au point p ont des rayons beaucoup plus grands. Le seul du même ordre de grandeur serait celui de l'arête $[p, c]$, mais lui aussi est plus grand puisque l'on suppose disposer d'un «bon» échantillon de points ($\varepsilon < 1$).

On peut remarquer que ce *rayon* r_g pour une facette g est aussi égal à la distance de l'une des extrémités de e à l'arête duale $\mathfrak{V}_g$. Ainsi, la facette de rayon minimum est la facette dont l'arête duale est la plus proche de e. Par ailleurs, on pourra, par exemple, choisir comme facette initiale, que l'on nomme le *germe*, la facette de $\mathfrak{D}$ ayant le plus petit rayon.

Le cas du sliver

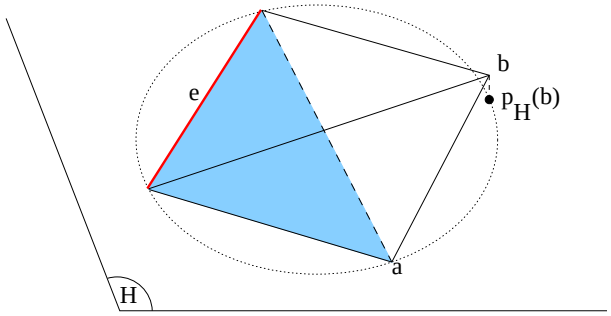


FIG. 3.10 – Illustration : cas du *sliver*.

Malheureusement, en dimension 3, la situation se complique, à cause d'une configuration spéciale, le *sliver*. En effet, il existe des tétraèdres aux sommets presque coplanaires, dont les quatre facettes ont un *rayon* du même ordre de grandeur. On pourra se référer à la figure 2.14 du chapitre 2 pour voir ces *slivers* grâce aux Alpha-formes, sur un échantillon de points pourtant considéré comme simple pour le problème de la reconstruction. En observant la figure 3.10, on peut alors s'apercevoir du problème que peuvent nous poser ces *slivers*. Supposons que l'on cherche un candidat pour l'extension de f, la facette en bleu, autour de l'arête e. Alors, la facette g dont le troisième sommet est b, quatrième sommet du tétraèdre *sliver*, possède un rayon de

[21] Attention ! Cette sphère peut ne pas être la sphère équatoriale à la facette g

l'ordre de celui de f. Cette facette g a donc des chances d'être choisie comme candidate pour l'extension de f si on se contente de prendre la facette de plus petit rayon. Il serait cependant tout à fait aberrant de la sélectionner pour notre reconstruction, l'angle qu'elle forme avec le plan de la facette f étant trop mauvais. Deux facettes adjacentes dans notre surface de reconstruction ne peuvent former un repli semblable, avec un angle proche de zéro. En effet, pour un «bon» échantillon, la densité des points est liée à la courbure.

Il est nécessaire de parvenir à repérer ces configurations pour les éliminer de $\mathcal{E}_e$ afin de déterminer une vraie facette candidate à l'extension de la surface. Pour ceci, on peut utiliser l'*angle du dièdre* entre les facettes f et g adjacentes à une même arête e. En effet, comme évoqué précédemment, une des caractéristiques de la situation problématique générée par ces *slivers* est que l'*angle du dièdre*, noté θ, entre f et g est proche de zéro. Ainsi, on peut exclure de $\mathcal{E}_e$ les facettes appartenant à un même tétraèdre que f qui forme un angle du dièdre $\theta < \alpha_{sliver}$, où α_{sliver} est un «petit» angle. En effet, les facettes exclues avec un tel angle, sont, en fait, totalement inutiles pour une reconstruction correcte, petit rayon ou pas.

Critère pour le choix d'une facette candidate

À partir de ces constatations, on peut formuler un critère de sélection efficace pour guider notre choix en dimension 3. Supposons être au cours de la reconstruction : on dispose alors de la partie de la surface déjà reconstruite et de son bord $\mathcal{B}$ (orienté pour obtenir une surface orientable). Il s'agit alors de trouver pour chaque arête e du bord $\mathcal{B}$, la facette voisine la plus plausible dans $\mathcal{E}_e$ pour étendre notre reconstruction.

Définition 3.10 *Soit $\mathcal{E}_e$, l'ensemble des facettes candidates pour prolonger f autour de l'arête e vérifiant l'une des conditions énoncées à la section 3.3.2 (propagation, fermeture ou recollement) :*

(i) On exclue de $\mathcal{E}_e$, si nécessaire, les facettes appartenant à un même tétraèdre que f formant un angle du dièdre *θ avec f, vérifiant $|\theta| < \alpha_{sliver}$*[22].

(ii) On choisit alors comme facette candidate pour l'extension *de f à partir de e, la facette de $\mathcal{E}_e$, possédant le plus petit rayon r.*

Note de pertinence associée aux facettes candidates

Pour établir une loi de propagation prudente, il faut maintenant être capable de comparer ces facettes candidates à l'extension. Il est important de déployer les facettes candidates une à une, en prenant toujours la plus *pertinente*. Il est, en effet, indispensable de progresser prudemment afin d'éviter de commettre des erreurs, qui ne pourraient être rattrapées par la suite. En contrepartie, cette progression prudente mêlée aux contraintes de déploiement (cf. 3.3.2), permet d'invalider certaines facettes candidates grâce à ce qui a déjà été reconstruit, pour en sélectionner de plus cohérentes par rapport aux choix déjà effectués. En d'autres termes, tout ce qui a été bien reconstruit ne peut être invalidé (algorithme glouton) et permet alors de guider la reconstruction par la suite. Notre intérêt, dans le but d'élaborer une notation prudente, est de reconstruire d'abord les zones où l'échantillon est plus dense que nécessaire.

Comment évaluer la *pertinence* d'une facette candidate pour notre reconstruction ? Pour cela, les deux aspects fondamentaux liés à la grandeur LFS rentrent en jeu, puisque l'on suppose disposer d'un «bon» échantillon. La densité de l'échantillon est alors dimensionnée par la grandeur LFS. Comme précédemment évoqué, LFS prend en compte, à la fois, la courbure et l'épaisseur

[22] Pour nous, en pratique $\alpha_{sliver} = \pi/6$

de l'objet au point d'évaluation. Or, la *courbure discrète* peut être appréciée localement, pendant l'examen des facettes candidates à l'extension à partir de l'arête e, par l'*angle du dièdre* θ. L'épaisseur est une notion globale, plus difficile à évaluer, au cours du déploiement. Cependant, nous avons maintenant quelques éléments pour nous permettre de traduire un aspect important dans le but d'agir avec prudence, à savoir reconstruire d'abord les zones plus denses en points que nécessaire.

Naturellement, on veut privilégier les facettes candidates à l'extension qui ont le plus petit rayon car elles se situent dans les zones échantillonnées le plus finement. Cependant, si ces facettes candidates imposent une courbure trop grande, elles nous entraînent dans des zones où la finesse de l'échantillon est vitale, voire sur des singularités où l'échantillon sera réellement mauvais. Les possibilités d'erreur sont alors grandes et bien qu'ayant un rayon petit, ces facettes sont, en fait, de mauvaise qualité. Notre intérêt se porte donc sur les zones où la courbure (donc les angles du dièdre θ) est faible et la densité des points est grande (donc des petits rayons pour les facettes). C'est cela qu'il faut prendre en compte pour être *prudent* : privilégier les facettes possédant un r petit et un θ proche de π .

Définition 3.11 *À chaque facette candidate pour l'extension de la surface, on attribue une* note de pertinence *:*

- *égale à son* rayon *si l'angle du dièdre θ qu'elle forme avec sa facette d'origine est compris entre $\pi - \beta_{good}$ et $\pi + \beta_{good}$*[23]*. Dans ce cas, on dira qu'elle possède une note de pertinence de* premier ordre *;*
- *égale à $|\pi - \theta|$, si tel n'est pas le cas. On dira alors qu'elle possède une note de pertinence de* de second ordre.

Les facettes possédant une note de pertinence de premier ordre seront toujours à traiter *avant* celles n'ayant qu'une note de second ordre. Pratiquement, on pourra exprimer ceci avec une unique notation. Celle que nous avons utilisée est la note de pertinence de la facette g, notée n_g, définie ainsi :

$$n_g = \begin{cases} -1/r_g & \text{si } \theta \in [\pi - \beta_{good}, \pi + \beta_{good}] \\ |\pi - \theta| & \text{sinon} \end{cases}$$

3.3.4 Description de l'algorithme de déploiement de surface

Fonction de calcul des candidats

Il s'agit, tout d'abord, de mettre en forme les différentes idées évoquées jusqu'ici et la manière dont elle interagissent entre elles. On va décrire, en premier, la fonction ComputeCandidate dont le but est, à tout instant au cours du déploiement de notre surface, de donner la meilleure facette candidate à l'extension d'une facette f autour d'une de ses arêtes e. On fournit, en même temps, la note de pertinence qui lui est associée. Notons, que ce meilleur candidat à l'extension n'a qu'une valeur *relative*. Il se peut, en effet, que le déploiement de notre surface invalide ce candidat (*i.e.* candidat n'appartenant plus à $\mathcal{E}_e$) avant de l'avoir considéré comme pertinent pour l'extension. Dans ce cas, on cherchera une nouvelle facette candidate à l'extension de f autour de e vérifiant les contraintes topologiques pour la portion de surface alors reconstruite.

[23] Pour nous, en pratique $\beta_{good} = \pi/6$.

Fonction : ComputeCandidate(f, e)

Require: f facette déjà sélectionnée et e arête examinée pour l'extension.
Ensure: f_c facette candidate à l'extension à partir de e et n_c sa note de pertinence.

```
soit f_circ un circulateur sur les facettes adjacentes à e ;
f_circ ⇐ f ;
r_c ⇐ +∞ ;
repeat
  sliverbool ⇐ (f_circ et f appartiennent au même tétraèdre) and (|(f, f_circ)^| < α_sliver)
  if (f_circ ∈ E_e) and (not sliverbool) then
    retrouver r_circ le rayon pré-calculé de f_circ
    if (r_circ < r_c) then
      f_c ⇐ f_circ ;
      r_c ⇐ r_circ ;
    end if
  end if
  f_c++ ;
until f_c ≠ f
if |π − (f, f_c)^| < β_good then
  n_c ⇐ −1/r_c ; {note de premier ordre}
else
  n_c ⇐ |π − (f, f_c)^| ; {note de second ordre}
end if
return (f_c, n_c) ;
```

Notons que les rayons des facettes sont considérés comme pré-calculés. En effet, cette opération aura été faite avant tout appel à la fonction ComputeCandidate afin d'élire une facette *germe* dans le corps de l'algorithme.

Fonction de déploiement

Maintenant que l'on sait se donner des candidats à l'extension, il faut encadrer le déploiement de la surface. Il s'agit, en effet, de vérifier qu'une facette ayant été une facette candidate à l'extension auparavant, vérifie toujours une des conditions de validité topologique pour l'extension de notre reconstruction. Dans le cas où on déploie effectivement notre surface, il faut alors veiller à la mise à jour de nos différentes structures : sélectionner la facette pour notre reconstruction, mettre à jour le bord $\mathcal{B}$ (orienté, si l'on souhaite une surface orientable), effectuer éventuellement certaines opérations exigées pour résoudre des singularités et calculer les nouveaux candidats à l'extension pour les nouvelles arêtes du bord.

Pour réaliser notre extension prudente, nous utiliserons une *queue de priorité*, $\mathfrak{P}$, dont les éléments sont les couples (f, e), où f est une facette candidate à l'extension à partir de e ; le tri par ordre croissant sur $\mathfrak{P}$ s'effectuera évidemment sur les notes de pertinence associées aux facettes candidates. Ainsi, on pourra choisir la facette la plus pertinente à chaque étape du déploiement de notre surface que l'on devra, alors, soumettre à la fonction ExtendBorder.

Fonction : ExtendBorder(f, e)

Require: f de pertinence n, proposée pour l'extension à partir de l'arête e.
Ensure: vérifie la validité topologique de f et étend la surface, si f est valide.

```
if f vérifie la condition de propagation then
    sélectionner f ;
    retirer e de B ;
    insérer e_cw et e_ccw, les deux autres arêtes de f, dans B ;
    (f_cw, n_cw) ⇐ ComputeCandidate(f, e_cw) ;
    (f_ccw, n_ccw) ⇐ ComputeCandidate(f, e_ccw) ;
    insérer (f_cw, e_cw) et (f_ccw, e_ccw) dans P selon leur pertinence ;
    return vrai ;
else if f vérifie la condition de fermeture (trou) then
    sélectionner f ;
    retirer les trois arêtes de f du bord B ;
    return vrai ;
else if f vérifie la condition de fermeture (oreille) then
    sélectionner f ;
    retirer de B les deux arêtes de f déjà présentes ;
    insérer e_c, la troisième arête de f, dans B ;
    (f_c, n_c) ⇐ ComputeCandidate(f, e_c) ;
    insérer (f_c, e_c) dans P selon sa pertinence ;
    return vrai ;
else if f vérifie la condition de recollement then
    for all e_bis ∈ {e_prec(b), e_succ(b)} l'ensemble des arêtes adjacentes au point b dans B do
        soit f_bis la facette déjà sélectionnée incidente à e_bis ;
        (f_bis, n_bis) ⇐ ComputeCandidate(f_bis, f_bis) ;
        if (f_bis est facette jumelle pour f) and (n_bis < n) then
            sélectionner f ;
            retirer e de B ;
            insérer e_cw et e_ccw, les deux autres arêtes de f, dans B ;
            (f_cw, n_cw) ⇐ ComputeCandidate(f, e_cw) ;
            (f_ccw, n_ccw) ⇐ ComputeCandidate(f, e_ccw) ;
            insérer (f_cw, e_cw) et (f_ccw, e_ccw) dans P selon leur pertinence ;
            ExtendBorder(f_bis, e_bis) ; {NB : par définition, f_bis vérifie la condition de fermeture}
            return vrai ;
        end if
    end for
end if
return faux ;
```

Algorithme principal

Décrivons maintenant le corps de l'algorithme de déploiement de surface. La méthode peut être séparée en deux phases distinctes. Une première étape de pré-traitement consiste à pré-calculer les *rayons* de toutes les facettes de la triangulation de Delaunay $\mathfrak{D}$. On en profite pour déterminer la facette initiale f_i qui constitue le *germe* de notre surface. La seconde étape effectue

le déploiement de notre surface de la manière la plus prudente possible. On cherche à étendre la surface, pas à pas, en prenant toujours, grâce à la queue de priorité $\mathfrak{P}$, la facette candidate la plus prudente. Si cette tentative échoue, on calcule alors un remplaçant pour ce candidat. Idéalement, dans le cas d'un «bon» échantillon, la méthode doit ainsi reconstruire une surface triangulée jusqu'à refermer entièrement les trous de la surface. On peut alors relancer l'algorithme sur les points restant, le cas échéant. On discutera des conditions d'arrêt à l'extension par la suite, pour le cas où l'on ne dispose pas d'un «bon» échantillon ou encore si l'on souhaite reconstruire des surfaces à bords.

Pour le calcul du *rayon* r_f associé à une facette f, on pourra se référer à la partie 2.4.3, où sont présentés les éléments nécessaires à ce calcul. On se placera évidemment dans le cas non pondéré (*i.e.* considérer les poids comme nuls). Il suffit, en effet, de calculer c_1 et c_2, les centres des sphères circonscrites aux deux tétraèdres incidents à f. Alors le segment $[c_1, c_2]$ est l'arête duale de f. Comme cette arête a pour droite support la médiatrice des trois sommets de f, le rayon de la plus petite sphère *vide* circonscrite à f, r_f, est égal à la distance entre le segment $[c_1, c_2]$ et l'un des sommets de f.

Algorithme de reconstruction par déploiement de surface

construire $\mathfrak{D}$, la triangulation de DELAUNAY des points de l'échantillon $\mathcal{S}$;
{ INITIALISATION : pré-calcul de rayons, initialisation du germe f_i}
for all f facette de $\mathfrak{D}$ **do**
  calculer et conserver r_f le rayon de f ;
  if $r_f < r_i$ **then**
    $f_i \Leftarrow f$;
  end if
end for
sélectionner la facette f_i ;
initialiser $\mathcal{B}$ avec les arêtes de f_i ;
for all e arête de $\mathcal{B}$ **do**
  $(f_c, n_c) \Leftarrow$ ComputeCandidate(f_i, e) ;
  insérer le couple (f_c, n_c) dans $\mathfrak{P}$ selon sa pertinence ;
end for
{ DÉPLOIEMENT DE LA SURFACE }
while $\mathfrak{P} \neq \emptyset$ **do**
  extraire le couple (f_e, e_e) le plus *pertinent* de $\mathfrak{P}$;
  if not ExtendBorder(f_e, e_e) **then**
    soit f_i la facette sélectionnée incidente à e_e ;
    $(f_{bis}, n_{bis}) \Leftarrow$ ComputeCandidate(f_i, e_e) ;
    if $f_{bis} \neq f_e$ **then**
      insérer le couple (f_{bis}, n_{bis}) dans $\mathfrak{P}$ selon sa pertinence ;
    end if
  end if
end while
return la surface triangulée constituée des facettes sélectionnées dans $\mathfrak{D}$

Par construction, l'ensemble des facettes sélectionnées forme ainsi une surface sans singularité, orientable si on l'a souhaité, et dont les bords, s'ils existent, sont connus.

§3.4 Heuristiques et implantation

3.4.1 Surfaces à bords, arêtes vives

Reconstruire des surfaces fermées

La méthode, telle qu'elle a été présentée, peut être considérée comme une version de base destinée à fournir une réponse satisfaisante à partir d'un «bon» échantillon pris sur un objet lisse. L'arrêt du déploiement survient idéalement quand il n'y a plus aucune arête sur le bord, ou alors faute de facettes candidates, aucune condition n'est posée sur la pertinence des derniers candidats à l'extension. Cela privilégie la reconstruction d'une surface fermée. Cette version de la méthode est donc particulièrement adaptée aux cas d'objets scannés en entier, dont la peau délimite un vrai volume. Pour traiter des types de topologies complexes, une fois une surface fermée entièrement reconstruite, on conserve la même triangulation de Delaunay afin de se laisser guider par ce qui a déjà été reconstruit et on relance l'algorithme de déploiement sur les points restant à partir d'un nouveau germe. À chaque fois, on reconstruit une surface délimitant une nouvelle composante connexe de l'objet.

Reconstruire des surfaces à bords

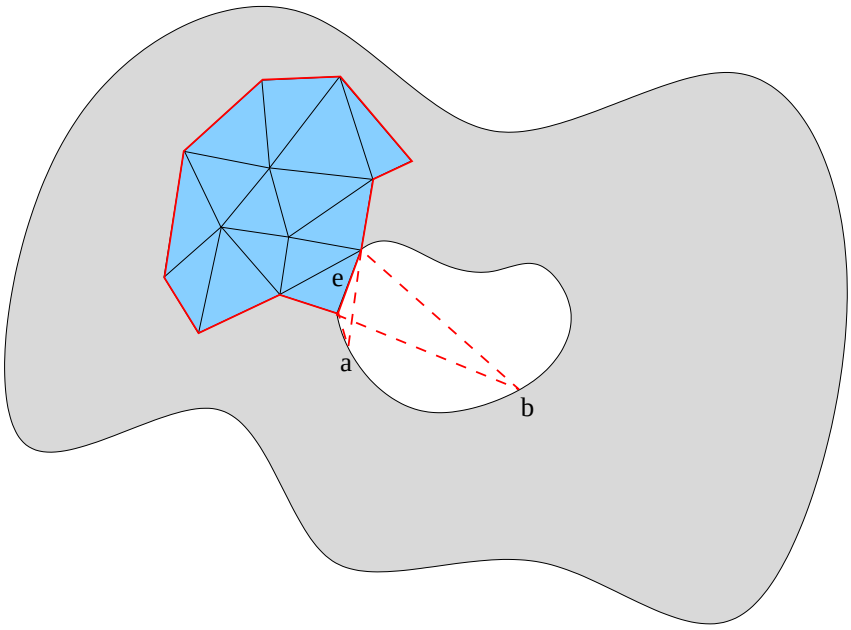


Fig. 3.11 – Illustration : repérer les bords d'une surface.

Comme annoncée, notre méthode se doit de pouvoir traiter le cas des surfaces à bord. Pour se représenter d'où proviennent ces données, on peut, par exemple, penser à des objets qui n'ont été numérisés que partiellement, ou encore que d'un côté. Très peu de méthodes sont à mêmes de traiter correctement ce genre de données. Il existe, en effet, des cas où il est nécessaire de manipuler réellement des surfaces pour espérer aboutir.

Pour pouvoir répondre à ce problème, il nous faut introduire des conditions d'arrêt sur la pertinence des candidats. En effet, il s'agit d'être capable d'interrompre le déploiement de la surface quand un bord de la surface originelle a été atteint. Ce problème n'est pas simple, car il requiert un échantillon de points qui, en plus de vérifier de bonnes conditions d'échantillonnage de type surfacique, possède aussi des bords correctement échantillonnés. Pour être plus précis, le problème de repérer les bords d'une surface consiste à reconstruire des courbes sur cette surface. On comprend alors que reconnaître des courbes sur une surface que l'on est en train de reconstruire est encore plus complexe.

Cependant, nous avons établi quelques heuristiques imposant des conditions d'arrêt, qui, en pratique, donnent des résultats tout à fait convaincants. Ces conditions reposent sur des considérations établies dans la situation suivante : s'il existe une courbe $\mathcal{C}$ «correctement» échantillonnée sur notre surface et que l'on examine une arête e qui se trouve précisément sur $\mathcal{C}$, les facettes de $\mathcal{E}_e$ possèdent-elles alors des propriétés caractéristiques ? On peut légitimement s'attendre à des configurations rares qui doivent éveiller un soupçon. Pour illustrer les propos qui vont suivre, on pourra se référer à la figure 3.11. Si l'arête e est sur un bord $\mathcal{C}$ de la surface réelle, on peut présumer que les points de l'échantillon «en face» de e sont loins. Les points de l'échantillon qui, par contre, sont proches de l'arête doivent être sur la courbe $\mathcal{C}$, puisqu'on suppose qu'elle est suffisamment échantillonnée. Pour formaliser cela, on a donc introduit deux limitations.

Tout d'abord, on a définit une limitation pour les facettes g de $\mathcal{E}_e$ sur les angles que le troisième sommet de g forme avec les extrémités de e. Ceci tente de retranscrire la volonté de ne pas continuer à étendre la surface si la facette candidate est susceptible de posséder elle aussi une arête du bord adjacente à e. Plus formellement, on peut donc énoncer une première contrainte heuristique au choix des candidats comme suit :

Heuristique 3.1 *Notons e_0 et e_1, les extrémités de l'arête e. On* exclut *de $\mathcal{E}_e$ les facettes dont le troisième sommet, disons le point a (figure 3.11), vérifie l'inégalité suivante, δ_{bord} étant une constante proche de* 0 *:*

$$\max(|(\widehat{\overrightarrow{e_0e_1}, \overrightarrow{e_0a}})|, |(\widehat{\overrightarrow{e_1e_0}, \overrightarrow{e_1a}})|) > \pi - \delta_{bord}$$

Cette condition correspondrait à des considérations sur la courbure discrète de la courbe $\mathcal{C}$ si l'on était dans le plan. D'une certaine manière on impose une courbure discrète maximale à la représentation polygonale de la courbe $\mathcal{C}$ au delà de laquelle on ne sait plus la repérer.

Le second critère peut être formulé ainsi, si la facette candidate à l'extension d'une facette f à partir de l'arête e possède un rayon très grand par rapport à celui de f, alors l'arête e est peut-être sur un bord. Cela signifie que pour un échantillon correct du bord $\mathcal{C}$, la facette de plus petit rayon possède son troisième sommet, disons le point b (figure 3.11), relativement loin de l'arête e. Cependant, si on se contente de cela, on touche alors à un point sensible pour la reconstruction, on requiert d'une certaine manière une certaine uniformité de l'échantillon dans sa globalité, y compris dans les zones où il n'y a pas de contours. Alors, on peut adoucir un peu nos exigences, en décidant de ne bloquer le déploiement autour d'une arête e que si sa facette candidate forme un mauvais angle du dièdre θ *et* que le troisième sommet de la facette est loin.

Autrement dit, on requiert une certaine uniformité de l'échantillon dans les zones courbées. Par contre, on ne saura toujours pas repérer les courbes tracées sur la surface quand celle-ci est peu courbée... On peut donc formuler un second critère limitant le déploiement de la surface ainsi :

Heuristique 3.2 *On décrète* invalide *chaque facette candidate g pour l'extension d'une facette f à partir d'une arête du bord e formant un angle du dièdre, θ, tel que $|\pi - \theta| > \beta_{good}$ (*i.e. *dont*

la note de pertinence est de second ordre) et qui vérifie $r_g/r_f > K$, où K est une constante supérieure à 1. L'arête e n'a alors plus aucune possibilité d'extension.

Rappelons que, suivant la définition 3.11, les notes de pertinence de second ordre sont attribuées aux facettes candidates formant un angle de pliure θ tel que $|\pi - \theta| > \beta_{good}$.

Le réglage de la constante K pourrait sembler problématique *a priori* car il dépend, en quelque sorte, de la géométrie des bords et des trous que l'on veut repérer. Il faut remarquer que choisir un K trop petit peut nous empêcher de prolonger la reconstruction dans des zones courbées où l'échantillon présente des irrégularités. Cette constante K n'est pas anodine ; elle interdit une évolution brutale du pas d'échantillonnage, puisque l'on présume que c'est ce qui suggère un bord. En pratique, le réglage de K ne s'avère pas vraiment sensible. *Grosso modo*, on distingue trois ordres de grandeur pour K, de l'ordre de l'unité pour des bords mal échantillonnés, de l'ordre de la centaine pour des échantillons normaux et enfin infini si on veut des surfaces fermées.

Traiter les arêtes vives

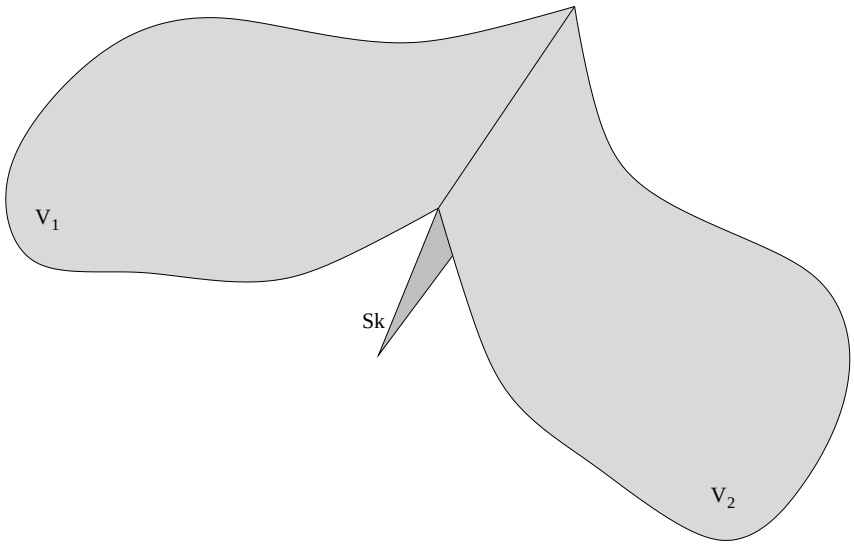


FIG. 3.12 – Illustration : une arête vive.

Les échantillons de points, provenant d'objets réels, possèdent souvent des zones où la qualité est très mauvaise dans des zones de très forte courbure. Ceci est le cas même sur des très gros modèles, puisqu'ils représentent, en général, des objets complexes très détaillés. Il suffit qu'un détail très fin ait été partiellement numérisé pour que les problèmes apparaissent. Bref, les échantillons dont on dispose sont rarement de «bons» échantillons sur l'ensemble de l'objet d'origine.

Ceci est d'autant plus vrai pour des objets non lisses, ce qui est le cas pour la majorité des objets réels. En effet, on peut songer aux objets manufacturés, par exemple, qui possèdent systématiquement des *arêtes vives* (*i.e.* les singularités de la surface de l'objet, cf. figure 3.12). Malheureusement pour nous, la présence de ces arêtes vives élimine tout espoir de disposer d'un «bon» échantillon, au moins autour de ces zones de singularités. En effet, l'axe médian de l'objet

rentre en contact avec les arêtes vives. Ainsi, sur une arête vive, la fonction LFS vaut exactement zéro et la densité requise pour un ε-échantillon est donc infinie, quelque soit ε.

Les conséquences par la plupart des méthodes sont souvent désastreuses. Le problème est en effet difficile. Autour des arêtes vives, l'échantillon ne vérifie plus aucune propriété susceptible d'indiquer que les points appartiennent à une surface. Cependant, notre méthode, grâce à la notion de *déploiement prudent*, repousse le traitement des cas problématiques, telles les arêtes vives ou plus généralement les zones ambiguës de l'échantillon, en fin de reconstruction. Ainsi, l'essentiel de la surface est déjà connu et une partie des problèmes ainsi réglée.

Dans cette dernière étape, les différents critères utilisés par notre méthode conduisent à considérer les points extrêmes, précisément sur l'arête vive, comme *marginaux* et à les laisser de côté. Ainsi, on reconstruit une surface lissée par rapport à l'originale, l'idée étant de réinsérer les points *marginaux* plus tard sur la surface, une fois la topologie correcte établie. Malheureusement les points laissés de coté qui appartiennent à l'arête vive posent de sérieux problèmes, on peut, en quelque sorte, les considérer comme des points «bruités» de l'échantillon d'une surface lisse. Ces points marginaux peuvent de temps à autre empêcher la résorption des bords dans les zones à problèmes. Ainsi, le déploiement de la surface achevé, on trouve parfois encore des trous localisés dans les zones où l'échantillon est de piètre qualité.

Nous proposons alors une étape de *post-traitement*, qui permet de tenter de refermer certains trous dans le but de terminer la reconstruction. L'idée est simple, elle consiste, tout d'abord, à retirer de la triangulation de Delaunay les points *marginaux* sur les présumées arêtes vives, suivant l'idée qu'ils nuisent, d'une certaine manière, à la qualité de l'échantillon. Pour ce faire, une fois le déploiement arrivé à son terme, on retire tous les points non sélectionnés (*i.e.* les points qui ne sont ni intérieurs ni sur le bord de notre surface). Ensuite on tente à nouveau de continuer le déploiement de la surface en recalculant des candidats pour chaque arête du bord.

Cette étape de post-traitement suffit dans la majorité des cas, cependant pour des échantillons possédant des zones vraiment problématiques, on peut encore espérer refermer notre surface en opérant un traitement heuristique plus brutal. On retire récursivement du bord les points que l'on estime empêcher la fermeture pour tenter à nouveau un redéploiement. Cette étape est empirique, mais donne de bons résultats en pratique. Elle se révèle plus lente que l'extension uniquement du fait du coût élevé en pratique du retrait d'un sommet de la triangulation de Delaunay.

3.4.2 *Détails d'implantation*

Une implantation de la méthode a été réalisée en C++ à l'aide de la libraire CGAL (cf. 2.1). La conception ainsi que la réalisation de notre algorithme doit d'ailleurs beaucoup à CGAL. En effet, c'est en grande partie grâce à la rapidité et la facilité avec laquelle nous avons pu tester nos idées, que cette méthode a vu le jour. Cependant, On peut noter que la méthode n'utilise réellement qu'une triangulation de Delaunay comme structure de Géométrie algorithmique de haut niveau. L'implantation utilisée pour la queue de priorité est celle de la STL. Le bord de l'objet est représenté par une liste chaînée de sommets.

En outre, l'implantation finale de notre méthode profite de nombreuses optimisations disponibles dans la librairie CGAL, ce qui lui assure de très bonnes performances, tant au niveau de la rapidité que de la robustesse. La triangulation de Delaunay utilisée est une triangulation de Delaunay hiérarchique robuste grâce à l'usage d'une arithmétique filtrée. En effet, le type de données à traiter pour la reconstruction se place dans les configurations les plus difficiles pour la triangulation de Delaunay, c'est à dire des échantillons de points volumineux pris sur la surface d'objets, ce qui génère énormément de configurations dégénérées. On notera, de plus, que CGAL a su profiter des problèmes que nous avons rencontrés avec cette application de reconstruction

et que nous avons soumis aux responsables des modules concernés. D'énormes progrès ont, par exemple, été réalisés par l'implantation de la triangulation de DELAUNAY en dimension 3 de CGAL, en partie grâce aux nouvelles exigences qui étaient les nôtres. On pourra apprécier les performances de la triangulation de DELAUNAY (CGAL version 2.4) que l'on a tenu à mettre en avant dans nos temps de calculs (tableau 3.1).

Pour les autres calculs, en particulier l'évaluation des rayons et des notes de pertinence sont réalisés avec un type de nombre simple. Ils ne présentent, en effet, pas un caractère critique ; les comparaisons entre les rayons, par exemple, sont franches, les bonnes facettes et les mauvaises bien séparées. Il serait, de plus, inefficace de penser résoudre le problème posé par les *slivers* par un calcul exact des rayons ; l'utilisation de l'angle du dièdre permet, elle, des comparaisons franches.

§3.5 Résultats

3.5.1 Conditions théoriques de succès

Pour un échantillon régulier

Remarque. Avant de continuer, il est important de souligner que la plus petite sphère circonscrite à une face f (*i.e.* celle dont le centre est dans le plan de la face) ou sphère de GABRIEL de la face f , que l'on notera $S_g(f)$, ne vérifie pas nécessairement la *propriété de la sphère vide*. Cette sphère $S_g(f)$ est, en général, différente de la plus petite sphère circonscrite à f, vide de points de l'échantillon, que nous avons utilisée jusqu'ici. Pour éviter toute ambiguïté, on notera $r_{S_g}(f)$ le rayon de la sphère de GABRIEL associée à la face f et on conservera la notation r_f pour le rayon de la plus petite sphère circonscrite à f vide de points de l'échantillon, comme auparavant.

Étant donné la parenté de notre méthode avec celle développée par S. PETITJEAN et E. BOYER dans leur article [PB01] revenons sur les résultats qu'ils ont énoncés pour la notion d'*interpolants réguliers* présentés dans l'introduction de ce chapitre (cf. 3.2.3). Tout d'abord, les auteurs montrent que tout interpolant régulier est composé de facettes de la triangulation de DELAUNAY. En outre, ils caractérisent les facettes de l'interpolant régulier $\mathcal{O}$ grâce à la propriété suivante :

Propriété 3.1

* *Si P_i, P_j et P_k sont trois sommets d'une même face d'un interpolant régulier $\mathcal{O}$ alors la plus petite sphère circonscrite à P_i, P_j et P_k, noté $S_{i,j,k}$ est vide d'autres points de $\mathcal{O}$. On dit aussi que la face est de* GABRIEL.
* *Considérons P_i et P_j, deux points adjacents sur un interpolant régulier $\mathcal{O}$. Alors, P_k et P_l, les sommets des deux faces de $\mathcal{O}$ incidentes à l'arête (P_i, P_j), sont les points qui vérifient les deux assertions :*
  - *(i) (P_i, P_j, P_k) et (P_i, P_j, P_l) sont des faces de* GABRIEL.
  - *(ii) Le rayon de $S_{i,j,k}$ et le rayon de $S_{i,j,l}$ sont les* deux plus petits *rayons associés aux sphères de* GABRIEL *des facettes incidentes à l'arête (P_i, P_j).*

Étudions le comportement de notre méthode dans le cas d'un échantillon régulier $\mathcal{S}_r$, *i.e.* qui dispose d'un interpolant régulier. Tout d'abord, on peut remarquer que la facette possédant la plus petite sphère circonscrite vide parmi toutes les facettes de la triangulation de DELAUNAY est nécessairement de GABRIEL. En effet, cette facette possède, en particulier, le plus petit rayon autour de chacune de ses arêtes. Or la convexité de la face de VORONOI associée à une arête e de la triangulation de DELAUNAY nous assure que la facette incidente à e qui possède le plus petit rayon est de GABRIEL. En effet, ce plus petit rayon est égal à la distance de l'arête e au polygone convexe qui délimite la face de VORONOI associée à e.

Nous pouvons maintenant montrer que notre algorithme reconstruit exactement un interpolant régulier pour l'échantillon $\mathcal{S}_r$. Plus précisément, formulons la propriété suivante :

Propriété 3.2 *Pour un échantillon de points régulier $\mathcal{S}_r$, la surface triangulée, résultat de notre algorithme de déploiement de surface, est un interpolant régulier de $\mathcal{S}_r$.*

Preuve. Raisonnons par récurrence pour montrer le maintien de la caractérisation formulée par la propriété 3.1 tout le long du déploiement de notre surface.

Initialisation. Notre facette *germe* h, selon la remarque précédente, est de GABRIEL. Par construction, elle possède aussi le plus petit rayon $r_{S_g}(h)$ pour chacune de ses arêtes.

Ajout d'une facette. Supposons être au cours du déploiement de notre surface qui possède jusqu'ici les caractéristiques de la propriété 3.1. On cherche alors à prolonger la surface à partir d'une facette (celle qui est la plus pertinente) autour d'une arête e sur le bord de la surface. La fonction ComputeCandidate a sélectionné la facette h qui possède le plus petit rayon r_h parmi les facettes disponibles, incidentes à e. Il suffit pour conclure de montrer que cette facette est nécessairement de GABRIEL. Raisonnons par l'absurde :

▷ *Considérons que h n'est pas de* GABRIEL. D'après la propriété 3.1, la facette h n'est donc pas une facette de $\mathcal{O}$, ainsi le centre de la plus petite sphère circonscrite à h vide de points est sur l'axe médian discret associé à $\mathcal{O}$. Cette sphère étant la plus petite sphère maximale, si on note a et b les extrémités de l'arête e, on a alors par définition de l'épaisseur locale $e_{\mathcal{O}}(a) \leq r_h$ et $e_{\mathcal{O}}(b) \leq r_h$. En outre, la propriété 3.1 nous assure de l'existence d'une facette de GABRIEL f de l'interpolant $\mathcal{O}$ autour de e pour continuer l'extension. De par le critère de choix de h, on a évidemment $r_{S_g}(f) \geq r_h$. Or par définition de la granularité locale on peut écrire pour $g_{\mathcal{O}}(a) \geq r_{S_g}(f)$ et $g_{\mathcal{O}}(b) \geq r_{S_g}(f)$. On déduit donc qu'en chaque extrémité de e on a $g_{\mathcal{O}}(a) \geq e_{\mathcal{O}}(a)$ et $g_{\mathcal{O}}(b) \geq e_{\mathcal{O}}(b)$ ce qui contredit $\mathcal{S}_r$ échantillon régulier. □

La facette h est validée par la fonction ExtendBorder. En effet, dans ce cas idéal, la *pertinence*, les contraintes topologiques imposées n'ont aucune importance puisque l'on reste tout au long du déploiement sur l'interpolant régulier. Fermetures, recollement et toutes les précautions topologiques que nous avons prises sont toujours vérifiées, puisque $\mathcal{O}$ est bien une *surface triangulée fermée*.

La récurrence est donc établie, ce qui garantit que l'algorithme reconstruit donc l'interpolant régulier pour tout échantillon régulier. □

Pourquoi alors avoir introduit tant de critères superflus ? C'est en fait, comme l'évoquent S. PETITJEAN et E. BOYER, que les conditions imposées à un échantillon pour qu'il soit régulier dans l'espace sont (très) exigeantes, beaucoup plus que leur écriture ne pourrait le laisser croire. En fait le cas du *sliver* étudié dans ce chapitre et exhibé grâce aux Alpha-formes (figure 2.14), est présent dans quasiment tous les échantillons de points en dimension 3. On peut, par exemple, songer aux procédés de numérisation qui fournissent des points correspondant à une grille. Alors dans les zones peu courbées de l'objet, on peut observer une multitude de configurations de points quasi coplonaires et quasi cocycliques, c'est à dire des *slivers*.

Ainsi, le cas des échantillons réguliers est un peu marginal en dimension 3 et nos conditions topologiques à l'extension, notre prudence au cours du déploiement est le plus souvent utile pour guider la reconstruction et éviter les erreurs.

Pour un bon échantillon

La notion d'ε-échantillon nous place, en quelque sorte, à l'extrême inverse de l'étude précédente. En effet, alors que la notion d'échantillons réguliers est un peu trop restrictive donc plus facile à maîtriser, les ε-échantillons sont beaucoup plus difficiles à manipuler. En effet, l'essentiel de leur propriétés découle de la surface lisse qui les définit et se retrouve, par passage à la limite, quand on considère un échantillon infiniment dense. Par contre, si l'on considère un ε-échantillon pour un ε précis, il est plus difficile de retrouver les propriétés attendues. Montrer un résultat

équivalent à celui obtenu pour les échantillons réguliers — c'est à dire, pour tout ε-échantillon avec un certain $\varepsilon < 1$, notre algorithme permet de reconstruire une surface triangulée, non plus égale, mais homéomorphe à la surface lisse d'origine — est beaucoup plus compliqué.

On peut cependant établir ce résultat dans le plan, où les choses se simplifient énormément, ne serait-ce que du fait de la disparition des problèmes posés par les *slivers*. Rappelons, tout d'abord, les notables simplifications de l'algorithme dans le plan. Il s'agit maintenant de déployer une courbe polygonale $\mathcal{P}$ sur un échantillon de points $\mathcal{S}$. Le bord de notre courbe correspond alors aux deux extrémités de la courbe $\mathcal{P}$, notées $a_{\mathcal{P}}$ et $b_{\mathcal{P}}$. Le choix d'une arête candidate autour d'un point du bord se fait toujours grâce au rayon du plus petit cercle circonscrit vide de points de $\mathcal{S}$. Le critère d'exclusion des *slivers* n'est évidemment plus utile. Les *contraintes topologiques* à l'extension d'une courbe polygonale $\mathcal{P}$ ne correspondent plus qu'à deux situations.
L'extension : quand on choisit à partir d'un point du bord une arête dont l'extrémité est un point non intérieur à $\mathcal{P}$.
La fermeture : quand l'arête ajoutée relie les deux points du bord et ferme ainsi la courbe $\mathcal{P}$.
On conserve les mêmes critères pour évaluer la pertinence des arêtes candidates, bien que cela soit beaucoup moins critique qu'en dimension 3.

Pour mieux cerner les différentes propriétés de la fonction LFS, on peut établir plusieurs résultats intéressants :

Propriété 3.3

(i) Pour tout point p et q de la courbe lisse $\mathcal{C}$, la fonction LFS *vérifie l'inégalité [ABE98] :*

$$\mathrm{LFS}(p) \leq \mathrm{LFS}(q) + \mathrm{d}(p,q)$$

(ii) Pour tout point p de $\mathcal{S}$, un échantillon quelconque de points pris sur $\mathcal{C}$, considérons $V(p)$, l'ensemble des sommets *de la cellule de* Voronoi *associée à p. Alors, l'évaluation de la fonction* LFS *en p satisfait l'inégalité suivante :*

$$\mathrm{LFS}(p) \leq 2\mathrm{d}(p, V(p))$$

Preuves.

* Pour montrer la propriété (i) établie par N. Amenta *et al.* [ABE98], il suffit de remarquer que la boule $\mathcal{B}(q, \mathrm{LFS}(q)) \subset \mathcal{B}(p, \mathrm{LFS}(q) + \mathrm{d}(p,q))$, ainsi la boule $\mathcal{B}(p, \mathrm{LFS}(q) + \mathrm{d}(p,q))$ contient un point de l'axe médian. Le rayon de cette boule est supérieur à $\mathrm{LFS}(p)$, par définition de LFS. □
* Pour vérifier la propriété (ii), considérons un point p, on remarquera que $r_{\mathcal{S}}(p)$ le plus petit des *rayons*[24] associés aux triangles incidents à p correspond exactement à $\mathrm{d}(p, V(p))$. Or, dans le plan, on peut énoncer le résultat suivant ([ABE98], lemme 2) : toute sphère circonscrite à trois points de $\mathcal{S}$ contient nécessairement un point de l'axe médian. Ainsi la distance de p à l'axe médian vérifie $\mathrm{LFS}(p) \leq 2r_{\mathcal{S}}(p)$. □

Reprenons, maintenant, quelques définitions et résultats formulés par N. Amenta *et al.* dans [ABE98] pour une courbe lisse $\mathcal{C}$ munie d'un «bon» échantillon de points $\mathcal{S}$, *i.e.* un ε-échantillon avec $\varepsilon < 1$:

Propriété 3.4 *Pour $\mathcal{S}$, un ε-échantillon avec $\varepsilon < 1$ pris sur une courbe lisse $\mathcal{C}$:*

(i) Deux points p et q de l'échantillon $\mathcal{S}$ sont adjacents *si et seulement si il existe un disque circonscrit à $[p,q]$ centré sur $\mathcal{C}$ vide de points de $\mathcal{S}$, que l'on nommera* disque de Voronoi de surface.

[24] Le rayon de la plus petite sphère circonscrite vide évidemment !

(ii) La triangulation de DELAUNAY *de $\mathcal{S}$ contient toutes les arêtes liant des points* adjacents *sur $\mathcal{C}$.*

(iii) Pour un point $p \in \mathcal{S}$, notons p_0 et p_1 les deux points adjacents à p sur $\mathcal{C}$, alors on a l'inégalité $\widehat{(p_0, p, p_1)} \geq \pi - 4\arcsin(\varepsilon/2)$.

Grâce à ces différents résultats nous allons pouvoir établir le résultat attendu. En fait, pour être plus précis, nous allons montrer qu'en tout point p de l'échantillon, les deux sommets adjacents à p sont liés à p dans la triangulation de DELAUNAY par les arêtes qui possèdent les deux plus petits *rayons* parmi les arêtes incidentes à p.

Propriété 3.5 *Pour tout ε-échantillon $\mathcal{S}$ avec $\varepsilon \leq 1/5$, pris sur une courbe lisse $\mathcal{C}$, notre algorithme de déploiement reconstruit une courbe polygonale homéomorphe à la courbe $\mathcal{C}$.*

Preuve. Tout d'abord, bornons la distance entre deux points adjacents. Par définition d'un ε-échantillon et grâce à la propriété 3.4 (i), pour tous points p et q adjacents, il existe un disque de VORONOI centré en un point c de $\mathcal{C}$ de rayon au plus $\varepsilon\mathrm{LFS}(c)$ vide de points de $\mathcal{S}$. Ainsi $\mathrm{d}(p,q) \leq 2\varepsilon\mathrm{LFS}(c)$. En utilisant la propriété 3.3 (i), on a :

$$
\begin{array}{lrcllcll}
 & \mathrm{LFS}(c) & \leq & \mathrm{LFS}(p) + \mathrm{d}(p,c) & \text{or,} & \mathrm{d}(p,c) & \leq & \varepsilon\mathrm{LFS}(c) \\
\text{ainsi, on a} & \mathrm{LFS}(c) & \leq & \frac{1}{1-\varepsilon}\mathrm{LFS}(p) & & & & \\
\text{d'où} & \mathrm{d}(p,q) & \leq & \frac{2\varepsilon}{1-\varepsilon}\mathrm{LFS}(p) & \text{et} & \mathrm{d}(p,c) & \leq & \frac{\varepsilon}{1-\varepsilon}\mathrm{LFS}(p) \\
\text{soit avec } \varepsilon \leq 1/5, & \mathrm{d}(p,q) & \leq & \frac{1}{2}\mathrm{LFS}(p) & \text{et} & \mathrm{d}(p,c) & \leq & \frac{1}{4}\mathrm{LFS}(p)
\end{array}
$$

Considérons un point p de l'échantillon $\mathcal{S}$ et notons p_0 et p_1 les deux points adjacents à p. De plus, notons $\mathfrak{V}_0$ l'arête duale de $[p_0, p]$ et $\mathfrak{V}_1$ celle de $[p_1, p]$. Étudions les arêtes de la cellule de VORONOI $\mathfrak{V}(p)$ privée de $\mathfrak{V}_0$ et de $\mathfrak{V}_1$. Cet objet $Sk(p)$ est constitué de deux courbes polygonales (éventuellement réduite à un point pour l'une des deux) qui se trouvent *de facto* exclues d'un certaine zone de l'espace (figure 3.13).

Cette *zone d'exclusion* est définie par quatre points que l'on peut considérer comme *extrêmes*, *i.e.* les positions extrêmes que peuvent prendre les sommets de $\mathfrak{V}(p)$. Ces points sont définis comme les intersections, V_i^0 et V_i^1 (figure 3.13), du cercle de centre p et de rayon $\frac{1}{2}\mathrm{LFS}(p)$ avec les médiatrices respectives, pour $i \in \{0,1\}$, des segments $[p_i^{max}, p]$, où p_i^{max} est le point d'intersection de C et de la demi-droite d'origine p support de l'arête $[p_i, p]$. Les arguments pour pouvoir considérer l'intérieur de cette zone comme non intersecté par $Sk(p)$ sont :

* la cellule de VORONOI $\mathfrak{V}(p)$ étant convexe, les courbes polygonales constituant $Sk(p)$ héritent des propriétés de convexité ;
* le disque $\mathcal{B}(p)$ de centre p et de rayon $\frac{1}{2}\mathrm{LFS}(p)$ ne contient aucun sommet de $\mathfrak{V}(p)$ (propriété 3.3 (ii)) ;
* enfin, pour $i \in \{0,1\}$, la médiatrice de $[p_i, p]$ intersecte le disque $\mathcal{B}(p)$ entre les médiatrices associées à p_0^{max} et à p_1^{max} par construction. Remarquons, tout d'abord, que l'intersection, r, des deux médiatrices est située à l'extérieur de $\mathcal{B}(p)$. En effet, comme $\mathrm{d}(p,r) = 1/(4\cos(\alpha/2))\ \mathrm{LFS}(p)$ et que, d'après la propriété 3.4 (iii), $\cos(\alpha/2) < 1/5$ pour $\varepsilon < 1/5$, on a $\mathrm{d}(p,r) > \mathrm{LFS}(p)$. De plus, étant donné que $[p_i, p] \subset [p_i^{max}, p]$ et comme $\mathrm{d}(p_i^{max}, p) = \frac{1}{2}\mathrm{LFS}(p)$, les deux points extrêmes V_i^0 et V_i^1 définis pour un point p_i existent et sont distincts ; ils forment même chacun des triangles équilatéraux avec p_i^{max} et p (figure 3.13).

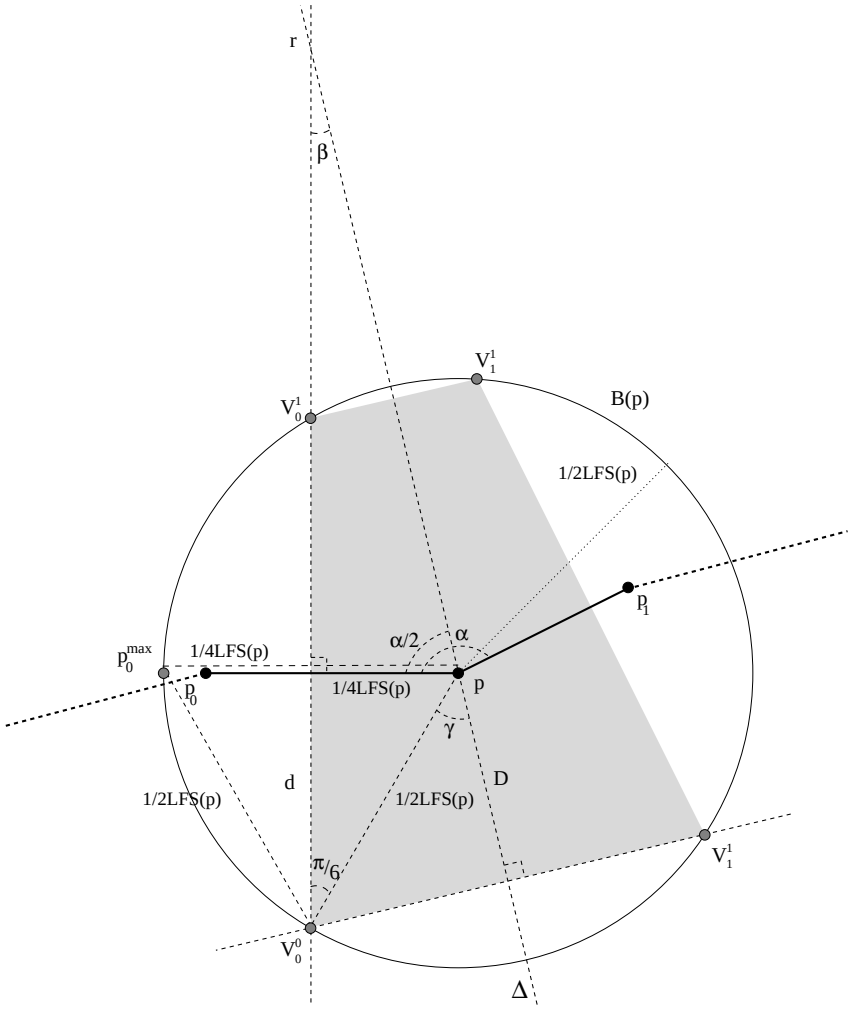


FIG. 3.13 – Illustration : zone d'exclusion pour $Sk(p)$, ensemble des contributions au squelette discret des arêtes incidentes à p et non adjacentes sur $\mathcal{C}$.

Les rôles de p_0 et p_1 étant symétriques, la zone d'exclusion possède aussi un axe de symétrie Δ, on peut donc poursuivre l'étude pour le couple (p_0, p), par exemple. On se reportera à la figure 3.13 pour y trouver les notations adoptées. Calculons les grandeurs caractéristiques de cette zone d'exclusion D et d. On a trivialement $d = \frac{\sqrt{3}}{4}\mathrm{LFS}(p)$. Pour $D = \frac{1}{2}\cos(\gamma)\mathrm{LFS}(p)$, il faut établir quelques résultats sur les angles : la propriété 3.4 (iii) nous donne $\alpha \geq \pi - 4\arcsin(\varepsilon/2)$. De plus, en utilisant les différents triangles rectangles, on peut obtenir $\beta = (\pi - \alpha)/2$ et $\gamma = \beta + \pi/6$; ainsi, on déduit l'inégalité $\gamma \leq 2\arcsin(\varepsilon/2) + \pi/6$. De plus, pour $\varepsilon \leq 1/5$, on vérifie l'inégalité *stricte* $\cos(\gamma) > 1/2$. On conclut finalement que $D > \frac{1}{4}\mathrm{LFS}(p)$.

Remarquons tout de suite que les arêtes $[p_0, p]$ et $[p_1, p]$ sont de GABRIEL. En effet, pour $[p_0, p]$ par exemple, considérons son disque de VORONOI de surface de centre c_0 sur la courbe $\mathcal{C}$. Comme $\mathrm{d}(p, c_0) \leq \frac{1}{4}\mathrm{LFS}(p)$, le disque $\mathcal{B}(p)$ contient le point c_0 qui appartient à l'arête duale de $[p_0, p]$ sans en contenir les extrémités. Ainsi, $\mathcal{B}(p)$ contient toute la portion de la médiatrice de $[p_0, p]$ qu'il intersecte et en particulier le milieu de $[p_0, p]$. De plus, on sait que, par construction, D est un minorant de la distance de p à $Sk(p)$. Par dualité, on déduit donc que $[p_0, p]$ et $[p_1, p]$ possèdent

les deux plus petits rayons r_0 et r_1 parmi toutes les arêtes incidentes à p. En effet, toutes les autres arêtes ont leur plus petit cercle circonscrit vide de rayon plus grand que $D > \frac{1}{4}\text{LFS}(p)$ et on a $\max(r_0, r_1) \leq \frac{1}{4}\text{LFS}(p)$. □

Cette borne de 1/5 peut certainement être améliorée vue les approximations faites, mais elle a le mérite de garantir la validité de notre approche dans le plan. On peut évidemment comparer cette condition à celle établie par N. Amenta pour le *Crust* [ABE98] $\varepsilon < 0,252$ que nous n'améliorons pas. Cependant, les bons résultats obtenus en pratique dans l'espace (cf. section 3.5.2) nous confortent dans l'idée que nos critères sont pertinents pour la résolution du problème de la reconstruction. Par ailleurs, les éléments de cette démonstration permettent d'établir un lien avec les interpolants réguliers. On peut ainsi énoncer le corollaire suivant :

Corollaire 3.6 *En dimension 2, un ε-échantillon avec $\varepsilon \leq 1/5$ est un échantillon* régulier.

Preuve. Considérons $\mathcal{S}$ un ε-échantillon d'une courbe lisse $\mathcal{C}$ avec $\varepsilon \leq 1/5$, alors la courbe polygonale $\mathcal{P}$ liant les points adjacents dans $\mathcal{S}$ est un *interpolant régulier*. En effet, en tout point p de $\mathcal{S}$ on peut vérifier $g_{\mathcal{O}}(p) \leq \frac{1}{4}\text{LFS}(p) < e_{\mathcal{O}}(p)$. □

La difficulté pour étendre la preuve précédente à la dimension 3 est essentiellement due à l'existence des *slivers*. En effet, même pour un échantillon très dense pris sur une surface lisse $\mathcal{V}$, cette configuration permet à certains sommets du diagramme de Voronoi d'être très proches de la surface $\mathcal{V}$ ce qui nous empêche de définir «aussi simplement» une zone d'exclusion pour le squelette discret dans l'espace.

3.5.2 *Etude des performances*

Etude qualitative

Avant de donner des résultats bruts, tentons de donner quelques ordres de grandeurs permettant d'apprécier les performances de notre approche. Les considérations qui vont suivre se fondent sur les résultats fournis par l'outil d'analyse d'exécution de programmes *gprof*. Ainsi, au cours d'une exécution typique de notre algorithme de déploiement, l'évaluation du temps passé pour chaque étape de la méthode, nous a permis de mieux cerner les opérations coûteuses et d'atteindre un niveau de performance tout à fait compétitif. Nous détaillerons ensemble ce qui constitue le corps de notre méthode, le calcul de la triangulation de Delaunay, suivi du déploiement de la surface, en réservant à l'étape de post-traitement un commentaire particulier.

Le premier constat est que, naturellement, le calcul de la triangulation de Delaunay représente une part non négligeable du temps d'exécution de notre algorithme de reconstruction. Cette étape a un coût certain, mais qui se révèle toutefois acceptable en pratique, surtout grâce à l'implantation fournie par CGAL et aux différentes optimisations qu'elle propose. Grâce à l'utilisation d'une arithmétique filtrée et d'une version de la triangulation de Delaunay hiérarchique (prochainement disponible dans CGAL version 2.4), le coût de ce pré-requis a été divisé par un facteur dix depuis la version de CGAL précédente. Il ne représente plus qu'environ la 50% du temps d'exécution global. On notera, par ailleurs, que le cas de points échantillonnés sur une surface est loin de représenter un cas facile pour le calcul de Delaunay, du fait de nombreuses configurations dégénérées.

Commentons, maintenant, l'étape d'*initialisation*, qui consiste essentiellement à calculer le rayon pour chaque facette de la triangulation de Delaunay afin de sélectionner un germe. Ces calculs, qui représentent l'essentiel du coût de notre méthode en calcul numérique, sont pleinement mis à profit, puisque l'on conserve les valeurs des rayons en mémoire. Ainsi, l'étape

de déploiement proprement dit, où les différents rayons sont comparés plusieurs fois entre eux, se trouve grandement allégée. On peut considérer qu'elle intervient pour 25% du temps d'exécution global.

Enfin, au cours de la phase de déploiement occupant les 25% du temps d'exécution restant, la *moitié* du temps est passé dans la fonction ComputeCandidate, chargée de fournir des candidats pour l'extension du bord, le reste étant occupé à la vérification et la validation de ces candidats — suivi de la mise à jour des différentes structures — par la fonction ExtendBorder, chargée du maintien d'une bonne topologie. On notera que cette étape qui construit réellement est proportionnellement très peu coûteuse, les étapes précédentes ayant, en quelque sorte, préparé le travail.

Donnons, enfin, quelques commentaires sur l'exécution de l'étape de post-traitement. Sa raison d'être étant, en quelque sorte, de corriger les éventuelles erreurs commises autour des arêtes vives ou dans des zones très mal échantillonnées, elle s'avère difficile à comparer avec les autres composantes d'une exécution moyenne. En fait, on peut considérer son coût comme négligeable dans la majorité des cas. Elle n'intervient que pour résoudre des situations, en général, très complexes et son exécution peut alors se révéler coûteuse. L'essentiel du coût est à imputer au coût élevé en pratique du *retrait d'un point* dans une triangulation de DELAUNAY, surtout pour les configurations particulièrement dégénérées qui sont les nôtres. Une implantation de cette opération plus performante que celle disponible actuellement dans CGAL permettrait certainement de ramener l'étape de post-traitement à un coût comparable aux étapes précédentes.

Etude statistique

Cette partie a pour but de commenter les résultats chiffrés obtenus sur l'ensemble des modèles à notre disposition. Les résultats présentés dans le tableau 3.1 ont été obtenus sur une machine équipée d'un processeur INTEL PENTIUM i686 1GHz et disposant de 1Go de RAM. Notre méthode a été testée sur des modèles très variés et de qualité inégale. On trouvera principalement deux types d'objets, ceux dont on peut présumer qu'ils représentent à l'origine un volume et ceux qui représentent des surfaces à bords, provenant de numérisation partielle par exemple. Les modèles que nous avons présentés dans cette catégorie des surfaces à bords sont parfois reconstruits comme des volumes par d'autres méthodes (en particulier les méthodes de nature volumique évidemment). On peut cependant facilement se convaincre au regard de l'échantillon que le volume ainsi défini n'est en fait qu'un artefact produit par la méthode.

Surfaces fermées. Commentons, tout d'abord, les modèles représentant des volumes. Si l'on considère que les échantillons de points ne sont pas bruités, on peut aisément apprécier la qualité de la reconstruction fournie par notre méthode en observant le nombre de points de la surface reconstruite par rapport au nombre de points total du modèle et le nombre «d'arêtes du bord», qui doit idéalement être nul.

Au regard du premier critère évoqué, on peut, tout d'abord, s'étonner de voir plusieurs modèles laisser un certain nombre de points de côté. Après examen, il s'avère que bien souvent ces points sont en fait de deux types : *soit* on peut réellement les considérer comme des points parasites et les considérer dans la reconstruction serait totalement aberrant, *soit* ces points sont situés autour d'arêtes vives ou plus généralement autour de singularités de l'objet originel. On peut, par exemple, citer le cas du modèle *cube* ($[0,1]^3$) échantillonné uniformément sur une grille, qui peut être aisément reconstruit en prenant l'enveloppe convexe, mais qui, d'un point de vue surfacique, possède des arêtes vives et pire encore, en chaque sommet une singularité où

l'échantillonnage est très ambigu vis à vis de notre approche. Surtout comme dans le cas du modèle *cube* où les données sont des *floats*, il est alors possible que notre méthode considère certains points anguleux comme marginaux.

Pour le second critère permettant d'apprécier la qualité de la reconstruction, à savoir fournir une surface réellement fermée, on peut observer que les résultats sont le plus souvent corrects. En fait, paradoxalement, c'est sur les échantillons les plus volumineux, donc *a priori* les plus fins, que l'on n'arrive pas à fournir une surface sans trous, même après un coûteux (mais utile) post-traitement. Pour expliquer cela, nous avons évidemment analysé les zones à problèmes et il s'est avéré que ces gros échantillons ne sont pas si «bons» que cela. Pour être plus précis, les objets qu'ils modélisent sont très complexes, pas nécessairement par leur topologie, mais ils possèdent des détails très fins. On peut même considérer que l'essentiel des points fournis dans ces modèles se situent dans des zones où ils ne sont d'aucune utilité car très simples à reconstruire, les zones difficiles se trouvant finalement relativement mal échantillonnées. On observe même parfois des zones difficiles à interpréter en tant que volume sans considérer qu'il manque des points sur tout un pan de la surface originelle.

Surfaces à bords. Parlons maintenant des modèles que nous avons considérés comme des surfaces à bord. Pour certain d'entre eux, nous aurions pu pousser plus loin la reconstruction et obtenir une surface fermée ; cependant, au regard de l'échantillon nous avons considéré qu'une frontière courbe semblait effectivement délimiter un trou sans aucun point intérieur sur l'objet. On peut classer dans cette catégorie les modèles : *Culbu*, *Rth*, *cactus*, *cat10*, *distcap*, *club0h*, *mannequin*. Pour les autres modèles, il n'y a aucune ambiguïté, les objets à reconstruire sont de nature surfacique, parfois même on ne peut les interpréter aisément comme des surfaces délimitant un volume intérieur (modèle *hypersheet* cf. figure 3.22 ou modèle *Tomo* cf. figure 3.23). On peut toujours faire les mêmes remarques vis à vis du nombre de points considérés dans la surface reconstruite que pour le cas des surfaces fermées. Par contre le nombre d'arêtes sur le bord n'a naturellement plus la même signification. On n'observe, en fait, que très peu d'erreurs sur l'ensemble des modèles de cette catégorie présentés dans le tableau 3.1. Les bords (que l'on a pris soin de repérer en rouge sur la surface reconstruite pour les captures d'écran présentées ci-après) correspondent à de vrais contours dessinés sur les surfaces, les trous qui peuvent être considérés comme des erreurs étant refermés par l'étape de post-traitement (excepté pour les modèles *Cardoor* et *Voiture* dont le temps de post-traitement peut déjà suggérer de vrais problèmes).

Performances moyennes. Sur la base des résultats du tableau 3.1, on peut établir un temps moyen de traitement pour les échantillons de points suffisamment «bons» (*i.e.* si l'on excepte le post-traitement des cas réellement problématiques, où l'on considère alors seulement $t_1 + t_2$) d'environ *6000 points par seconde* pour obtenir la reconstruction à partir des données brutes. Un autre aspect important est le coût pratique en espace mémoire. Une étude empirique sur les échantillons de notre base de données et un calcul rapide du coût des différentes structures utilisées, nous permettent d'évaluer une moyenne *de 1 octet par point*, pour l'exécution de l'ensemble, triangulation, déploiement et post-traitement. On peut évidemment abaisser un peu ce coût si l'on ne conserve pas les *rayons en mémoire*. Mais on peut considérer que l'on se trouve encore limité à des échantillons de l'ordre du million de points avec notre approche, au regard du matériel actuel ($\sim$ 1Go).

Données		DELAUNAY		Reconstruction					$\sum t_i$
Modèle	points	cellules	t_1	points	facettes	arêtes du bord	t_2	t_3	
* Surfaces fermées :									
Mechanic	12593	85158	0,99	12593	25194	0	1,03	0	2,02
blade	882954	6071275	103,76	*878724*	1757123	1005	105,28	20,38	229,42
bunny	35946	249345	2,75	*35943*	71882	0	3,19	0,01	5,95
cube	866	4172	2,78	*861*	1718	0	0,04	0,02	2,84
dragon	435545	314951	49,02	*422441*	844546	442	47,86	97,75	194,63
happy	542548	3864194	63,36	*522701*	1045009	567	70,02	182,27	315,65
knot108s	10000	93505	0,83	10000	20000	0	1,03	0,01	1,87
mechpart	4102	30638	0,38	*4098*	8204	0	0,33	0,03	0,74
papaine1	12577	92619	0,9	*12562*	25120	0	1,07	0,02	1,99
papaine2	18059	131943	1,39	*18045*	36086	0	1,54	0,02	2,95
hand	327323	2302290	38,57	*327290*	654596	0	33,25	0,56	72,38
holes3	3999	29352	0,25	3999	8006	0	0,33	0,01	0,59
horse	48485	342428	3,94	*48473*	96942	0	4,46	0,39	8,75
oilpmp	30933	200516	2,91	*30919*	61834	0	2,58	0,04	5,53
sphere	926	5070	0,52	926	1848	0	0,04	0	0,56
torus_random	499	3633	0,03	499	998	0	0,04	0	0,07
volume_Inria	187151	1291512	24,6	*186799*	373503	205	18,11	2,26	44,97
* Surfaces à bords :									
Avant_Auto	59497	394366	5,46	59491	118232	748	5,27	0,25	10,98
Culbu	71150	483383	6,12	71150	142210	88	6,52	0,03	12,67
BootSki	33711	239977	2,94	*33700*	66005	1393	3,04	3,29	9,27
British_Museum	51095	340275	4,15	*51053*	101899	251	4,42	0,18	8,75
CarDoor	489205	3217628	56,86	*488278*	973402	3334	49,12	13,43	119,48
DryerBody	31013	192207	6,66	*31010*	61093	921	2,34	0,33	9,33
DryerHandle1	49786	319549	18,77	49786	98811	759	3,94	0,02	22,73
DryerHandle2	64464	408629	15,67	64464	128260	666	5,13	0,02	20,82
HairDryer	51563	321389	14,75	51563	10248	641	3,95	0,02	18,72
Robinet	13962	101468	1,17	*13884*	27309	455	1,4	0,17	2,74
Rth	13894	94824	1,07	13894	27557	231	1,16	0	2,23
Tomo	47861	330959	4,72	*47839*	92861	2883	15,99	0,12	20,83
Voiture	407577	2646522	46,29	*403629*	801332	7524	37,46	44,58	128,33
bear	29648	203529	3,64	29621	58553	687	2,4	0,11	6,15
cactus	3280	22908	2,69	*3277*	6532	20	0,24	0	2,93
cat10	10000	66700	0,69	10000	19908	90	0,82	0	1,51
ch1000	59104	383939	5,57	59104	117707	499	5,26	0,03	10,86
distcap	12745	86887	0,9	*12744*	25325	161	1,04	0	1,94
engine	11360	73253	1,02	11360	22360	356	0,98	0,01	2,01
hypersheet	6752	52366	0,6	6752	12575	935	0,61	0,01	1,22
club0h	15245	103129	1,17	*15232*	30438	24	1,30	0,06	2,53
foundry	4101	28111	0,26	4101	7958	242	0,34	0	0,6
mannequin	12772	86663	0,91	12768	25366	168	1,03	0,03	1,97
mmal25	25921	172441	2,89	25921	51201	639	2,98	0,02	5,89
mmma25	6561	43359	0,6	6561	12833	287	0,56	0	1,16
moeller	20020	137183	1,62	20020	39964	74	1,71	0,01	3,34
monkey2	10000	68462	2,98	10000	19611	387	0,84		3,82
nascar	20621	129788	1,58	*20619*	40725	515	1,65	0,01	2,88
seat	14812	96922	5,16	14812	29248	374	1,29	0	6,45
skidoo_red	37973	257065	3,08	*37960*	75389	531	3,33	0,05	6,46

TAB. 3.1 – Temps de calcul CPU (en secondes) pour les modèles de notre base de données.

Abréviations :

t_1 = calcul de DELAUNAY — t_2 = initialisation + déploiement
t_3 = post-traitement — $\sum t_i$ = temps total $t_1 + t_2 + t_3$

Provenance des modèles : Nous tenons à remercier, DASSAULT SYSTÈMES grâce à qui le projet PRISME dispose de modèles originaux qui nous ont grandement aidés dans la confrontation de nos idées à des cas concrets. La plupart des autres modèles proviennent du web, on pourra les trouver aux adresses suivantes :
`http://www.research.microsoft.com/~hoppe/`
`http://www.cc.gatech.edu/projects/large_models/`
`http://www.swisstopo.ch/`

3.5.3 *Comparaison avec les méthodes standard*

Cette partie a pour objectif de présenter une étude empirique entre les performances de notre méthode et les différentes approches évoquées dans la littérature. Nous avons donc établi un tableau récapitulatif de l'ensemble des résultats présentés dans les différents articles décrivant les méthodes afin de tenter une comparaison sur les modèles traités en commun. Les résultats de cette étude sont présentés dans le tableau 3.2. Cette comparaison ne peut être faite sur les temps de calcul bruts, elle doit évidemment être relativisée du fait de l'exécution sur des matériels différents. Bien que cela soit contestable, nous avons considéré sur la base de tests effectués avec notre propre algorithme que le rapport r_{cpu} des cadences CPU pouvait permettre une comparaison approximative entre des temps de traitement pour des machines différentes. De plus, pour établir les chiffres fournis sur la ligne «temps comparés» du tableau 3.2, nous avons tenu à prendre le temps, pondéré par r_{cpu}, qui nous est le plus défavorable (atteint pour le modèle dont le temps est précédé d'une étoile $*$) pour calculer le rapport de comparaison avec la méthode concurrente. Ce tableau n'a que peu de prétentions à l'objectivité, il prétend juste montrer que notre méthode se place parmi les algorithmes efficaces en ce qui concerne les temps d'exécution pour la résolution du problème de la reconstruction.

Par ailleurs, il faut évidemment prendre en considération la qualité de la reconstruction obtenue. A ce sujet, les résultats décrit dans la littérature ne sont que rarement explicites. On peut noter, parmi les articles, ceux où les auteurs se plaignent de problèmes pour la reconstruction obtenue. On présumera donc que si aucune mention n'est faite, c'est que le résultat est correct. Ainsi, la méthode *LD* [GKS00] paye sa grande rapidité par des résultats d'une qualité contestable, les résultats fournis possédant de nombreux trous dans le cas d'échantillons non uniformes (par exemple pour le modèle *holes3*) et parfois même une triangulation non valide de la surface reconstruite. On citera aussi les commentaires sur les résultats obtenus par SUPER COCONE [TKDH01], REVIVER, COCONE [ACDL00] pour lesquels il est fait mention de problèmes concernant des trous ou des irrégularités dans la surface reconstruite. En ce qui concerne l'utilisation de la mémoire les seuls résultats évoqués sont ceux de SUPER COCONE, qui utilise approximativement $0,3$ octet par point, soit environ un tiers de ce dont nous avons besoin. Au regard de ces différents angles de comparaison, nous pouvons donc considérer que notre méthode se place de manière concurrentielle par rapport aux méthodes standard.

Algorithmes :		DS	SC	PC	AS	RI	LD	HH	RV	CR
r_{cpu} = rapports des cadences CPU		1	0,7	0,7	0,25	0,27	0,2	0,15	0,7	0,3
Modèle	points									
* Surfaces fermées :										
Mechanic	12593	2,02	-	27	-	-	-	-	-	-
blade	882954	229,42	3000	-	-	-	-	-	-	-
bunny	35946	5,95	-	-	1176	-	18,64	-	*40	-
dragon	435545	194,63	1080	-	-	-	-	-	-	-
happy	542548	315,65	*1680	-	-	-	-	-	-	-
knot108s	10000	1,87	-	-	-	-	-	75	-	-
mechpart	4102	0,74	-	-	-	*16	-	14	-	-
papaine2	18059	2,95	-	398,68	-	-	-	-	-	-
hand	327323	72,38	900	-	-	-	-	-	-	-
oilpmp	30933	5,53	-	-	-	-	20,99	*104	-	-
* Surfaces à bords :										
Avant_Auto	59497	10,98	-	167	-	-	-	-	-	-
Culbu	71150	12,67	-	190	-	-	-	-	-	-
BootSki	33711	9,27	-	*65	-	-	-	-	-	-
Robinet	13962	2,74	-	32	-	-	-	-	-	-
Rth	13894	2,23	-	-	-	-	-	-	392	-
cactus	3280	2,93	-	-	-	-	-	-	25	-
distcap	12745	1,94	-	-	-	-	-	40	-	-
club0h	15245	2,53	-	-	246	-	7,9	-	-	189
mannequin	12772	1,97	-	-	*126	-	16,66	52	35	-
mmal25	25921	5,89	-	-	-	198	-	-		
moeller=foot	20020	3,34	-	-	-	-	10,53	-	-	*234
skidoo_red	37973	6,46	-	-	-	-	*16,66	-	-	-
* Temps comparés (min pour $*$) :										
$\min r_{cpu} T_i / T_{DS}$		1	3,73	4,91	15,99	6,08	0,54	2,82	4,7	20,94

TAB. 3.2 – Comparaison des temps de calcul CPU (en secondes) avec les méthodes standard.

Abréviations :

DS = notre algorithme par déploiement de surface (INTEL PENTIUM 1GHz),
SC = SUPER COCONE [TKDH01] (INTEL PENTIUM 733MHz),
PC = PC NUAGES [BC00] (INTEL PENTIUM 733MHz),
AS = *Sampling and Reconstructing Manifolds Using Alpha-Shapes* [BB97] (SGI 250MHz),
RI = *Regular and non-regular point sets : Properties and reconstruction* [PB01] (SUN 270MHz),
LD = *Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation* [GKS00] (SGI 200MHz),
HH = *Surface Reconstruction from Unorganized Points* [HDD+92] (SGI 150MHz) résultats provenant du web `http://www.research.microsoft.com/~hoppe/`,
RV = REVIVER : logiciel implantant le Crust (INTEL PENTIUM 733MHz) résultats provenant du web `http://www.ams.sunysb.edu/~piyush/reviver/`,
CR = COCONE, *i.e.* le *Crust*, par N. AMENTA *et al.* [ACDL00] (SUN 300Mhz).

3.5.4 *Résultats et commentaires*

Les reconstructions présentées ci-après correspondent aux calculs détaillés dans le tableau 3.1. Laissons maintenant place à quelques résultats en images :

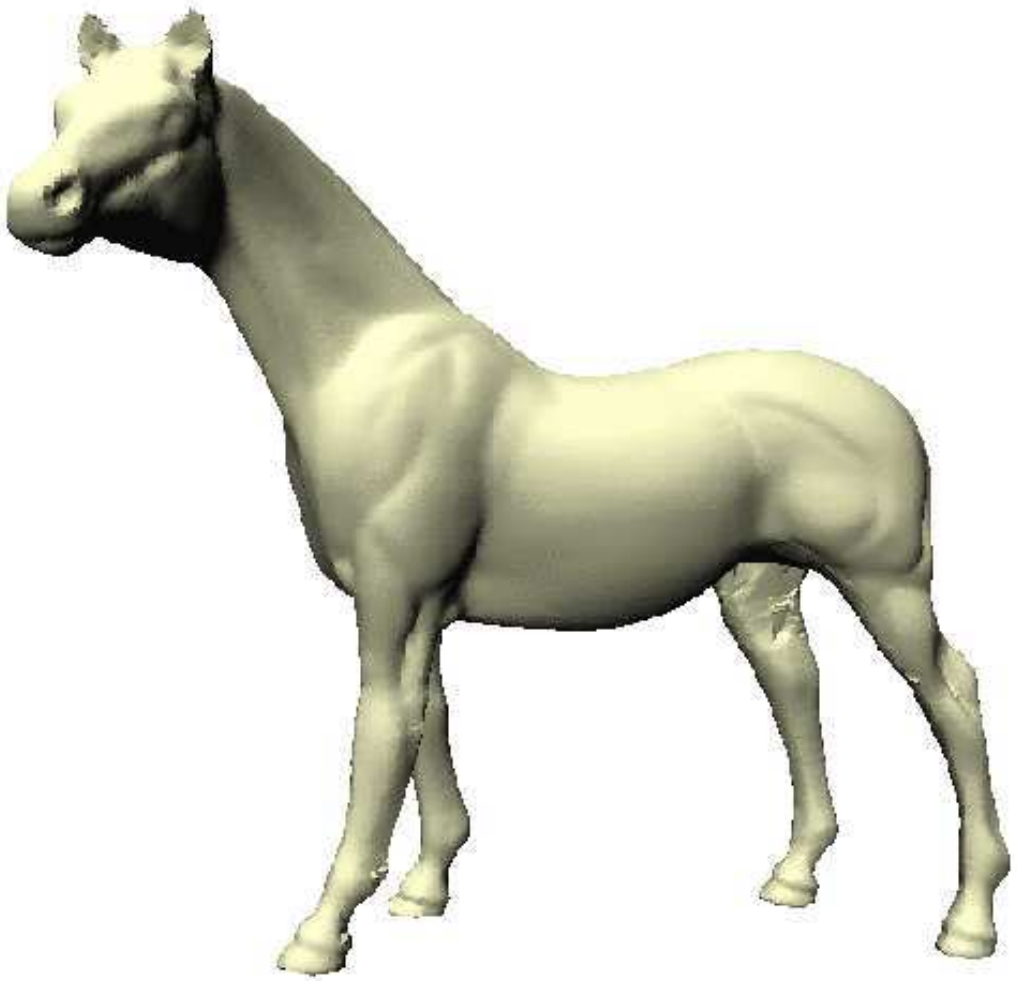


FIG. 3.14 – Résultat : un cheval, modèle *horse*.

Figure 3.14. Un modèle intermédiaire, qui ne présente *a priori* pas de réelles difficultés. En fait, cet échantillon ayant vraisemblablement été numérisé à partir d'une statuette, certaines zones sont très mal échantillonnées, avec de fortes variations de densité. Notre méthode permet tout de même d'obtenir une surface fermée.

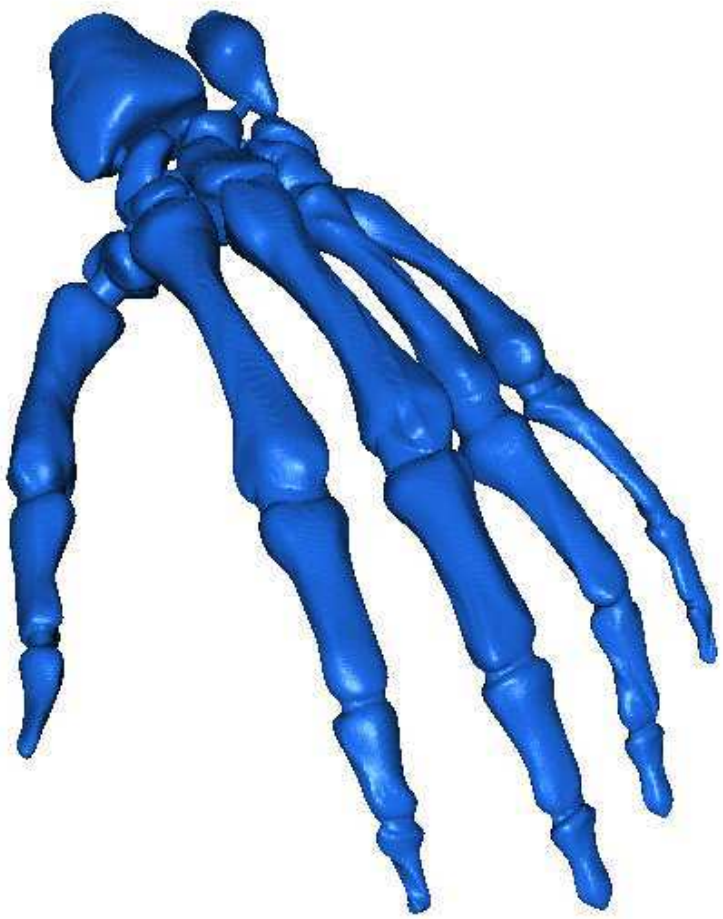


FIG. 3.15 – Résultat : une main, modèle *hand*.

Figure 3.15. Ce modèle présente plusieurs particularités, qui en font une grande réussite de notre méthode. Tout d'abord l'échantillon est relativement volumineux ($\sim$ 330.000 points), ce qui le place hors de portée d'une méthode trop gourmande en mémoire. Cependant bien que l'échantillonnage soit dense, il est à peine suffisant dans certaine zones du modèle, comme les jointures des doigts qui présentent une très forte courbure ou encore sur le poignet qui, étant creux, présente des parois intérieures et extérieures très proches les unes des autres. La reconstruction fournie est là encore correcte (*i.e.* tous les bords ont été résorbés).

FIG. 3.16 – Résultat : une pièce mécanique, modèle *Mechanic*.

Figure 3.16. Ce modèle présente plusieurs aspects intéressants : tout d'abord l'échantillonnage n'est pas uniforme, mais surtout il possède une topologie non simple et permet de montrer le succès de notre méthode pour des modèles volumiques non triviaux. Bien sûr, la surface obtenue est fermée.

FIG. 3.17 – Résultat : une figurine, modèle *happy*.

Figure 3.17. Ce modèle fait partie des gros échantillons dont nous disposons ($\sim$ 540.000 points), il possède cependant des détails très fins et se révèle, en fait, très complexe. La reconstruction fournie est correcte dans l'ensemble. Il reste cependant des bords non résorbés, même après l'étape de post-traitement. Le résultat fourni ne délimite donc pas un volume. Après examen, les erreurs subsistent principalement autour de zones où la cohérence de l'échantillon peut être mise en doute.

FIG. 3.18 – Résultat : un dragon, modèle *dragon*.

Figure 3.18. Cet autre modèle volumineux ($\sim$ 435.000 points) présente des caractéristiques équivalentes à celles observées pour le modèle 3.17. La reconstruction d'ensemble est correcte, les erreurs localisées autour de zones ambiguës de très forte courbure.

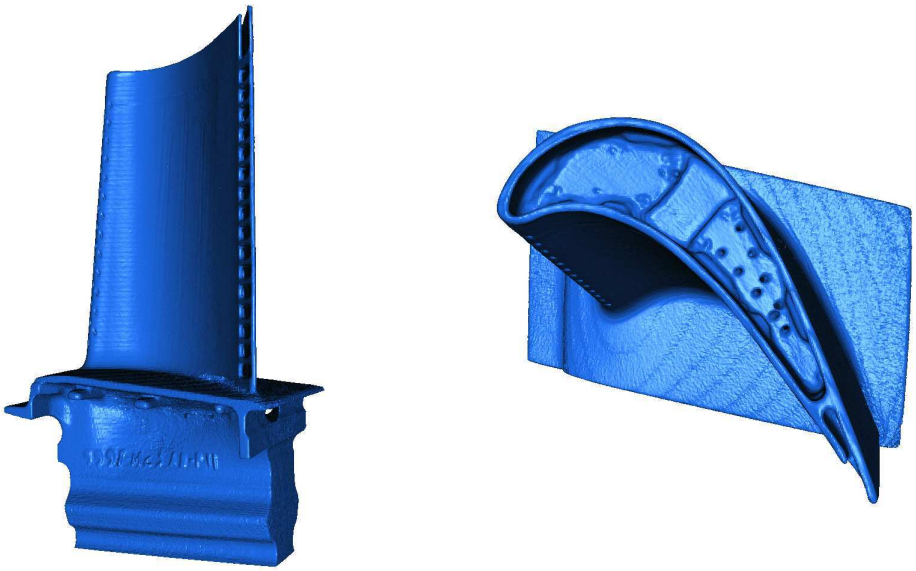


FIG. 3.19 – Résultat : une lame, modèle *blade*.

Figure 3.19. Voici le plus gros modèle ($\sim$ 880.000 points) en notre possession, qui correspond, à peu près, à ce que nous pouvons traiter de plus volumineux avec le matériel dont nous disposons aujourd'hui ($\sim$ 1Go). Par ailleurs, on notera qu'il présente les mêmes «défauts de fabrication» que les gros modèles 3.17 et 3.18. Sans être parfait, le résultat est là encore satisfaisant au moins dans l'ensemble.

FIG. 3.20 – Résultat : une statue du *British Museum*, modèle *British_Museum*.

Figure 3.20. Ce modèle est intéressant pour plusieurs raisons. D'une part, son échantillonnage présente des variations de densité vraiment très importantes y compris dans des zones très courbées, sur les angles du socle par exemple. A cette difficulté s'ajoutent de nombreux points parasites sur le sol au pied du socle, que l'on peut considérer comme du bruit. On peut voir que notre méthode conserve, vis à vis de ces deux difficultés, un comportement acceptable. La reconstruction fournie est totalement correcte si l'on excepte le bas du socle.

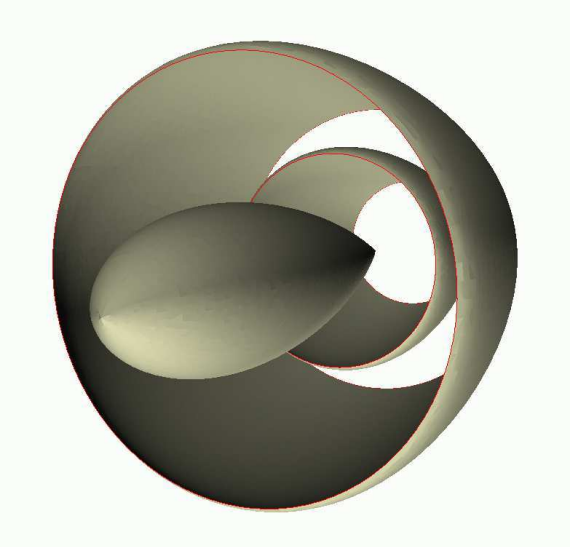


FIG. 3.21 – Résultat : une tuyère, modèle *engine*.

Figure 3.21. Voilà un joli modèle de surface à bords ! Cet exemple nous permet d'illustrer les possibilités spécifiques de notre méthode. Ce modèle fait partie de ceux qui ne peuvent être facilement reconstruits comme des volumes alors qu'ils proposent un échantillon de points sans difficulté, puisque uniforme et suffisamment dense. On pourra remarquer les bords repérés en rouge.

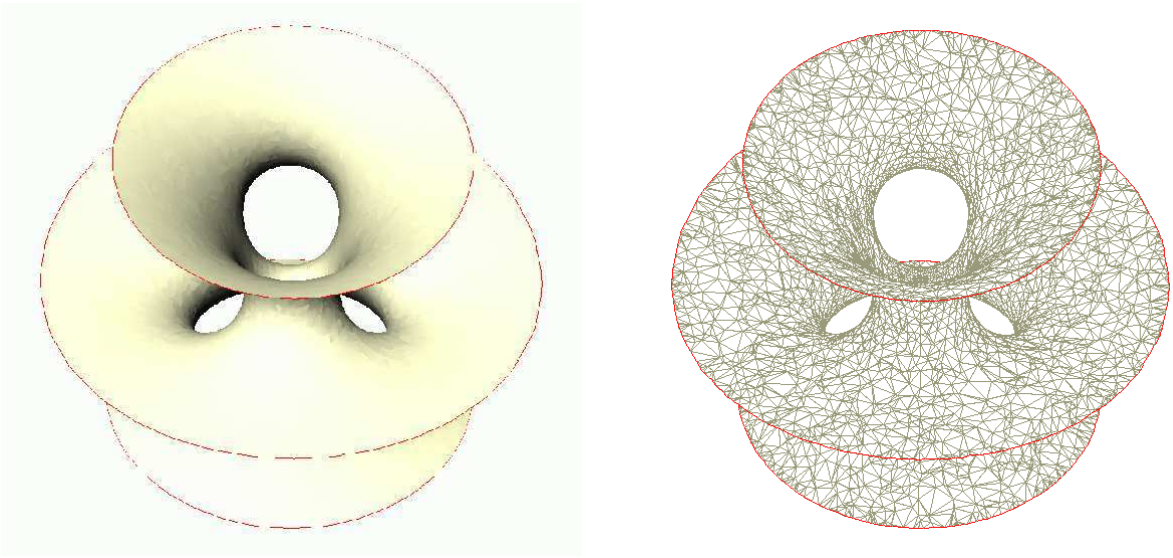


FIG. 3.22 – Résultat : une surface orientable, modèle *hypersheet*.

Figure 3.22. Ce modèle représente une surface orientable pour laquelle il est, là encore, difficile de délimiter un volume intérieur cohérent sans briser la symétrie : le feuillet central peut être considéré comme extérieur en même temps sur ses deux faces, pour des raisons de symétrie, quel que soit le critère utilisé. On notera, de plus que cet objet n'est pas échantillonné uniformément.

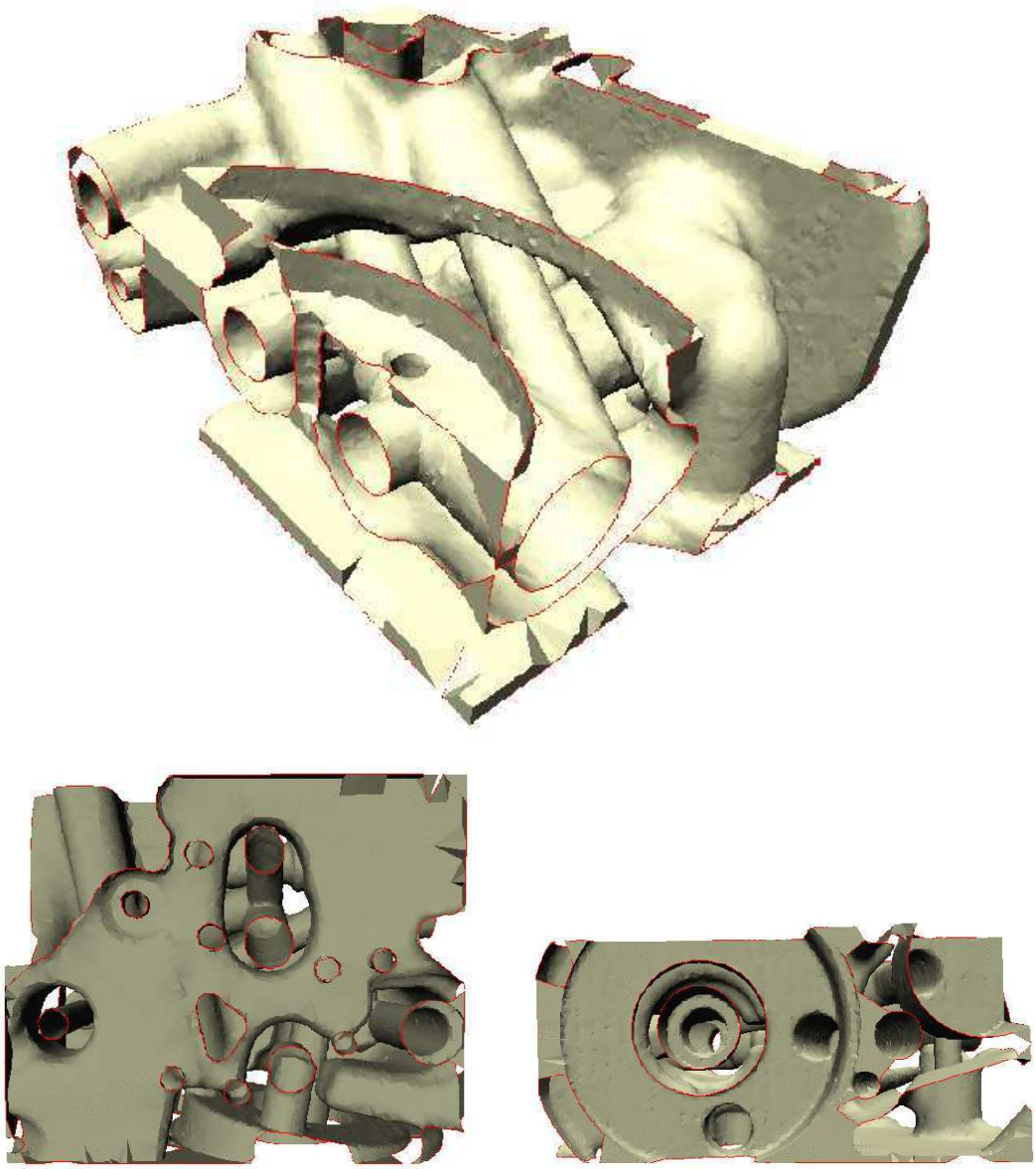


FIG. 3.23 – Résultat : une pièce usinée, modèle *Tomo*.

Figure 3.23. Enfin, un autre exemple de surface difficile (voir impossible) à appréhender pour bien des méthodes, de par la difficulté à lui associer un volume intérieur cohérent. Contrairement aux deux modèles précédents 3.21 et 3.22, cet objet est issu de la numérisation d'un objet réel et présente ainsi une configuration des points plus complexe. Là encore, les bords sont repérés en rouge. On peut considérer la reconstruction comme parfaite excepté en quelques endroits sur les arêtes du parallépipède délimitant la zone de mesure où se trouvent des points marginaux qui induisent de légères «erreurs».

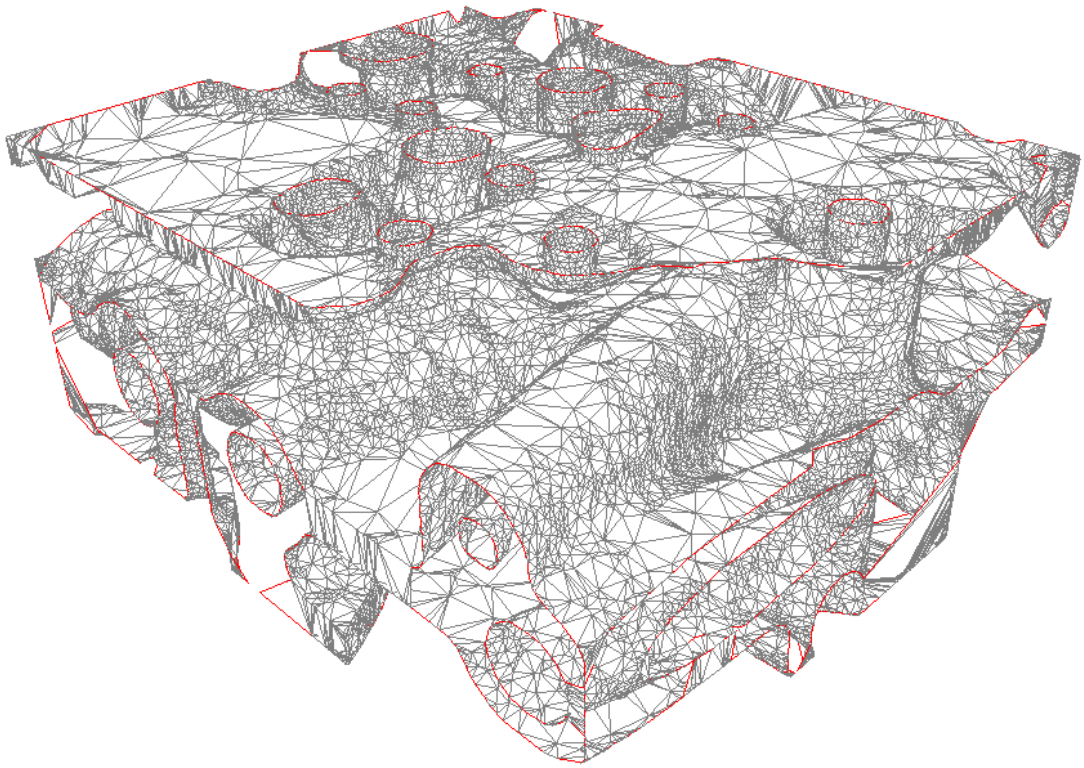


FIG. 3.24 – Résultat : échantillonnage de la pièce usinée, modèle *Tomo*.

Figure 3.24. Cette figure tente d'illustrer les variations de la densité d'échantillonnage pour la pièce précédente 3.23, ceci afin de mieux se représenter les difficultés posées par ce modèle.

Chapitre 4

Reconstruction à partir de sections

Sommaire

§4.1 *Introduction*

4.1.1 *De la spécificité des enjeux du cas de sections*

L'objet de ce chapitre est d'examiner une autre déclinaison de notre problème. Considérons, cette fois, que nos données possèdent une certaine structure. En effet, de nombreux procédés de mesure fournissent des échantillons de points organisés. Dans le cas d'un échantillon suffisamment dense, cette information supplémentaire n'est pas toujours indispensable, et on peut alors appliquer une méthode de reconstruction qui ne présume rien sur les données (par exemple, la méthode développée au chapitre 3) ; cependant, nombreux aussi sont les cas où on ne peut la négliger. En effet, la connaissance de cette organisation permet, d'une certaine manière, de relaxer les exigences de densité sur l'échantillonnage des objets à reconstruire. En profitant de la structure des données, on peut guider la reconstruction et ainsi espérer aboutir dans des cas où aucune méthode générale ne peut fournir de réponse satisfaisante.

Nous étudierons ici, le cas où l'échantillon se compose d'une suite de sections planes dans lesquelles les points forment des contours fermés orientés. Dans chaque section, les contours représentent l'intersection de l'objet avec le plan de coupe, ils sont souvent représentés par des lignes polygonales fermées. La méthode que nous allons présenter dans ce chapitre considère des sections prises orthogonalement à un axe, disons z. Nous verrons aussi comment l'adapter au cas de sections d'un objet prises radialement autour d'un axe par un simple changement de système de coordonnées.

4.1.2 *Un champ d'application propre*

Plusieurs procédés de numérisation d'objets sont susceptibles de fournir des jeux de données de cette nature. Pour n'en citer que quelques uns, on peut, par exemple, évoquer les différentes techniques de tomographie (CT), l'imagerie par résonance magnétique (IRM), ainsi que la microscopie électronique, ou encore la découpe réelle des objets en sections. Ainsi, grâce à ces différentes méthodes et une fois chaque section traitée par des algorithmes de détection de contours, on dispose de données présentées sous la forme adéquate.

De fait, cette formulation du problème trouve un terrain d'application concret dans de nombreux domaines. On peut citer, entre autres, le domaine médical, où l'imagerie et la reconstruction prennent une importance croissante à des fins d'aide au diagnostic, de simulation d'interventions chirurgicales. De même la biologie cellulaire, jusqu'ici limitée à des vues de coupes issues de la microscopie électronique, peut profiter de méthodes de reconstruction de sections. Sans entrer dans le détail, en topographie pour la représentation de terrain, en sismologie par l'analyse de la propagation d'ondes sismiques, des données sont extraites sous la forme qui nous intéresse ici. Plus généralement, l'industrie, pour des applications de *reverse engineering* par exemple, est amenée à employer ces mêmes techniques de numérisation et se trouve ainsi confrontée à notre problème.

Notons de plus que l'organisation en sections prend une importance capitale, quand le nombre de sections est faible, ce qui pour des raisons concrètes de risque, ou encore de coût, peut être imposé de manière irrévocable. On peut songer à minimiser l'exposition au champ magnétique pour un patient à qui on fait subir un examen par IRM ; on peut aussi considérer le prix exorbitant de certains procédés de mesure, les limites physiques sur l'épaisseur minimum des sections si on coupe réellement l'objet en tranches, ou encore la difficulté d'extraire des sections...

Pour illustrer ce propos, citons un jeu de données célèbre se présentant sous la forme de sections, le projet d'envergure *visible human project* [oM86], maintenant arrivé à son terme, qui propose la numérisation complète d'un homme et d'une femme.

4.1.3 Positionnement parmi les travaux antérieurs

De nombreuses approches ont déjà été proposées pour répondre au problème consistant à reconstruire des données organisées en sections. Rappelons brièvement les deux grandes classes de méthodes, l'approche *surfacique* et l'approche *volumique*. Par ailleurs, une autre caractéristique des différentes méthodes est à apprécier, la mise en correspondance des contours. Pour une présentation des différentes méthodes existantes, on se référera à la section 1.4 du chapitre 1. Nous nous bornerons ici à énoncer les caractéristiques de notre approche en tentant de la situer dans le contexte des travaux antérieurs, ceci afin d'en permettre une meilleure compréhension et donc une utilisation plus efficace.

Tout d'abord, la méthode que nous allons décrire peut être à la fois considérée comme une méthode surfacique *et* volumique. On peut la situer dans la lignée des méthodes développées par J.-D. Boissonnat [Boi88], par B. Geiger [Gei95, Gei93], ou plus récemment par S. Nullans [Nul98] ou par S. W. Cheng et T. K. Dey [CD99]. En fait, d'un point de vue pratique, les calculs concrètement réalisés apparentent notre approche à une méthode surfacique. On ne manipule que des structures représentant la peau, la surface de l'objet. On ne fournira donc pas un maillage de l'intérieur de l'objet comme d'autres méthodes. Cependant, ces opérations ayant pour but de représenter un certain volume défini *implicitement* par une *fonction d'interpolation*, notre approche s'interprète aisément en tant que méthode volumique. Elle reprend, sous cet aspect, des idées développées par J.-D. Boissonnat et F. Cazals dans [BC00] dans le cadre de la reconstruction d'objets dans l'espace à partir de nuages de points.

Ainsi, on pourra profiter de la rapidité de calcul des méthodes surfaciques *et* aussi de la fiabilité théorique des méthodes volumiques en ce qui concerne les changements complexes de topologie. La mise en correspondance des contours obéit à une règle simple[25], fondée sur l'existence d'un *recouvrement* de l'espace intérieur à ces contours dans un plan projeté le long de l'axe z. Cette règle de mise en correspondance est relativement classique dans la résolution du problème de l'interpolation de sections [ZJH87], surtout en ce qui concerne les méthodes capables de gérer des sections où coexistent plusieurs contours. La principale contrepartie à ce critère de mise en correspondance est que l'axe le long duquel sont prises les sections se doit d'être adapté à l'objet, faute de quoi il sera nécessaire de disposer d'une densité de sections plus grande.

Dans le cas où les conditions de mise en correspondance sont satisfaites, notre méthode permet de fournir l'interpolation polyédrique d'une surface *lisse* entre les contours en correspondance de deux sections consécutives. Notons, par ailleurs, que l'espacement entre les sections,*i.e.* la densité des sections le long de l'axe z, peut varier et que l'échantillonnage dans chaque section peut être non-homogène. De surcroît, afin de pouvoir traiter le cas de sections très (ou même trop) détaillées, *i.e.* où les contours sont décrits par un nombre démesuré d'arêtes, la méthode permet d'incorporer un prétraitement, dit de *simplification* des données, ayant pour but d'extraire les points significatifs pour chaque contour, et ainsi d'améliorer les temps de reconstruction pour un résultat comparable.

[25] Simple, dans le sens où l'on connaît la nature du résultat de la reconstruction avant d'exécuter la méthode.

§4.2 Au sujet des voisins naturels

4.2.1 Propriétés utilisées

Revenons, tout d'abord, sur les possibilités offertes par les *coordonnées naturelles* (cf. section 2.5 du chapitre 2) en ce qui concerne l'interpolation. L'enjeu principal de la méthode présentée dans ce chapitre est de fournir une interpolation lisse entre les sections successives d'un objet. Exposons, plus précisément, en quoi le système de coordonnées barycentriques défini grâce aux voisins naturels est parfaitement adapté à nos attentes. Comparé aux autres systèmes présentant des propriétés comparables, tels les *coordonnées de* LAPLACE, il possède l'avantage de proposer un compromis acceptable entre un calcul rapide et des propriétés de dérivabilité (requises pour une interpolation lisse) suffisantes. Cependant, on peut songer à employer d'autres schémas d'interpolations qui posséderaient des propriétés de dérivabilité équivalentes [Bro97] ou même supérieures [HS00]. Les seules propriétés réellement attendues pour définir la fonction d'interpolation qui est à la base de notre méthode sont celles énoncées par la propriété 2.4. Rappelons en ici l'essentiel, en reprenant les notations de la section 2.5. On notera donc $\mathcal{S} = \{p_i\}_{i=1\cdots n}$, un échantillon de points et $\{\lambda_i\}_{i=1\cdots n}$ le système des fonctions coordonnées naturelles associé. On rappelle alors les propriétés suivantes :

* *la localité*,*i.e.* $\lambda_i(p) = 0$ si p_i n'est pas voisin naturel de p ;
* les fonctions λ_i sont $\mathscr{C}^0$ partout et $\mathscr{C}^1$ sauf aux points de l'échantillon $\mathcal{S}$;
* la propriété du *barycentre* : $p = \sum_{p_i \in \mathcal{S}} \lambda_i(p).p_i$.

4.2.2 Hors de l'enveloppe convexe

Pour l'usage que nous voulons en faire, le problème se pose de savoir comment étendre la définition des coordonnées naturelles dans le plan *en dehors de l'enveloppe convexe*. En effet, même dans un cas simple, pour l'interpolation de deux contours contenus dans deux sections consécutives, l'essentiel des requêtes sur les voisins naturels s'effectuera en dehors de l'enveloppe convexe d'au moins un des contours. Pour remédier à ce problème, il existe des réponses théoriques, on pourra se référer à [Bro97], où est décrit une extension possible dans le plan pour les voisins naturels en dehors de l'enveloppe convexe. Pour être plus précis, les auteurs présentent un formalisme afin de définir des systèmes de coordonnées barycentriques dans le plan. L'idée est de se servir des coordonnées barycentriques classiques définies pour les sommets d'un triangle comme fonctions génératrices d'autres systèmes de coordonnées barycentriques plus complexes.

Considérons la triangulation de DELAUNAY d'un ensemble de points $\{p_i\}_{i=1\cdots n}$. Il s'agit de définir un système de coordonnées barycentriques pour un point x dans le plan. Pour ceci, définissons $b_i^k(x)$ égal à la coordonnée barycentrique de x par rapport au point p_i dans le triangle d'indice k si p_i est un sommet de ce triangle et égal à zéro dans le cas contraire. Considérons, de plus, une famille de fonctions $\{w^k\}_{k=1\cdots m}$ associée à chaque triangle indicé k de la triangulation, qui seront positives, $\mathscr{C}^1$ et nulles exactement en dehors du disque circonscrit au triangle d'indice k. Considérons alors, pour $i \in [1, n]$, les fonctions s_i définies ainsi :

$$s_i = \frac{\sum_{k=1}^{m} b_i^k(x) w^k(x)}{\sum_{k=1}^{m} w^k(x)}$$

La famille de fonctions $\{s_i\}_{i=1\cdots n}$ est un *système de coordonnées barycentriques*. Ce système de coordonnées est défini sur l'union des disques vides de DELAUNAY et donc en dehors de l'enveloppe convexe.

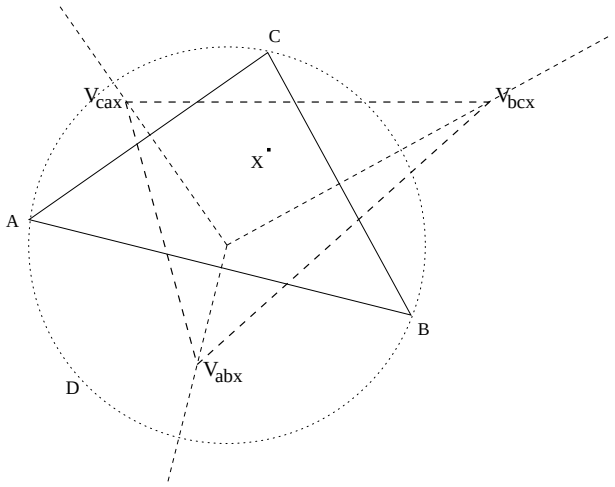


FIG. 4.1 – Illustration : définition d'une fonction w_k pour le triangle, ABC, orienté positivement : $w_k(x) = s \times \text{aire}_{\text{triangle}}(V_{abx}, V_{bcx}, V_{cax})$ avec $s = 1$ si $x \in D$ et $s = -1$ sinon.

Par ailleurs, l'article mentionne un jeu de fonctions $\{w^k\}_{k=1\cdots m}$, pour lequel le système de coordonnées $\{s_i\}_{i=1\cdots n}$ coïncide exactement avec les coordonnées naturelles à l'intérieur de l'enveloppe convexe des points p_i. Il suffit de prendre pour $w_k(x)$ la fonction calculant l'aire signée du triangle dont les sommets sont les centres des cercles circonscrits au triangle formé par x et les arêtes du triangle d'indice k prise une à une (figure 4.1). Bien que n'étant pas nulle en dehors du disque circonscrit au triangle indicé k, cette famille de fonctions particulières convient et permet de définir un système de coordonnées barycentriques $\{s_i\}_{i=1\cdots n}$ qui coïncide précisément avec les coordonnées naturelles à l'intérieur de l'enveloppe convexe.

Cependant, même ce prolongement en dehors de l'enveloppe convexe ne peut suffire pour notre application. En effet, nombreux sont les cas où des sections consécutives sont très différentes (par exemple, l'apparition d'un nouveau contour) et où l'on sera amené à effectuer des requêtes en dehors de l'union des disques vides de DELAUNAY dans au moins une des deux sections à interpoler. Ainsi, nous nous sommes résignés à construire une *boîte englobante* pour nos données, afin de nous ramener à des calculs à l'intérieur de l'enveloppe convexe. Les coefficients naturels associés aux points définissant cette boîte englobante ne seront pas considérés dans la définition de nos fonctions d'interpolation.

§4.3 Interpolation de sections grâce aux voisins naturels

4.3.1 Description qualitative

Notre objectif est de fournir une interpolation lisse par morceau des données, c'est à dire la représentation d'une surface lisse du bord de l'objet entre chaque section. Cette approche s'apparente à celle présentée par J.-D. Boissonnat et F. Cazals dans [BC00]. Nous allons, en effet, utiliser une fonction d'interpolation dont il s'agira de représenter le niveau zéro. Cependant, pour tirer parti de l'organisation des données en sections, la fonction utilisée est évidemment adaptée. Celle que nous proposons offre, tout d'abord, l'avantage de se calculer à partir des coordonnées naturelles en dimension 2. De plus, elle guide la reconstruction afin de répondre dans le cas spécifique des sections où la méthode générale décrite dans [BC00] échouerait, par exemple, pour des sections très écartées.

Plus précisément, la particularité de notre méthode réside dans la définition *pour chaque section* d'une fonction $\mathscr{C}^1$ presque partout de $\mathbb{R}^2$ dans $\mathbb{R}$ grâce aux voisins naturels en dimension 2, dont la courbe de niveau zéro fournit une reconstruction *lisse* des contours contenus dans cette section. En outre, grâce à cette fonction associée à chaque section, on peut le cas échéant procéder à une étape de *simplification* des données (cf. 4.3.5), qui ne conservera que les points «utiles» à la définition du niveau zéro de la fonction à une certaine précision.

Entre deux sections successives, une *interpolation linéaire* des fonctions associées à chacune des deux sections fournit alors une fonction $\mathscr{C}^1$ de $\mathbb{R}^3$ dans $\mathbb{R}$. La surface de niveau zéro de cette fonction d'interpolation est une variété de $\mathbb{R}^3$, elle aussi, $\mathscr{C}^1$ qui interpole les deux sections. Notons, de plus, que ces fonctions sont définies simplement par le fait de construire une triangulation de Delaunay dans chaque section grâce aux coordonnées naturelles. Leur calcul, comme le montre l'étude réalisé section 2.5.7, ne sera donc pas excessivement coûteux.

Pour chaque tranche comprise entre deux sections consécutives, nous allons représenter la surface de niveau zéro de cette fonction d'interpolation par une surface triangulée. Là encore, nous utiliserons exclusivement une structure de dimension 2, dont le relèvement en dimension 3 nous donnera le résultat escompté. La structure utilisée est une triangulation de Delaunay *contrainte* planaire que l'on pourra raffiner dans le but de représenter le niveau zéro de la fonction d'interpolation et obtenir une représentation de la transition lisse souhaitée entre chaque section.

4.3.2 Notations

On considère un ensemble de sections planes $\{\mathcal{S}_i\}_{i=1\cdots m}$ prises orthogonalement à un axe z sur un objet à reconstruire $\mathcal{O}$. Considérons, pour fixer les idées, les sections indicées par hauteur croissante. Par ailleurs, on peut considérer que les sommets des contours polygonaux contenus dans la section $\mathcal{S}_i$ sont représentés en coordonnées cartésiennes par des triplets (x, y, z_i), où z_i est la hauteur fixe de la section d'indice i.

Pour chaque contour polygonal orienté, on associe à chacun des sommets le vecteur unitaire de la bissectrice des deux arêtes adjacentes à ce sommet dans le plan de la section $\mathcal{S}_i$. On pourra donc considérer par la suite que chaque section $\mathcal{S}_i$ est composée de n_i points p_j munis de leur normales $\vec{n}_j$.

4.3.3 Définition de l'objet interpolant

Comme précédemment évoqué, on considère que le résultat de l'interpolation est décrit par un objet mathématique, la variété de $\mathbb{R}^3$ définie comme le niveau zéro d'une fonction continue H, $\mathscr{C}^1$ presque partout. Cette fonction $H : \mathbb{R}^3 \to \mathbb{R}$ est définie par morceaux, pour i variant de 1 à m :

On considère la *tranche* définie par deux sections consécutives $\mathcal{S}_i$ et $\mathcal{S}_{i+1}$.

Définition 4.1 *Dans chacune des sections, on définit, à l'aide des voisins naturels, une* fonction d'interpolation 2D *de* $\mathbb{R}^2$ *dans* $\mathbb{R}$ *:*

$$\mathrm{h}_i(p) = \sum_{p_j \in \mathcal{S}_i} \lambda_\mathrm{j}(p)\mathrm{d}_j(p)$$

où $\mathrm{d}_j(p) = (p_j - p).\vec{n}_j$ *est la distance signée à la tangente en* p_j*.*

On notera que cette fonction est $\mathscr{C}^0$ partout et $\mathscr{C}^1$ sauf aux points de l'échantillon, comme les fonctions λ_j : de plus, elle vaut zéro en chaque point de l'échantillon. Les fonctions h_i sont définies implicitement simplement en construisant la triangulation de Delaunay des points contenus dans la sections d'indice i. Notons, de plus, que l'on peut insérer chaque point dans Delaunay à partir de son prédécesseur sur le contour orienté auquel il appartient, ainsi le coût de la localisation du point est quasiment constant.

On définit alors H sur cette tranche comme l'interpolation linéaire en z entre les fonctions associées aux sections $\mathcal{S}_i$ et $\mathcal{S}_{i+1}$. Pour tout point $P \in \mathbb{R}^3$ tel que $P(x, y, z)$, on note P_{xy} le point 2D *projeté* le long de l'axe z.

Définition 4.2 *On définit alors la* fonction d'interpolation 3D *de* $\mathbb{R}^3$ *dans* $\mathbb{R}$ *par morceaux. Pour tout point* P *de coordonnées* $(x, y, z) \in \mathbb{R}^2 \times [z_i, z_{i+1}]$ *:*

$$\mathrm{H}(P) = \frac{(z_{i+1} - z)\mathrm{h}_i(P_{xy}) + (z - z_i)\mathrm{h}_{i+1}(P_{xy})}{z_{i+1} - z_i}$$

Cette définition est valide, puisque H coïncide avec h_i sur le plan de la section $\mathcal{S}_i$ et avec h_{i+1} sur le plan de la section $\mathcal{S}_{i+1}$. Ainsi, la fonction H définie par morceaux est même continue globalement. Elle est même $\mathscr{C}^1$ par morceaux sur l'intérieur de chaque tranche (*i.e.* excepté sur les plans de sections) et vaut zéro en chaque point de l'échantillon. L'objet mathématique que l'on considère comme interpolation de nos sections est la surface de niveau zéro $\mathrm{H}^{-1}(0)$. On peut aussi considérer le volume intérieur à cette surface de niveau zéro, défini comme l'ensemble $\{P \in \mathbb{R}^3 / \ \mathrm{H}(P) > 0\}$. Ainsi, on garantit la reconstruction d'une surface lisse délimitant un certain volume intérieur défini implicitement.

4.3.4 Propriétés et représentation

$\mathrm{H}^{-1}(0)$ est une variété de dimension 2, qui plus est lisse par morceaux, sur chaque tranche. En outre, on notera une propriété intéressante :

Propriété 4.1 *Pour chaque tranche, il existe au plus un zéro de* H *sur un segment vertical compris entre les deux sections.*

Preuve. En effet, entre les sections $\mathcal{S}_i$ et $\mathcal{S}_{i+1}$, l'équation $\mathrm{H}(P) = 0$ donne au plus une unique solution sur une verticale (x, y), à savoir le point $P(x, y, z)$ où :

$$z = \frac{z_{i+1}\mathrm{h}_i(P_{xy}) - z_i\mathrm{h}_{i+1}(P_{xy})}{\mathrm{h}_i(P_{xy}) - \mathrm{h}_{i+1}(P_{xy})}$$

La solution est, alors, considérée comme valide si et seulement si $z \in [z_i, z_{i+1}]$. Remarquons que c'est toujours le cas pour P_{xy} vérifie $\mathrm{h}_i(P_{xy})\mathrm{h}_{i+1}(P_{xy}) \leq 0$, étant donné que $z_i < z_{i+1}$. □
On peut alors établir le corollaire suivant :

Corollaire 4.2 *Il existe exactement un zéro de* H *sur un segment vertical donné,* $\{P_{xy}\} \times [z_i, z_{i+1}] \subset \mathbb{R}^3$*, si et seulement si l'inégalité* $\mathrm{h}_i(P_{xy})\mathrm{h}_{i+1}(P_{xy}) \leq 0$ *est vérifiée.*

Preuve. Pour conclure, il ne reste qu'à écrire $\mathrm{H}(x, y, z_i) = \mathrm{h}_i(x, y)$ et $\mathrm{H}(x, y, z_{i+1}) = \mathrm{h}_{i+1}(x, y)$. Ainsi, si on a $\mathrm{h}_i(x, y)\mathrm{h}_{i+1}(x, y) \leq 0$, on conclut en appliquant le *théorème des valeurs intermédiaires* sur H avec $\mathrm{H}(x, y, z_i)\mathrm{H}(x, y, z_{i+1}) \leq 0$, d'où l'existence d'au moins un zéro de H. □

Cette propriété est particulièrement intéressante pour la représentation de l'objet d'interpolation $\mathrm{H}^{-1}(0)$. En effet, on s'aperçoit que sur chaque tranche prise une à une, on peut définir la fonction de $\mathbb{R}^2$ dans $\mathbb{R}$, qui à $P_{xy}(x, y)$ associe une hauteur z telle que l'on ait un nouveau point sur le niveau zéro de H, *i.e.* $\mathrm{H}(P) = 0$ pour $P(x, y, z)$. Ainsi, la surface polyédrique destinée à fournir une représentation du niveau zéro de H, pourra être considérée comme un *terrain* entre deux sections consécutives. On pourra donc représenter et effectuer les opérations de raffinement de la représentation en dimension 2 pour chaque tranche de l'objet. La structure employée sera une triangulation de *terrain* (à chaque point 2D est associée une hauteur) de DELAUNAY contrainte.

4.3.5 *Simplification des données*

Présentation

Ce qui suit a pour but de présenter un prétraitement possible dans le cas où les sections sont sur-échantillonnées. En effet, l'organisation des données en sections, se traduit généralement par un espacement entre chaque section relativement important par rapport au pas d'échantillonnage à l'intérieur des sections. En effet, comme évoqué dans l'introduction de ce chapitre, nombreux sont les cas où le pas d'acquisition des sections ne peut être abaissé suffisamment, quand il n'est pas purement et simplement fixé. C'est d'ailleurs une raison fondamentale du besoin de méthodes particularisées au cas des sections.

Par contre, les progrès des matériels de numérisation et des méthodes d'extraction de contours permettent, aujourd'hui, d'acquérir des représentations toujours plus fines des contours dans chacune des sections. Naïvement, on peut même être tenté de compenser la faiblesse du pas d'échantillonnage entre les sections par des contours très finement échantillonnés. Cette stratégie est à l'origine de divers désagréments. Elle ne permet, en aucun cas, une reconstruction meilleure, l'information apportée par ce sur-échantillonnage des sections étant, en fait, redondante. Le volume de données étant alors plus important, les différentes méthodes mettent évidemment plus de temps à fournir des résultats, sans réel espoir de les voir s'améliorer. De plus, pour des méthodes sensibles à la non-homogénéité de l'échantillonnage, les résultats obtenus ont toutes les chances de se dégrader.

Algorithme de simplification

Pour pouvoir prendre en compte ce problème, nous présentons ici, une méthode permettant de simplifier les données qui vont servir à définir nos fonctions d'interpolation. En fait, il s'agit, pour chaque section, de déterminer les points nécessaires pour définir une fonction, h_i^t, dont l'évaluation en chaque sommet de la section serait quasiment nulle (disons à ε_s près). On pourra alors considérer les contours originaux comme une représentation acceptable du niveau zéro de notre fonction partielle et se servir de ces fonctions partielles pour définir la fonction H.

Nous proposons ici une adaptation, au cas des sections, de la méthode décrite dans [BC01]. Pour cela, considérons, pour initialiser le processus, une partie $\mathcal{P}_i$ de l'ensemble des sommets de la section $\mathcal{S}_i$, sur laquelle on construit une triangulation de DELAUNAY $\mathfrak{D}$. Cette triangulation définit alors implicitement une fonction d'interpolation initiale, notée h_i^t. Présentons, maintenant, la boucle permettant de sélectionner les points nécessaires pour garantir que l'ensemble des sommets contenus dans la section $\mathcal{S}_i$ est sur le niveau zéro de la fonction h_i^t à ε_s près.

Algorithme de simplification de données

Require: $\mathcal{P}_i$ une partie de l'ensemble des sommets de la section $\mathcal{S}_i$.
Ensure: $\mathfrak{D}$ une triangulation de DELAUNAY, définissant implicitement h_i^t.

```
soit 𝔇 la triangulation de DELAUNAY des points de 𝒫_i ;
soit ℰ ⇐ 𝒮_i \ 𝒫_i l'ensemble des sommets restant ;
repeat
  soit p un itérateur sur ℰ ;
  repeat-test ⇐ false ;
  while (ℰ ≠ ∅) and (p ≠ past-the-end) do
    if h_i^t(p) > ε_s then
      insérer p dans la triangulation 𝔇 ;
      ℰ ⇐ ℰ \ {p} ;
      repeat-test ⇐ true ;
    end if
    p ⇐ successeur de p dans ℰ ;
  end while
until repeat-test
return 𝔇 ;
```

Après l'exécution de l'algorithme de simplification de données, on peut considérer que les points restant éventuellement dans l'ensemble $\mathcal{E}$ sont évalués à zéro pour la fonction h_i^t à un niveau de précision ε_s . L'idée de cette approche est légèrement différente de celle détaillée dans [BC01]. En fait, dans le cas de sections, nous disposons déjà des contours originaux comme représentation de l'objet dans chaque section. Ainsi, si on décide d'opérer cette étape de simplification des données, on considère, en fait, déjà que les contours originaux sont, eux même, une représentation suffisamment fine de notre objet.

On ne cherchera donc pas à raffiner les contours dans chaque section et on introduira les contours tels quels dans la triangulation contrainte chargée de représenter la reconstruction. L'*erreur de corde* par rapport aux données sera donc *nulle*. On fournira juste une interpolation entre les sections grâce aux fonctions h_i^t définies sur une fraction des points, au lieu de la totalité, ce qui améliorera les performances.

4.3.6 *Description de l'algorithme*

Introduction

La méthode que nous allons développer permet de fournir une représentation raffinée d'une surface lisse entre chaque section. Alors que nous avons vu comment sélectionner parmi des données trop riches celles qui sont pertinentes, il est, malgré tout, aussi fréquent de ne disposer que de contours polygonaux de mauvaise qualité. On pourra donc chercher à obtenir une représentation de ces contours plus lisse en approximant dans chaque section le niveau zéro des fonctions h_i, avant d'interpoler les sections entre elles. Pour ce faire, notons ε_1 la précision attendue pour la représentation des contours dans chaque section et ε_2 la précision de représentation souhaitée entre deux sections consécutives.

Algorithme de raffinement de contours

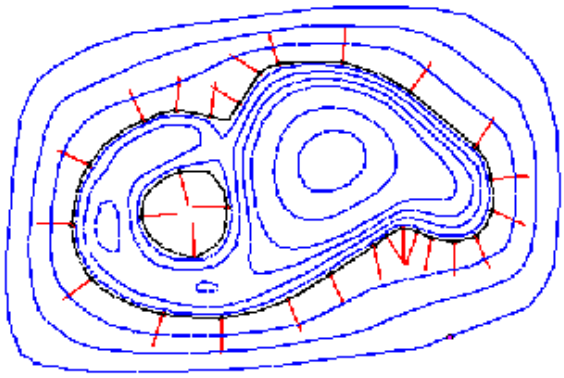


FIG. 4.2 – Illustration : différents niveaux, dont le niveau zéro en noir, d'une fonction d'interpolation h_i.

À l'extrême inverse du cas présenté dans le cadre de la simplification de données, on peut songer à la situation fréquente, où les contours polygonaux contenus dans les sections sont très anguleux et ne comportent que peu de sommets. Dans le but d'obtenir une interpolation lisse entre les sections, il est alors nécessaire de lisser aussi ces contours. Il s'agit donc de fournir une représentation polygonale, à ε_1 près, de la courbe $\mathrm{h}_i^{-1}(0)$. On disposera ainsi pour la suite d'un ensemble de contraintes $\mathfrak{C}_i$, représentation polygonale du niveau zéro dans la section d'indice i, plus «lisse» que les contours originaux. L'algorithme employé est la transcription en dimension 2 de celui décrit dans [BC00] pour les nuages de points dans l'espace. On s'intéressera ici surtout aux possibilités de raffinement de la représentation, plus qu'à l'aspect reconstruction, puisqu'on pourra prendre autant de points que nécessaire sur les contours polygonaux afin que soient vérifiées les conditions d'échantillonnage requises par la méthode.
Donnons, tout d'abord, quelques définitions afin de clarifier la présentation de l'algorithme :

* On dit d'une arête e de la triangulation de DELAUNAY qu'elle est *bipolaire* si l'évaluation de h_i sur son arête duale n'est pas de signe constant ; en particulier, si h_i prend des signes opposés aux extrémités de l'arête duale de e.

* On appelle *raffiner*, l'opération consistant pour une arête bipolaire e à calculer un zéro de h_i sur son arête duale (par dichotomie pondérée, par exemple) et à l'insérer dans la triangulation de DELAUNAY $\mathfrak{R}$ support au raffinement.
* Par ailleurs, les arêtes bipolaires se voient attribuer une *note* égale à l'évaluation de h_i en leur milieu.

De plus, on peut considérer qu'on initialise $\mathfrak{R}$ en rajoutant autant de points que nécessaire sur les contours d'origine pour que les arêtes *bipolaires* de $\mathfrak{R}$ au départ représentent exactement les contours d'origine.

Algorithme de reconstruction de courbe

Require: $\mathcal{S}_i$ l'échantillon de points et h_i une fonction dont le niveau zéro est à interpoler.
Ensure: un ensemble d'arêtes $\mathfrak{C}_i$, représentation polygonale de $h_i^{-1}(0)$ à ε_1 près.
  soit $\mathfrak{R}$ la triangulation de DELAUNAY utilisée pour raffiner la représentation ;
  soit $\mathfrak{P}_1$ une queue de priorité ;
  insérer toutes les arêtes *bipolaires* pour h_i selon leur *note* dans $\mathfrak{P}_1$;
  while $\mathfrak{P}_1 \neq \emptyset$ **do**
    extraire e l'arête possédant la plus *mauvaise note* de $\mathfrak{P}_1$;
    raffiner e et mettre à jour $\mathfrak{R}$;
    calculer les notes des deux demi-arêtes bipolaires issues du raffinement de e dans $\mathfrak{R}$;
    insérer dans $\mathfrak{P}_1$ les demi-arêtes si leur note est supérieure à ε_1 ;
  end while
  return l'ensemble des arêtes *bipolaires* de $\mathfrak{R}$;

Le résultat obtenu est alors une représentation linéaire par morceaux de la courbe de niveau zéro de h_i, donc de H dans la section $\mathcal{S}_i$, à ε_1 près.

Algorithme principal

Que l'on ait simplifié les données contenues dans chaque section ou qu'au contraire on en ait construit une représentation raffinée, il s'agit maintenant de fournir une interpolation entre les sections. L'algorithme suivant reconstruit l'objet tranche par tranche en considérant les sections deux à deux. Pour représenter la tranche de l'objet comprise entre les deux sections consécutives d'indice i et $i+1$, on utilise une triangulation de DELAUNAY contrainte, notée $\mathfrak{CD}_{i+1}$. Ces triangulations $\mathfrak{CD}_i$ sont, en fait, des triangulations de *terrain*, dont le calcul se fait en 2D et qui peuvent s'interpréter comme un terrain en 3D, où chaque point est simplement relevé à une hauteur z. Précisons, par ailleurs, quelques éléments utiles pour la présentation de la méthode :

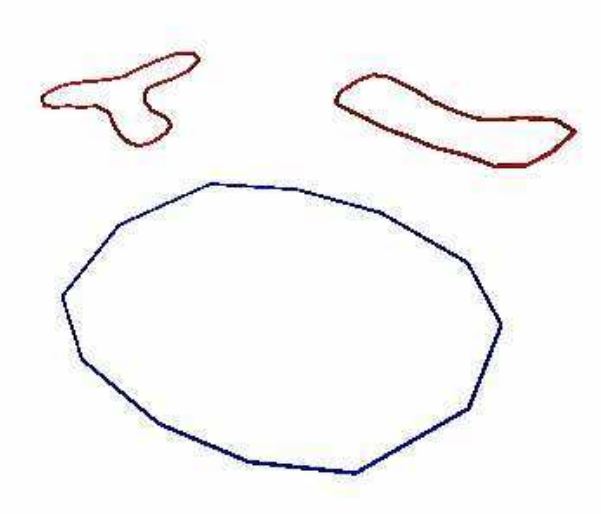
* Par abus de langage, on dira que l'*arête duale* d'un triangle d'une triangulation $\mathfrak{CD}_i$ est le segment vertical passant par le centre du cercle circonscrit en 3D de ce triangle compris entre les deux sections courantes.
* On dira d'une arête d'une triangulation $\mathfrak{CD}_i$ qu'elle est de GABRIEL si sa sphère diamétrale est vide de points en 3D.
* On nommera *conformer*, l'opération consistant à insérer dans $\mathfrak{CD}_i$ le milieu d'une arête e qui n'est pas de GABRIEL.
* Dans une triangulation de DELAUNAY contrainte $\mathfrak{CD}_i$, on dira d'un triangle qu'il est *bipolaire* pour la fonction H si et seulement si l'évaluation de H sur son *arête duale* n'est pas de signe constant, *i.e.* quand les signes des évaluations de H en chaque extrémité de l'arête duale sont opposés. On peut alors remarquer, d'après le corollaire 4.2, qu'un

triangle est *bipolaire* pour la fonction H entre les sections $\mathcal{S}_i$ et $\mathcal{S}_{i+1}$, si et seulement si $\mathrm{h}_i(C_{xy})\mathrm{h}_{i+1}(C_{xy}) < 0$, avec C_{xy} est la projection de C, centre du cercle circonscrit au triangle en dimension 3, le long de l'axe z.

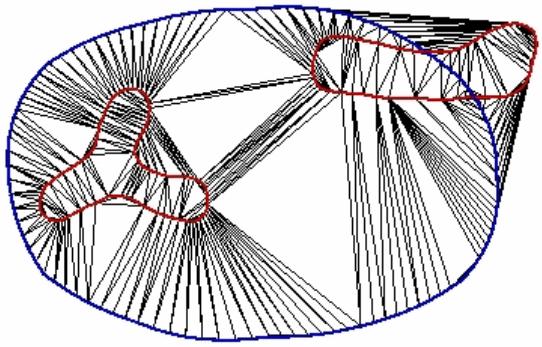
* On attribue à chaque triangle bipolaire une *note* égale à l'évaluation de la fonction H en C, centre du cercle circonscrit au triangle en dimension 3.
* On nommera *raffiner-conformer*, l'opération consistant pour un triangle bipolaire à calculer un zéro de H sur son arête duale (*i.e.* calculer la hauteur correspondant au zéro de H pour la verticale passant par C_{xy}, cf. section 4.3.4), puis à insérer ce point dans la triangulation $\mathfrak{CD}_i$ et à *conformer* les contraintes avec lesquelles il est en conflit (*i.e.* les contraintes qui ne sont plus de Gabriel du fait de ce nouveau point).
* Enfin, on note *simplification-opt* un booléen, permettant d'appliquer la simplification de données (cf. 4.3.5) ou non.

Algorithme d'interpolation de sections

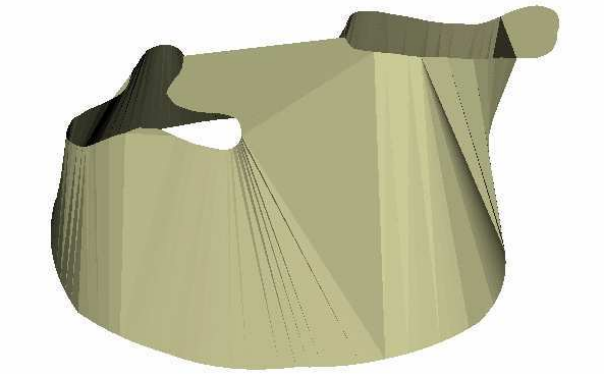
soit $\mathfrak{D}$ et $\mathfrak{D}'$ deux triangulations de Delaunay ; {début de l'initialisation}
if simplification-opt **then**
  $\mathfrak{D} \Leftarrow$ *simplification* de $\mathcal{S}_1$; {$\mathfrak{D}$ définit h_1 à ε_s près}
  $\mathfrak{C} \Leftarrow$ contour original ;
else
  $\mathfrak{D} \Leftarrow$ triangulation de Delaunay de $\mathcal{S}_1$, qui définit implicitement h_1 ;
  $\mathfrak{C} \Leftarrow$ *raffinement* du contour $\mathrm{h}_1^{-1}(0)$ à ε_1 près ;
end if{fin de l'initialisation}
for $i = 2 \cdots m$ **do**
  if simplification-opt **then**
    $\mathfrak{D}' \Leftarrow$ *simplification* de $\mathcal{S}_i$; {$\mathfrak{D}'$ définit h_i à ε_s près}
    $\mathfrak{C}' \Leftarrow$ contour original ;
  else
    $\mathfrak{D}' \Leftarrow$ triangulation de Delaunay de $\mathcal{S}_i$, qui définit implicitement h_i ;
    $\mathfrak{C}' \Leftarrow$ *raffinement* du contour $\mathrm{h}_i^{-1}(0)$ à ε_1 près ;
  end if{notons qu'à ce stade, la fonction $\mathrm{H}_{i-1} = \mathrm{H}_{|\mathbb{R}^2 \times [z_{i-1}, z_i[}$ est déjà définie}
  soit $\mathfrak{CD}_i$ une triangulation de Delaunay contrainte ; {représentation de la tranche entre $\mathcal{S}_{i-1}$ et $\mathcal{S}_i$}
  insérer $\mathfrak{C}$ et $\mathfrak{C}'$ dans $\mathfrak{CD}_i$;
  conformer toutes les contraintes de $\mathfrak{CD}_i$;
  soit $\mathfrak{P}_2$ une queue de priorité ;
  insérer tous les triangles suivant leur *note* pour H_{i-1} dans $\mathfrak{P}_2$;
  while $\mathfrak{P}_2 \neq \emptyset$ **do**
    extraire le triangle t possédant la plus *mauvaise note* de $\mathfrak{P}_2$;
    raffiner-conformer t et mettre à jour $\mathfrak{CD}_i$;
    calculer les *notes* des nouveaux triangles bipolaires de $\mathfrak{CD}_i$;
    insérer les triangles dont la note est supérieure à ε_2 dans $\mathfrak{P}_2$;
  end while
  $\mathcal{B}_i \Leftarrow$ ensemble des triangles *bipolaires* de $\mathfrak{CD}_i$; {représentation de l'objet $\mathrm{H}_{i-1}^{-1}(0)$}
  $\mathfrak{D} \Leftarrow \mathfrak{D}'$;
  $\mathfrak{C} \Leftarrow \mathfrak{C}'$;
end for
return $\bigcup_{i=1}^{m} \mathcal{B}_i$;



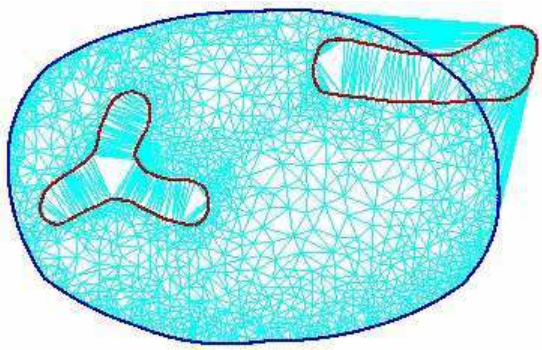
exemple de données organisées en sections



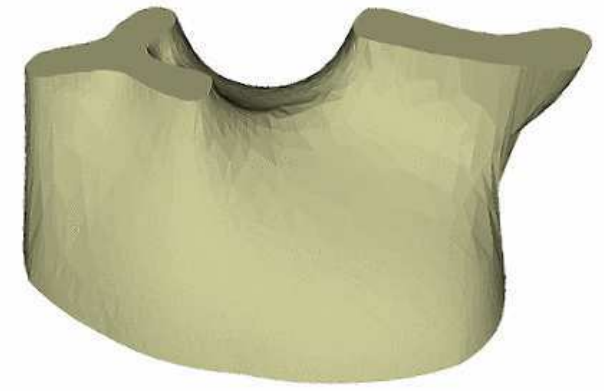
superposition des contours dans une triangulation de DELAUNAY contrainte



extraction des triangles bipolaires, formant le bord initial de notre objet



la triangulation de DELAUNAY contrainte après raffinement-conformation



relèvement 3D de la représentation de la surface interpolante

FIG. 4.3 – Illustration : différentes étapes de l'interpolation de sections.

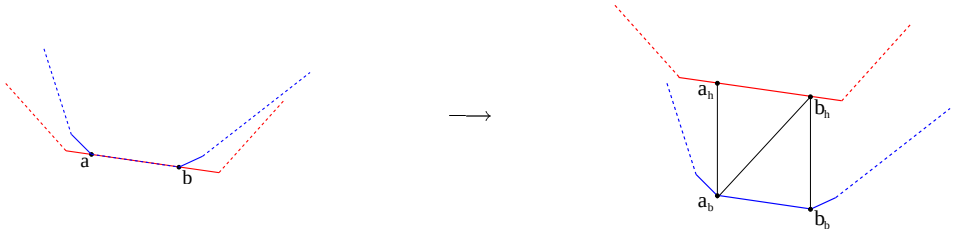
Pour mieux se représenter le fonctionnement de l'algorithme, on pourra se référer à la figure 4.3, où sont représentées les étapes essentielles du calcul sur un exemple synthétique, représentatif des possibilités de notre méthode. La représentation à ε_2 près de l'objet lisse $\mathrm{H}^{-1}(0)$ est donnée par l'ensemble des triangles bipolaires des triangulations contraintes $\{\mathfrak{CD}_i\}_{i=1\cdots m}$. Notons aussi que la méthode permet de reconstruire en ligne un modèle, en même temps que sont récupérées les sections une à une.

4.3.7 *Le relèvement en dimension 3*

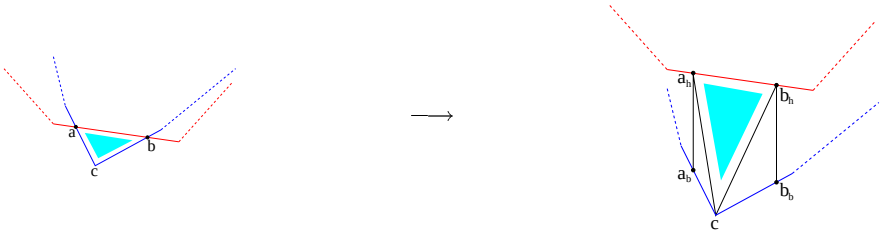
Décrivons avec précision comment extraire des triangulations contraintes 2D, $\mathfrak{CD}_i$, le bord 3D de notre reconstruction, $\mathcal{B}_i$. Chaque facette de la triangulation correspond évidemment à une facette de notre bord en interprétant ses sommets comme des points 3D. En fait, la seule difficulté est d'interpréter, en dimension 3, ce qui survient autour des points intersections des contours qui représentent l'objet dans les sections successives. On peut, en effet, considérer ces points comme *doubles* ; ils représentent, en fait, un point sur le contour du haut et un autre sur celui du bas, liés par une arête dans leur représentation en dimension 3. Deux configurations sont à distinguer : le cas où il y a deux intersections liées par une arête et le cas où le point intersection n'est lié dans la triangulation contrainte qu'à des points *simples* (*i.e.* non doubles).

Cas de deux points doubles liés par une arête

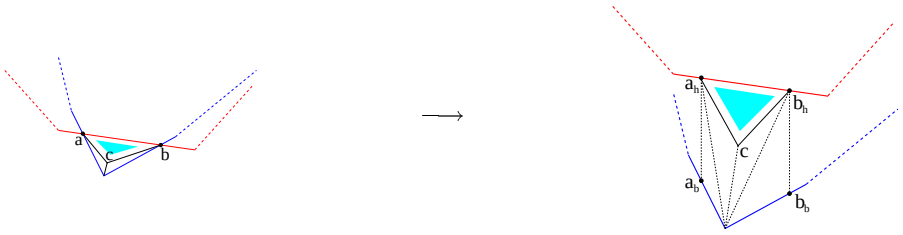
* Soit, il n'y a aucune facette sélectionnée adjacente à cette arête et on peut couper le quadrilatère formé par les deux paires de points par une diagonale dans la représentation du bord en dimension 3 ;



* Soit, la facette sélectionnée adjacente à cette arête possède son troisième sommet sur le contour du bas et on peut relever une facette liant ce point et les deux versions hautes des points doubles, ainsi que deux autres facettes reliant ce point avec les versions hautes et basses de chacun des points doubles ;



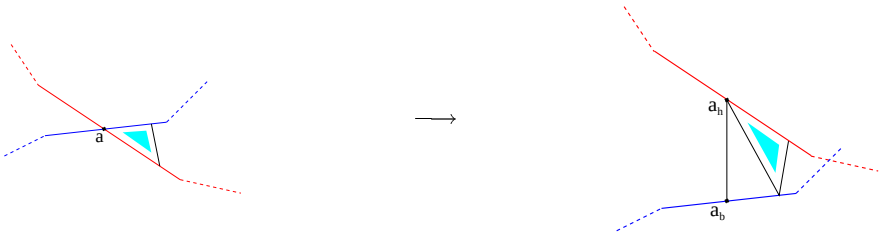
* Enfin, si le troisième sommet est entre les deux sections et on relève la facette liant ce point et les deux versions hautes des points doubles, puis les facettes adjacentes à chaque point double comme ce point n'était pas adjacent à un autre point double (cf. cas suivant).



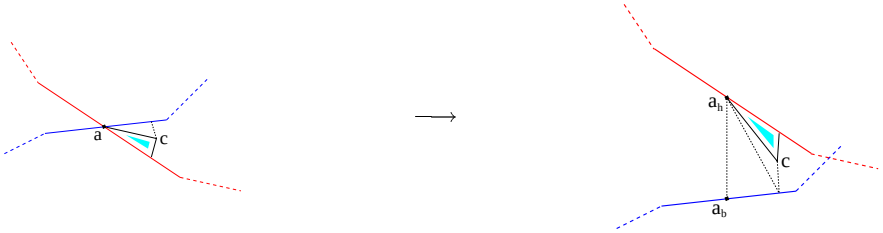
Cas d'un point double uniquement adjacent à des points simples

On considère chacune des facettes sélectionnées adjacentes à ce point séparément.

* Soit, la facette possède un sommet sur le contour du bas, et on relève, en plus de cette facette, une facette formée de ce point et des deux versions, haute et basse, du point double ;



* Soit, la facette ne possède pas de sommet sur le contour du bas et on relève alors uniquement cette facette (cas normal).



§4.4 Détails sur l'implantation réalisée

4.4.1 Commentaires sur certains choix

L'algorithme proposé a été implanté à l'aide de la librairie CGAL (cf. 2.1). Notons que la méthode combine différentes structures sophistiquées de la Géométrie algorithmique et que l'implantation a été grandement facilitée par la généricité de la librairie CGAL.

Pratiquement, l'implantation a été réalisée en C++ . On a cherché à préserver des possibilités de substitution à différents stades dans la méthode, ceci afin de permettre l'emploi d'autres fonctions d'interpolation tout en profitant des méthodes de raffinement des représentations. On trouvera une description complète de l'architecture de l'implantation grâce au schéma 4.4.

D'autre part, une attention particulière a été portée à ce que l'implantation soit robuste et ceci sans dégrader trop les performances. Ainsi, plusieurs arithmétiques sont utilisées en même temps. En effet, le calcul des triangulations de Delaunay (contraintes ou non) sur des échantillons à reconstruire se révèle être souvent sensible. Les objets numérisés présentent, en effet, bien souvent des configurations dégénérées pour Delaunay. L'emploi d'une arithmétique robuste, s'est donc avérée être incontournable pour ce type de calcul. Par contre, pour l'évaluation de nos fonctions d'interpolation, les calculs peuvent être effectués avec moins de rigueur. L'emploi d'un type de

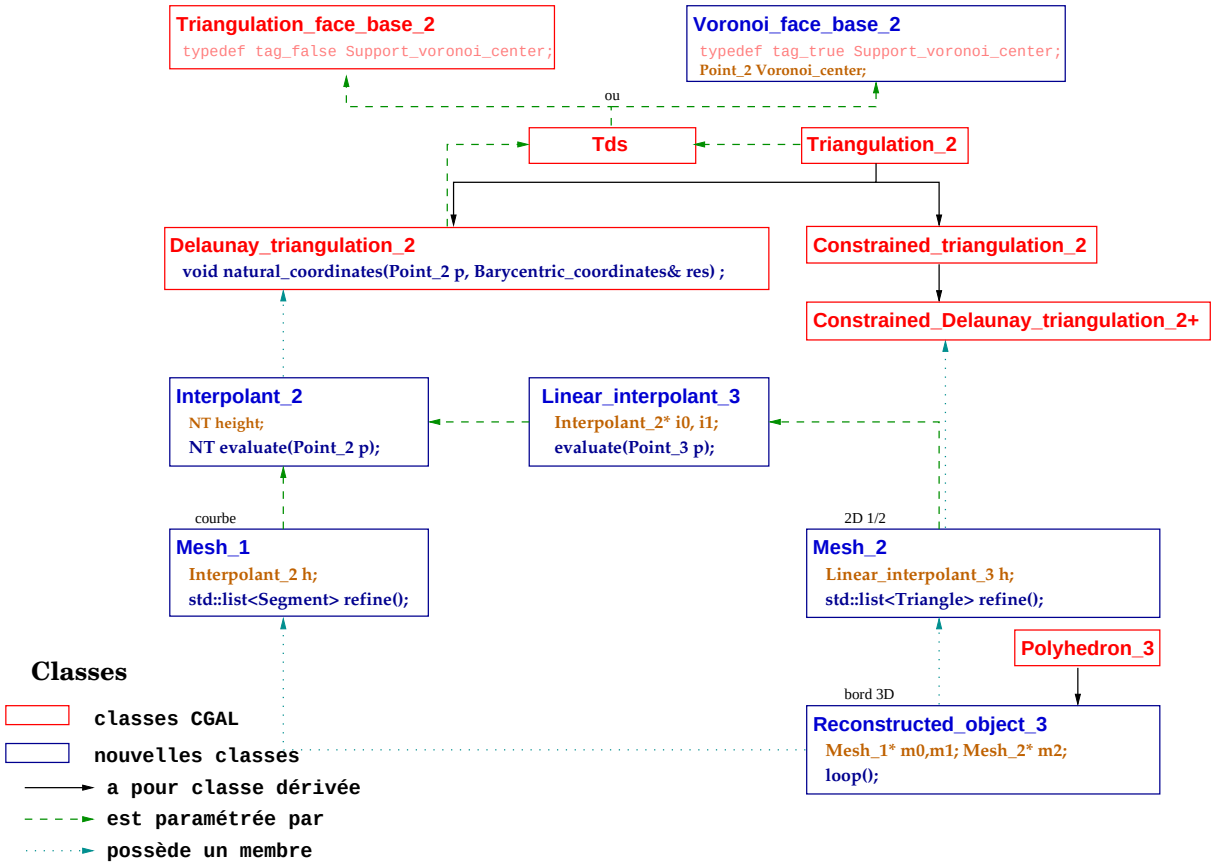


FIG. 4.4 – Illustration : organisation de l'implantation réalisée pour la reconstruction de sections.

nombre simple suffit amplement. Avec ce compromis, l'implantation s'est révélée être vraiment fiable à l'utilisation et d'une efficacité acceptable.

4.4.2 *Fonctionnalités offertes*

Adaptation au cas de sections radiales

Remarquons que l'on peut aussi songer à employer notre méthode dans le cas où chaque paire de sections $(\mathcal{S}_i, \mathcal{S}_{i+1})$ est prise radialement autour d'un axe Δ_i. Si l'on considère que les sections sont décrites par les coordonnées cylindriques autour de l'axe Δ_i. On prendra alors θ_i l'angle de la section $\mathcal{S}_i$ comme origine des angles ($\theta_i = 0$). Ainsi les sommets des contours polygonaux de $\mathcal{S}_i$ seront représentés en coordonnées cylindriques $(r, 0, z)$ et les sommets de $\mathcal{S}_{i+1}$ par les triplets (r, θ_{i+1}, z), où θ_{i+1} est fixe. L'interpolation linéaire entre deux sections s'effectue alors entre les angles $\theta_i = 0$ et θ_{i+1}. Ainsi à tout point P de coordonnées (r, θ, Z) compris entre les deux sections ($\theta \in [0, \theta_{i+1}]$) on peut associer un «projeté radial» P_{rz} de coordonnées (r, z) en dimension 2 pour le calcul des fonctions d'interpolation h_i et h_{i+1} dans chaque section. On définit alors H ainsi :

$$\forall P(r,\theta,z) \in \mathbb{R} \times [\theta_i, \theta_{i+1}] \times \mathbb{R},\ \mathrm{H}(P) = \frac{(\theta_{i+1} - \theta)\mathrm{h}_i(P_{rz}) + (\theta - \theta_i)\mathrm{h}_{i+1}(P_{rz})}{\theta_{i+1} - \theta_i}$$

Pour obtenir une représentation de l'objet global, il ne reste alors qu'à exprimer chaque tranche dans une base cartésienne de référence.

Considérons $(O, \vec{i}, \vec{j}, \vec{k})$ une base orthonormée de référence. L'axe Δ_i est défini par un point O_i et un vecteur unitaire $\vec{d_i}$, considérons alors $\vec{x}_i$ le vecteur normal à $\vec{d_i}$ définissant une base pour la section $\mathcal{S}_i$ et $\vec{y}_i$ le vecteur manquant pour former la base orthonormée de l'espace $(O_i, \vec{x}_i, \vec{y}_i, \vec{d_i})$, base associée à l'interpolation des sections $\mathcal{S}_i$ et $\mathcal{S}_{i+1}$. Rappelons comment opérer le changement de base entre coordonnées cylindriques (r, θ, Z), et coordonnées cartésiennes (X_i, Y_i, Z_i) dans la base $(O_i, \vec{x}_i, \vec{y}_i, \vec{d_i})$:

$$\left\{ \begin{array}{rcl} X_i & = & r\cos\theta \\ Y_i & = & r\sin\theta \\ Z_i & = & Z \end{array} \right.$$

Il ne reste alors qu'à opérer un changement entre les deux bases orthonormées $(O_i, \vec{x}_i, \vec{y}_i, \vec{d_i})$ et $(O, \vec{i}, \vec{j}, \vec{k})$. En notant M la matrice des coordonnées de $\vec{x}_i$, $\vec{y}_i$ et $\vec{d_i}$ dans la base de référence, on a enfin :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = O_i + M \begin{pmatrix} r\cos\theta \\ r\sin\theta \\ Z \end{pmatrix}$$

Calcul de sections intermédiaires

Pour certaines applications, le besoin se fait sentir de calculer des sections intermédiaires entre les sections de données, plutôt que les points éparpillés qui nous servent à raffiner la représentation des tranches. Dans ce but, on peut adapter l'algorithme de raffinement de contours afin de pouvoir représenter la fonction H à une hauteur déterminée z_c. Il suffit juste de pouvoir initialiser correctement la triangulation support au raffinement. Pour ceci, on pourra, par exemple, prendre comme contour approché l'intersection du plan d'équation $z = z_c$ et du bord représenté par les triangles bipolaires de la triangulation de Delaunay contrainte $\mathfrak{CD}_i$ avant tout raffinement, qui devrait capter la topologie de la courbe recherchée. Alors on peut calculer les zéros des arêtes bipolaires de ce contour approché pour la fonction $\mathrm{h}_{z_c}(p) = \mathrm{H}(p, z_c)$ et ne conserver que ces points exacts pour initialiser $\mathfrak{R}$ la triangulation support de l'algorithme de raffinement de contours. On pourra alors procéder comme pour une section de données et fournir une approximation polygonale du niveau zéro de la fonction h_{z_c} qui s'écrit ainsi :

$$\mathrm{h}_{z_c}(p) = \frac{(z_{i+1} - z_c)\mathrm{h}_i(p) + (z_c - z_i)\mathrm{h}_{i+1}(p)}{z_{i+1} - z_i}$$

§4.5 Garanties théoriques

4.5.1 Critère de mise en correspondance des contours

Précisons maintenant comment les contours de deux sections consécutives sont mis en correspondance par la méthode. Notons, tout d'abord, que pour les méthodes dédiées à la reconstruction de sections, le problème de la mise en correspondance des contours est couplé à la gestion des changements de topologie entre les sections. En effet, si l'on observe certaines méthodes de type surfacique, on s'aperçoit que la possibilité de «guider» la mise en correspondance des contours se paye bien souvent par une grande difficulté à gérer des changements de topologie entre les sections. Ceci est, en fait, assez compréhensible, la question porte sur les informations que l'on possède sur la nature des objets que la méthode aura à reconstruire. Si l'on prétend, par exemple, réaliser une méthode qui n'aura à reconstruire que des objets homéomorphes à une sphère, on peut forcer la mise en correspondance des contours dans toutes les situations entre les sections et l'on est déchargé du problème du changement de topologie. Notre méthode se place dans un cadre différent, elle présuppose uniquement que les objets à reconstruire sont des *volumes*. Il peut, par contre, y avoir plusieurs objets par section, possédant chacun des types topologiques complexes (*i.e.* avec un ou plusieurs trous, par exemple). La souplesse de gestion du changement de type de topologie et la possibilité de fournir une transition lisse entre les sections, imposent, en quelque sorte, un critère de mise en correspondance assez exigeant par rapport à ceux d'autres méthodes moins générales. Cependant, le critère qui est le nôtre a pour avantage d'être simple à visualiser et ainsi de permettre à l'utilisateur de savoir si notre méthode fournira un résultat acceptable pour ses données.

Pour fixer les idées, dans le cas d'un raffinement infini, notons $\mathcal{O}_r$ le volume intérieur délimité par notre surface reconstruite, représentation exacte de l'ensemble $\{P \in \mathbb{R}^3 / \ \mathrm{H}(P) > 0\}$. Considérons $\mathfrak{C}_1$ un contour de la section $\mathcal{S}_i$ et $\mathfrak{C}_2$ un contour de la section suivante $\mathcal{S}_{i+1}$. On définit *l'intérieur* délimité par un contour $\mathfrak{C}$ comme la composante connexe finie de $\Pi \setminus \mathfrak{C}$, où Π est le plan de section contenant $\mathfrak{C}$. Remarquons que, par nature, $\mathfrak{C}_1$ et $\mathfrak{C}_2$ sont des courbes planes fermées tracées sur la surface délimitant de $\mathcal{O}_r$. On notera, de plus, $\mathcal{O}_{r_{|[z_i,z_{i+1}]}}$ l'intersection de $\mathcal{O}_r$ avec la portion de l'espace $\mathbb{R}^3$ comprise entre les deux sections, $\mathbb{R}^2 \times [z_i, z_{i+1}]$. Ainsi, on peut formaliser la définition intuitive de correspondance entre contours ainsi :

Définition 4.3 *On dira que* $\mathfrak{C}_1$ *et* $\mathfrak{C}_2$ *sont* mis en correspondance *si l'intérieur délimité par* $\mathfrak{C}_1$ *et celui délimité par* $\mathfrak{C}_2$ *sont inclus dans une même composante connexe de* $\mathcal{O}_{r_{|[z_i,z_{i+1}]}}$.

La mise en correspondance des contours résulte de la nature même de l'objet $\mathrm{H}^{-1}(0)$ que l'on cherche à représenter. Pour préciser les conditions de mise en correspondance, étudions donc comment il est construit. Comme précédemment énoncé (cf 4.3.4), dans une tranche comprise entre deux sections successives d'indice i et $i+1$, la surface $\mathrm{H}^{-1}(0)$ peut être représentée par une *fonction* de l'ensemble $D_i = \{(x,y) \in \mathbb{R}^2 / \ \mathrm{h}_i(x,y)\mathrm{h}_{i+1}(x,y) < 0\}$ dans $]z_i, z_{i+1}[$, qui associe à tout couple de coordonnées $(x,y) \in D_i$ une hauteur z tel que $\mathrm{H}(x,y,z) = 0$. Grâce au corollaire 4.2, on peut donc considérer qu'à l'ensemble $D_i \subset \mathbb{R}^2$ correspond le bord de l'objet reconstruit $\mathcal{O}_r$.

Examinons ce qui se passe dans le reste du plan $\mathbb{R}^2$. Dans ce but, considérons les deux ensembles suivant :

$$\begin{aligned} D_i^- &= \{(x,y) \in \mathbb{R}^2 / \ \mathrm{h}_i(x,y) \leq 0 \text{ et } \mathrm{h}_{i+1}(x,y) \leq 0\} \\ D_i^+ &= \{(x,y) \in \mathbb{R}^2 / \ \mathrm{h}_i(x,y) \geq 0 \text{ et } \mathrm{h}_{i+1}(x,y) \geq 0\} \end{aligned}$$

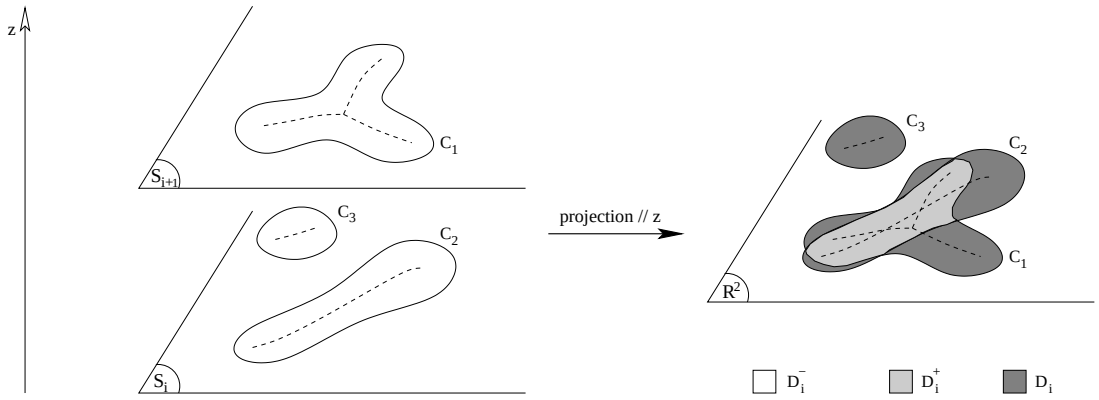


FIG. 4.5 – Illustration : *décomposition* de la projection de la portion d'espace comprise entre deux sections et *mise en correspondance* des contours : ici, C_1 et C_2 sont mis en correspondance, C_1 et C_3 non.

Notons, tout d'abord, que D_i, D_i^+ et D_i^- réalisent une partition de $\mathbb{R}^2$. On peut, de plus, remarquer que pour $(x, y) \in D_i^+$ et $z \in [z_i, z_{i+1}]$, on a $\mathrm{H}(x, y, z) > 0$. Ainsi, le cylindre $D_i^+ \times [z_i, z_{i+1}] \subset \mathbb{R}^3$ est *intérieur* à l'objet $\mathcal{O}_r$. Par symétrie, on peut aussi écrire que le cylindre $D_i^- \times [z_i, z_{i+1}] \subset \mathbb{R}^3$ est *extérieur* à l'objet $\mathcal{O}_r$.

Propriété 4.3 *Deux contours $\mathfrak{C}_1$ et $\mathfrak{C}_2$ sont* mis en correspondance *par notre méthode de reconstruction si et seulement si les projections, parallèlement à l'axe z, de leur intérieur respectif s'intersectent dans $\mathbb{R}^2$.*

Preuve. Notons tout d'abord que, si les projections, parallèlement à l'axe z, de l'intérieur de $\mathfrak{C}_1$ et de l'intérieur de $\mathfrak{C}_2$ s'intersectent en une partie non-vide $\Delta \subset \mathbb{R}^2$, alors $\mathfrak{C}_1$ et $\mathfrak{C}_2$ sont *mis en correspondance* au sens de la définition 4.3. En effet, Δ est incluse[26] dans D_i^+ et le cylindre $\Delta \times [z_i, z_{i+1}]$, intérieur à $\mathcal{O}_r$, connecte les *intérieurs* délimités par $\mathfrak{C}_1$ et $\mathfrak{C}_2$.
Réciproquement, supposons que la projection de l'intérieur de $\mathfrak{C}_1$ et la projection de l'intérieur de $\mathfrak{C}_2$ soient disjointes. Raisonnons par l'absurde, supposons alors que $\mathfrak{C}_1$ et $\mathfrak{C}_2$ sont *mis en correspondance*, cela signifie qu'il existe un chemin continu l dans $\mathcal{O}_{r|[z_i,z_{i+1}]}$ reliant un point P_1 de l'intérieur de $\mathfrak{C}_1$ à un point P_2 de l'intérieur de $\mathfrak{C}_2$. Notons p_z la projection parallèlement à l'axe z. La projection étant une application continue, $\mathrm{p}_z(l) \subset \mathrm{p}_z(\mathcal{O}_{r|[z_i,z_{i+1}]})$ est un chemin continu qui relie la projection de l'intérieur de $\mathfrak{C}_1$ à la projection de l'intérieur de $\mathfrak{C}_2$. Ces deux ensembles étant disjoints, il existe au moins un point sur $\mathrm{p}_z(l)$ dans D_i^-. Or, on a $\mathrm{p}_z(\mathcal{O}_{r|[z_i,z_{i+1}]}) = D_i \cup D_i^+$, ce qui est contradictoire. □

On peut remarquer que la mise en correspondance des contours ne dépend pas de l'espacement entre les sections, puisque l'on décide de la mise en correspondance après projection le long de l'axe z. Ainsi, notre méthode est relativement sensible au choix de l'axe z pour la numérisation de sections. Pour un objet possédant une géométrie donnée, un choix malencontreux de l'axe peut conduire notre méthode à exiger une forte densité de sections. On notera, quand même, qu'à l'opposé un choix adapté de l'axe de mesure peut permettre un échantillonnage très lâche en z. Le problème se pose évidemment des cas où l'acquisition des données est limitée, en même temps, en ce qui concerne le choix de l'axe et de l'espacement entre les sections.

[26] Δ est, en fait, une composante connexe de D_i^+.

Pour une mise en correspondance *plus fine*, on peut exiger que les squelettes intérieurs des contours de $\mathfrak{C}_1$ et de $\mathfrak{C}_2$ soient eux aussi mis en correspondance. En effet, pour espérer obtenir une surface reconstruite opérant des transitions de bonne qualité entre les sections, c'est l'interpolation des squelettes et de leur points singuliers[27] qu'il faut contrôler. Pour notre méthode, il peut ainsi être souhaitable que les squelettes des contours $\mathfrak{C}_1$ et $\mathfrak{C}_2$ se projettent tous deux à l'intérieur de la même composante connexe de D_i^+ (*i.e.* l'intersection de la projection des intérieurs de $\mathfrak{C}_1$ et $\mathfrak{C}_2$). La reconstruction se trouve alors, au moins en pratique, beaucoup plus conforme à ce que l'on peut attendre.

§4.6 *Résultats*

4.6.1 *Etude des performances*

Etude qualitative

Avant d'étudier les performances pratiques de notre implantation, formulons quelques remarques qualitatives sur le coût relatif des différentes opérations intervenant lors d'une exécution typique. Ces considérations ont pour base les résultats de l'outil d'analyse d'exécution *gprof*. Nous nous intéresserons surtout au cas du raffinement, la reconstruction non raffinée ne mettant principalement en œuvre que des opérations du type insertion dans un triangulation. Pour une exécution où on demande un raffinement des contours proportionné au raffinement de la surface reconstruite entre les sections[28], on s'aperçoit que le calcul du raffinement de contours est assez coûteux, environ 60% de l'ensemble. Ce coût est surtout dû au calcul du zéro associé à une arête bipolaire. Cette opération nécessite un nombre d'évaluations de la fonction d'interpolation non borné (bien qu'il soit aux alentours de 5 en moyenne, pour des précisions courantes) et constitue à elle seule la moitié du temps de raffinement des contours. À titre de comparaison, le calcul du raffinement entre les sections prend aussi 30% du temps total, où la moitié du temps (ce qui représente donc 15% du temps d'exécution) est consacrée au calcul de la hauteur z pour les points de raffinement entre les sections. On voit, ainsi, l'avantage substantiel que l'on pourra avoir, si les contours sont suffisamment fins, à choisir un ε_1 plus grand.

Etude statistique

Passons maintenant à des considérations plus précises sur les performances concrètes de notre méthode. Les temps CPU, détaillés dans les tableaux de résultats 4.1 et 4.2, sont exprimés en secondes. Ils ont été obtenus sur une machine équipée d'un processeur Intel Pentium i686 1GHz et disposant de 1Go de RAM.

Interpolation non raffinée. Dans le cas où aucun raffinement n'est demandé, l'algorithme n'effectue, en fait, que des opérations du type insertion dans une triangulation. On peut, alors, s'attendre légitimement à un taux de points traités par seconde relativement stable. En effet, la dernière colonne du tableau 4.1 exhibe un nombre de points traités par seconde entre 1000 et 4000 suivant le jeu de données avec une moyenne située aux environs de *2000 points par seconde*, performance qui devrait être améliorée grâce aux progrès encore réalisables en ce qui concerne

[27] C'est, par exemple, cet aspect qui est retenu pour les applications de *morphing*.

[28] En pratique, on utilise $\varepsilon_1 = \varepsilon_1/2$ qui donne de bons résultats, puisque les contours sont assez anguleux.

Données			Reconstruction		t	n/t
Modèle	nombre de points (n)	nombre de sections	nbre de points d'intersections	nombre de facettes		
AnevIliaqueToulouse	1139	33	85	2516	0,83	1372
HH08_STCD	8708	80	762	18957	5,19	1678
HH08_STPU	6025	56	570	12581	3,26	1848
brain	3205	15	944	8574	0,89	3601
c_75_16_48_a	499	7	8	961	0,15	3327
c_75_2_52_a	4161	53	235	9208	1,33	3129
c_75_4_52_a	2098	27	73	4410	0,66	3179
c_75_8_52_a	1064	14	14	2164	0,32	3325
cartilage	222	11	56	514	0,11	2018
head	856	17	202	1993	0,26	3292
heart	1222	29	207	2835	0,55	2222
hip	926	14	189	2138	0,47	1970
lung	3034	34	703	7398	1,45	2092
pelvis	2068	23	440	5160	0,49	4220
pelvis_high	7175	45	1098	16984	3,19	2249
poly_mod_zone1	84564	21	1101	164407	83,8	1009
poly_mod_zone2	90810	16	2410	175877	85,41	1063
skull	24453	113	5099	67743	7,56	3235
z	4450	10	0	8192	1,69	2633

TAB. 4.1 – Temps de calcul CPU (en secondes) *sans raffinement* pour les modèles de notre base de données.

l'implantation des triangulations 2D de CGAL. On peut expliquer les variations observées, en partie, par le coût du calcul des points générés par les intersections des contours quand on les projette dans les triangulations de DELAUNAY contraintes. Les opérations d'insertion de contraintes qui s'intersectent sont complexes et amoindrissent sensiblement les performances. Évidemment, il faut aussi considérer la présence en nombre de configurations dégénérées qui demanderont un calcul plus poussé pour les prédicats filtrés. C'est, par exemple, le cas des modèles AnevIliaqueToulouse, poly_mod_zone1 et poly_mod_zone2.

Interpolation raffinée. Lorsqu'un raffinement de la représentation est demandé, il est plus difficile d'établir une mesure objective des performances. En effet, pour une qualité équivalente (celle des résultats en image de ce manuscrit, par exemple), la structure contours dans chaque section, la nature de l'objet à reconstruire, nécessite d'adapter la précision de raffinement demandée. En général, les résultats sont obtenus avec $\varepsilon_1 = 0,5$ et $\varepsilon_2 = 1$. Cependant, certains modèles ont été traités avec des options particulières, on aura pas cherché à raffiner les contours (*i.e.* $\varepsilon_1 = +\infty$) pour les modèles poly_mod_zone1 et poly_mod_zone2, les contours étant déjà «trop» précis. Comme on a établi que le raffinement des contours coûte proportionnellement plus cher que le raffinement entre les sections. On observe, de plus, une baisse de performance sur des modèles aux contours très anguleux, comme c_75_16_48_a, par exemple. Les résultats du tableau 4.2 sont, en fait, surtout destinés à montrer que le raffinement est accessible dans des temps de calcul raisonnables (pour peu que la précision demandée le soit aussi...).

Provenance des modèles : tous nos remerciements au projet M3N de l'INRIA Rocquencourt pour le modèle AnevIliaqueToulouse, au projet ÉPIDAURE de l'INRIA Sophia-Antipolis pour les

Données			Reconstruction		t
Modèle	nombre de points	nombre de sections	nombre de points calculés	nombre de facettes	
AnevIliaqueToulouse	1139	33	123	2592	2,02
HH08_STCD	8708	80	812	19033	12,02
HH08_STPU	6025	56	601	12622	8,43
brain	3205	15	14325	33367	14,7
c_75_16_48_a	499	7	911	3704	9,73
c_75_2_52_a	4161	53	4620	14351	9,91
c_75_4_52_a	2098	27	1747	6967	5,09
c_75_8_52_a	1064	14	932	6079	18,93
cartilage	222	11	772	2867	3,5
head	856	17	1613	5606	16,21
heart	1222	29	445	5781	16,79
hip	926	14	1407	5107	9,48
lung	3034	34	1428	14224	39,89
pelvis	2068	23	7262	23779	37,43
pelvis_high	7175	45	7965	39808	118,92
poly_mod_zone1	84564	21	6099	174387	443,7
poly_mod_zone2	90810	16	4498	180053	351,77
skull	24453	113	16493	82864	29,8
z	4450	10	2696	13987	21,01
* Avec simplification :					
poly_mod_zone2 ($p = 1\%$, $\varepsilon_s = 0,1$)			4535	180127	193,24
poly_mod_zone2 ($p = 1\%$, $\varepsilon_s = 0,01$)			4500	180057	288,25
poly_mod_zone2 ($p = 1\%$, $\varepsilon_s = 0,001$)			4499	180055	353,39

TAB. 4.2 – Temps de calcul CPU (en secondes) *avec raffinement* pour les modèles de notre base de données.

modèles HH08_STCD, HH08_STPU et z, et enfin à TURBOMECA pour poly_mod_zone1 et poly_mod_zone2. Les autres modèles proviennent de la base de données de NUAGES, logiciel développé au sein de notre équipe.

Analyse de la simplification de données

Tentons d'évaluer l'apport de la simplification de données dans le cas de modèles dont les contours dans chaque section sont fournis très précisément. Les modèles concernés étant dans notre base de données poly_mod_zone1 et poly_mod_zone2, les résultats établis dans le tableau 4.3 ont, pour but, de montrer le gain substantiel que l'on peut attendre en utilisant la simplification des données à une précision ε_s raisonnable. Rappelons que dans le cas où l'on demande une simplification des données, ce sont les contours originels qui sont introduits dans notre reconstruction, ainsi l'erreur de cordes par rapport aux données est *nulle*. C'est juste l'interpolation entre les sections qui n'est pas calculée sur l'ensemble des données. Grâce à la simplification, on peut obtenir un résultat totalement équivalent à celui obtenu par un raffinement classique en presque deux fois moins de temps (cf. tableau 4.2, poly_mod_zone2 ($p = 1\%$, $\varepsilon_s = 0,1$)). Rappelons que dans le cas précédemment évoqué le raffinement entre les sections est demandé par $\varepsilon_2 = 1$, ce qui rend ε_s acceptable. Une étude plus fine, dont les résultats sont décrits dans le tableau 4.3, compare tranche par tranche, le temps de raffinement, $t_r(i-1, i)$ pour $\varepsilon_2 = 1$ avec

Modèle : poly_mod_zone2		Simplification			
Section	nombre de points (n_i)	nombre de points ajoutés (n_i^+)	taux de simplification $s_i = 100(pn_i + n_i^+)/n_i$	t_i	$\frac{100t_r(i-1,i)}{t_{ref}(i-1,i)}$
* $p = 1\%$, $\varepsilon_s = 0,1$:					
0	11188	1474	14,17	2,74	-
1	11252	1465	14,02	2,19	39,55
2	11164	1492	14,36	2,81	44,70
3	11148	1529	14,71	2,17	41,24
4	11272	1511	14,4	2,9	44,88
* $p = 1\%$, $\varepsilon_s = 0,01$:					
0	11188	5772	52,58	2,62	-
1	11252	5991	54,24	2,74	75,66
2	11164	5909	59,92	2,65	77,68
3	11148	5917	54,07	2,71	77,66
4	11272	6129	55,37	2,73	78,92
* $p = 1\%$, $\varepsilon_s = 0,001$:					
0	11188	10238	92,50	1,82	-
1	11252	10360	93,06	1,98	100,61
2	11164	10262	92,91	2	102,20
3	11148	10234	92,79	1,87	101,88
4	11272	10316	92,51	1,99	101

TAB. 4.3 – Résultats expérimentaux pour la simplification d'échantillons.

simplification au temps de référence, $t_{ref}(i-1,i)$, pour le raffinement de la même tranche, les fonctions étant définies sur l'ensemble des données contenues dans les sections. On s'aperçoit, en corrélation avec les résultats du tableau 4.2, que les performances sont bien meilleures pour un raffinement avec simplification, mais qu'elles se dégradent, jusqu'à devenir un handicap, si l'on est trop exigeant sur ε_s.

4.6.2 *Comparaison avec les méthodes standard*

Malheureusement, la comparaison avec les méthodes considérées comme classiques pour la reconstruction de sections n'est pas aisée. En effet, les données ne vérifiant pas, par nature, de «bonnes» conditions d'échantillonnage, comme décrites au chapitre 3, toutes les méthodes ne s'avèrent pas capables de traiter correctement les mêmes échantillons. De plus, la qualité des résultats rentre aussi en ligne de compte, plus encore que pour les méthodes généralistes, puisqu'il convient ici de traiter correctement la mise en correspondance et les problèmes de changement de topologie. Ainsi, malgré nos recherches, il nous a été impossible de trouver plus de données pour nos tests et encore moins de modèles reconstruits par d'autres méthodes, afin d'apprécier la qualité de la reconstruction, ni même de tableaux de résultats contenant certaines de nos données pour pouvoir comparer les performances. La seule méthode à notre disposition, mais qui par chance est une méthode de référence, pour ce problème est NUAGES, développée par B. GEIGER. Bien qu'ayant une règle de mise en correspondance différente, les résultats obtenus par les deux méthodes sur les modèles de la base de données NUAGES sont de qualité équivalente. Par ailleurs, les modèles qui nous ont été fournis par ÉPIDAURE ou TURBOMECA l'ont tout d'abord étés du fait que le résultat fourni par NUAGES ne convenait pas. En terme de temps de calcul, si l'on se réfère aux résultats de la thèse de B. GEIGER [Gei93], les temps de calcul pour NUAGES (~ 500 points à la seconde en 1993) aujourd'hui et ceux pour notre reconstruction sans raffinement sont comparables.

4.6.3 *Résultats et commentaires*

Les résultats présentés dans cette section correspondent aux calculs décrits dans le tableau 4.2, la surface polyédrique reconstruite a donc été raffinée. Laissons maintenant place à quelques résultats en images :

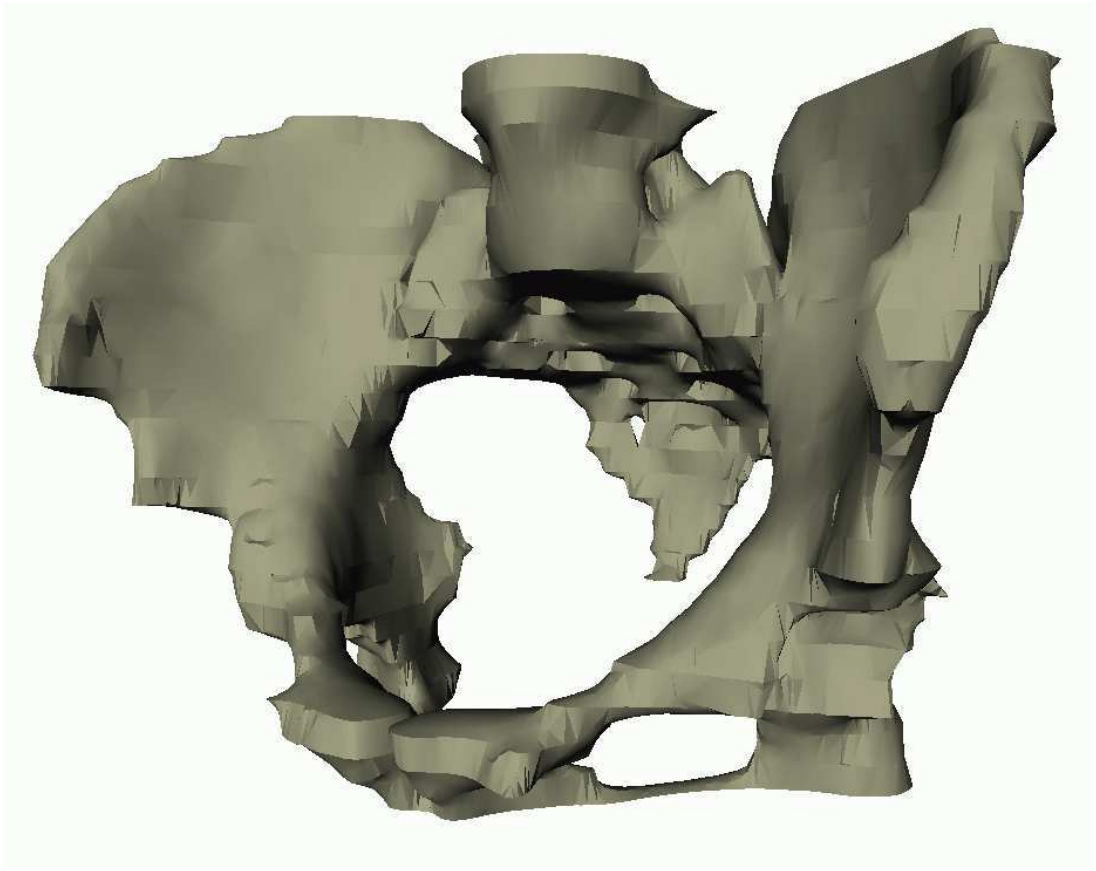


FIG. 4.6 – Résultat : un bassin, modèle *pelvis*.

Figure 4.6. Le bassin, modèle de la base de données de NUAGES, présente l'intérêt de posséder des sections très dissemblables tout en vérifiant notre critère de mise en correspondance. Ainsi, il permet d'exploiter les possibilités de raffinement de la représentation de la surface reconstruite et met en valeur la transition lisse opérée pour raccorder les contours entre les sections.

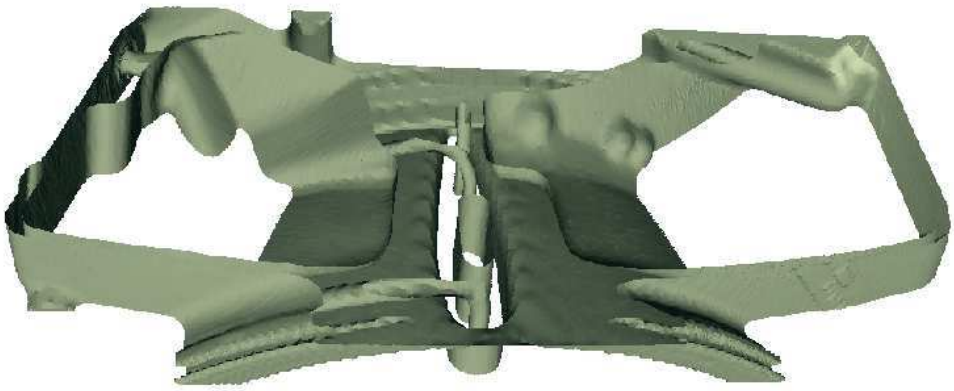


FIG. 4.7 – Résultat : une pièce mécanique, modèle *poly_ mod_ zone2*.

Figure 4.7. Le modèle a pour particularité de posséder des contours représentés très (trop) précisément. Bien que l'objet à reconstruire possède une topologie relativement complexe et que certains contours soient très proches dans une même section, la méthode fournit une reconstruction correcte aussi bien avec simplification de données que sans.

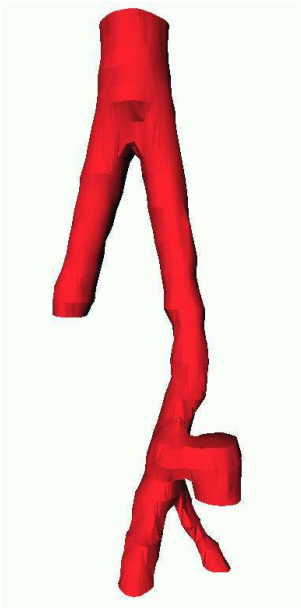


FIG. 4.8 – Résultat : un anévrisme sur l'artère iliaque, modèle *AnevIliaqueToulouse*.

Figure 4.8. Le modèle ne présente pas de difficultés techniques particulières. On peut observer le cas, dit *de bifurcation*, qui pose tant de problèmes aux méthodes de type surfacique décrites en 1.4. Pour notre méthode, ce cas ne diffère en rien des autres et les possibilités de raffinement nous permettent d'obtenir un résultat appréciable.

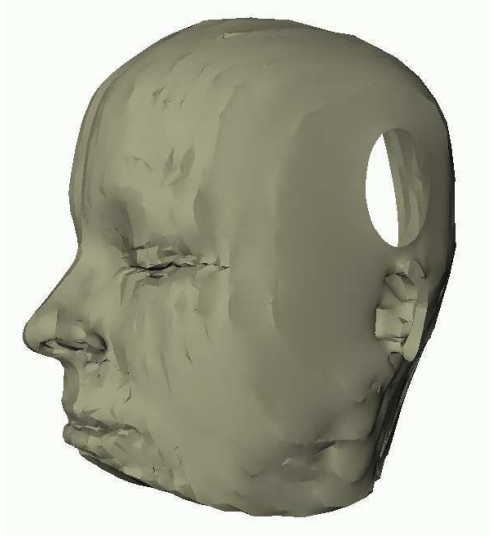


FIG. 4.9 – Résultat : une tête, modèle *head*.

Figure 4.9. Un autre modèle de la base de données de NUAGES qui ne pose pas de grandes difficultés, on peut, cependant remarquer la légère discontinuité de la direction normale à la surface aux passages des plans de section.

FIG. 4.10 – Résultat : un crâne, modèle *skull*.

Figure 4.10. Ce modèle propose un cas intéressant. Les premières sections sont relativement faciles à interpoler, puis du fait du procédé de mesure, les mesures sont faussées autour de la bouche et surtout des dents du patient (à cause de plombages dentaires, semble-t-il). Malgré des contours «bruités» par ces interférences, notre méthode est tout de même capable de fournir un résultat acceptable.

FIG. 4.11 – Résultat : un cœur, modèle *heart*.

Figure 4.11. Encore un cas sans difficultés particulières, mais où la discontinuité de la direction normale à la surface aux passages des plans de section est particulièrement marquée au milieu du modèle. La situation n'est, en fait, pas simple, le contour où l'on observe cette discontinuité, étant en retrait par rapport à ceux des sections adjacentes.

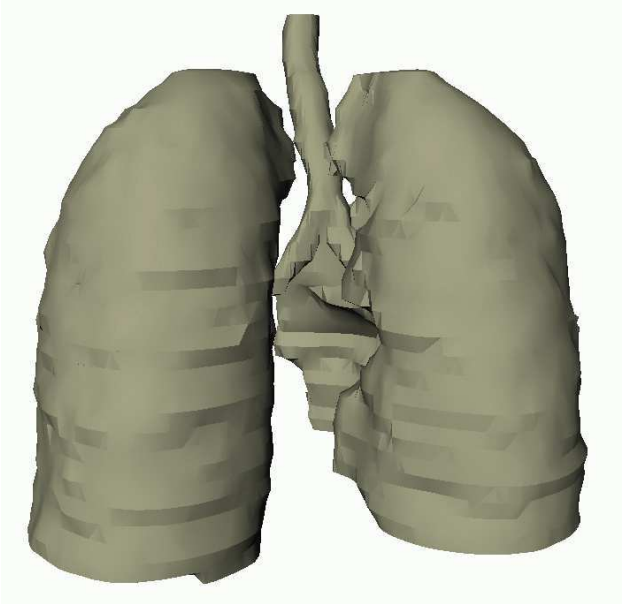


FIG. 4.12 – Résultat : des poumons, modèle *lung*.

Figure 4.12. Un dernier modèle qui ne présente pas de réels problèmes pour notre méthode. On pourra, tout de même, observer la bifurcation des bronches correctement traitée.

Conclusion et perspectives

§1 Réflexions sur les méthodes proposées

Dans ce document, nous avons envisagé le problème de la reconstruction de modèles géométriques sous différents aspects. Après avoir établi le cadre de notre travail en rappelant quelques notions de Géométrie algorithmique, nous avons tenté de présenter un état de l'art des méthodes existantes en précisant, pour chacune, leurs forces et leurs faiblesses. Dans un deuxième temps, la mise en pratique d'une méthode de référence, les Alpha-formes, nous a permis de mieux cerner une configuration spéciale, particulière à la triangulation de DELAUNAY pour des points échantillonnés sur une surface, *i.e.* le *sliver*. Par ailleurs, nous avons aussi présenté l'implantation efficace d'un outil très puissant pour le problème de l'interpolation, les coordonnées barycentriques associées aux *voisins naturels*. Revenons maintenant sur les deux méthodes proposées pour la reconstruction dans le cas de nuages de points et dans le cas de sections.

Reconstruction par déploiement de surface

La méthode présentée au chapitre 3 pour la reconstruction d'un échantillon de points sans structure présente différentes caractéristiques remarquables par rapport aux solutions précédentes. Tout d'abord, elle développe une approche originale du problème, en encadrant le déploiement d'une surface sur l'échantillon de points par des *contraintes topologiques* permettant d'obtenir une surface sans singularités. Par ailleurs, pour une efficacité concrète de la méthode sur des échantillons s'écartant du cas idéal, on introduit la notion de *prudence* au cours du déploiement. Ce déploiement prudent est traduit grâce à des critères locaux, dont le calcul peut être réalisé très efficacement. Par ailleurs, profitant de la nature même de la méthode, différentes conditions heuristiques d'arrêt au déploiement ont été énoncées pour permettre de reconstruire les surfaces à bords.

Les résultats de cette approche sont convaincants sous plusieurs aspects. Tout d'abord, on apporte une garantie théorique sur la validité de la reconstruction dans le cas d'un échantillon *régulier* de points dans l'espace. De plus, on établit aussi la validité de notre méthode pour un ε-échantillon de points dans le plan, quand $\varepsilon < 1/5$. Par ailleurs, les nombreux tests expérimentaux et les bons résultats obtenus pour la plupart des modèles qui nous ont été soumis nous laissent à penser que les critères utilisés se comportent correctement, même lorsque l'on s'écarte un peu d'un cas d'échantillonnage idéal. Ainsi, notre méthode se révèle non seulement robuste et surtout efficace, aussi bien, en terme de temps de calcul que de qualité de la reconstruction obtenue.

Reconstruction par interpolation de sections

La méthode présentée au chapitre 4 propose une interpolation lisse dans le cas de données organisées en sections. Pour cela, elle tire parti des bonnes propriétés d'une fonction d'interpolation définie grâce au système de coordonnées barycentriques associées aux voisins naturels. Il s'agit, en fait, de reconstruire le niveau zéro de cette fonction, définie implicitement, simplement par le calcul de la triangulation de DELAUNAY dans chaque plan de section. Ainsi, profitant de l'existence de cet objet qui définit l'interpolation entre les sections, différentes possibilités sont offertes. D'une part, dans le cas de gros échantillons de points, on pourra opérer une étape de *simplification* des données, afin de ne conserver que les points nécessaires à la définition de la fonction d'interpolation à un certain niveau de précision. D'autre part, l'objet interpolant étant lisse entre deux sections consécutives, on peut demander un raffinement de la représentation afin d'obtenir une transition moins abrupte pour des contours très dissemblables ou dans le cas d'un changement de topologie des contours entre les sections.

En terme de résultats, un critère simple de mise en correspondance, considérant le recouvrement ou non des contours par projection, a été démontré. Cette condition permet de décrire précisément le comportement de l'algorithme suivant la nature des données. La méthode offre ainsi une reconstruction maîtrisée. On notera qu'aucune règle de mise en correspondance claire n'est, en général, précisée pour les méthodes volumiques. De plus, la condition établie, quoique restrictive par rapport à certains critères formulés pour des méthodes de nature surfacique, se révèle réaliste pour la plupart des modèles. Ainsi, on peut considérer que la méthode présentée propose un compromis acceptable entre la mise en correspondance et la gestion des changements de topologie qui ne présente, pour nous, aucune difficulté. En terme d'efficacité, grâce à des calculs opérés uniquement dans le plan, on dispose de performances tout à fait compétitives pour une reconstruction brute, le raffinement étant possible pour un coût acceptable en pratique.

§2 Perspectives de recherche

Le problème théorique de la reconstruction est évidemment loin d'être clos, pour peu que cela soit possible. Les méthodes et réflexions développées dans ce document ne cherchent qu'à apporter une contribution qui permettra peut-être de mieux cerner différents aspects importants et d'aboutir un jour à une compréhension complète des liens entre un échantillon de points, l'objet dont il provient et les différentes reconstructions que l'on peut opérer.

Plus pratiquement, nous pensons avoir développé des méthodes efficaces susceptibles de fournir des reconstructions correctes dans bien des cas. En fait, à partir des modèles qui nous échappent encore, on peut identifier différents enjeux importants si l'on souhaite progresser. Tout d'abord, l'étude précise des *slivers* — ou plus encore, des configurations où plusieurs *slivers* sont enchevêtrés — permettrait de lever bien des problèmes pour les différentes méthodes de reconstruction. En effet, ces tétraèdres particuliers créent des ambiguïtés qui se manifestent par des problèmes (trous, singularités...) plus ou moins identifiés dans la reconstruction finale obtenue grâce à la plupart des méthodes.

De plus, un autre problème pour lequel nous sommes encore plus démunis est la caractérisation de «bonnes» propriétés pour un échantillon de points autour d'une *arête vive*. En effet, pour le moment, on peut considérer qu'aucune méthode ne sait réellement traiter le cas spécial des arêtes vives. On observe, en fait, deux types de comportement : certaines méthodes arrondissent, en quelque sorte, les arêtes vives ; alors que d'autres, en général des méthodes fonctionnelles (souvent plus robustes au bruit), traitent un peu mieux les arêtes vives, la contrepartie étant

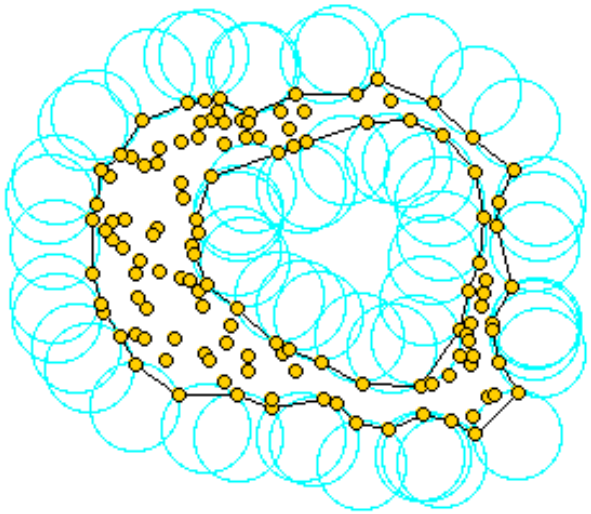
alors qu'elles lissent aussi les détails éventuellement présents dans le modèle.

En ce qui concerne le cas des sections (pour lequel on peut toujours envisager de nouveaux critères de mise en correspondance ou encore une nouvelle gestion des embranchements) nous considérons que les méthodes volumiques offrent un cadre bien meilleur pour la reconstruction que l'approche surfacique jusqu'ici développée. En fait, les méthodes volumiques permettent une gestion robuste et automatique des changements de topologie entre les sections, inaccessible aux méthodes surfaciques. Il s'agit, en fait, surtout d'établir des critères de mise en correspondance précis et peut-être plus flexibles que le nôtre (paramétrable, par exemple) pour les futures méthodes volumiques pour l'interpolation de sections.

Annexe A

Documentation CGAL : *Alpha-formes* 2D

2D Alpha Shapes (user manual)



Assume we are given a set S of points in 2D or 3D and we'd like to have something like "the shape formed by these points." This is quite a vague notion and there are probably many possible interpretations, the α-shape being one of them. Alpha shapes can be used for shape reconstruction from a dense unorganized set of data points. Indeed, an α-shape is demarcated by a frontier, which is a linear approximation of the original shape [BB97].

As mentionned in Edelsbrunner's and Mücke's paper [EM94], one can intuitively think of an α-shape as the following. Imagine a huge mass of ice-cream making up the space $\mathbb{R}^3$ and containing the points as "hard" chocolate pieces. Using one of these sphere-formed ice-cream spoons we carve out all parts of the ice-cream block we can reach without bumping into chocolate pieces, thereby even carving out holes in the inside (eg. parts not reachable by simply moving the spoon from the outside). We will eventually end up with a (not necessarily convex) object bounded by caps, arcs and points. If we now straighten all "round" faces to triangles and line segments, we have an intuitive description of what is called the α-shape of S. Here's an example for this process in 2D (where our ice-cream spoon is simply a circle) :

And what is α in the game ? α is the squared radius of the carving spoon. A very small value will allow us to eat up all of the ice-cream except the chocolate points themselves. Thus we already see that the α-shape degenerates to the point-set S for $\alpha \to 0$. On the other hand, a huge value of α will prevent us even from moving the spoon between two points since it's way too large. So we will never spoon up ice-cream lying in the inside of the convex hull of S, and hence the α-shape for $\alpha \to \infty$ is the convex hull of S.

A.1 Definitions

We distinguish two versions of alpha shapes. *Basic alpha shapes* are based on the Delaunay triangulation. *Weighted alpha shapes* are based on its generalization, the regular triangulation, replacing the euclidean distance by the power to weighted points.

There is a close connection between alpha shapes and the underlying triangulations. More precisely, the α-complex of S is a subcomplex of this triangulation of S, containing the α-exposed k-simplices, $0 \leq k \leq d$. A simplex is α-exposed, if there is an open disk (resp. ball) of radius $\sqrt{\alpha}$ through the vertices of the simplex that does not contain any other point of S, for the metric used in the computation of the underlying triangulation. The corresponding α-shape is defined as the underlying interior space of the α-complex (see [EM94]).

In general, an α-complex is a non-connected and non-pure polytope, it means, that one k-simplex, $0 \leq k \leq d-1$ is not necessary adjacent to a $(k+1)$-simplex.

The α-shapes of S form a discrete family, even though they are defined for all real numbers α with $0 \leq \alpha \leq \infty$. Thus, we can represent the entire family of α-shapes of S by the underlying triangulation of S. In this representation each k-simplex of the underlying triangulation is associated with an interval that specifies for which values of α the k-simplex belongs to the α-shape. Relying on this fact, the family of α-shapes can be computed efficiently and relatively easily. Furthermore, we can select an appropriate α-shape from a finite number of different α-shapes and corresponding α-values.

A.2 Functionality

The class *CGAL::Alpha_shape_2<Dt>* represents the family of α-shapes of points in a plane for *all* positive α. It maintains the underlying triangulation *Dt* which represents connectivity and order among squared radius of its faces. Each k-dimensional face of the *Dt* is associated with an interval that specifies for which values of α the face belongs to the α-shape. There are links between the intervals and the k-dimensional faces of the triangulation.

The class *CGAL::Alpha_shape_2<Dt>* provides functions to set and get the current α-value, as well as an iterator that enumerates the α-values where the α-shape changes.

It provides iterators to enumerate the vertices and edges that are in the α-shape, and functions that allow to classify vertices, edges and faces with respect to the α-shape. They can be in the interior of a face that belongs or does not belong to the α-shape. They can be singular/regular, that is be on the boundary of the α-shape, but not incident/incident to a triangle of the α-complex.

Finally, it provides a function to determine the α-value such that the α-shape satisfies the following two properties, or at least the second one if there is no such α that both are satisfied :

(i) The number of components equals a number of your choice and
(ii) all data points are either on the boundary or in the interior of the regularized version of the α-shape (no singular edges).

The current implementation is static, that is after its construction points cannot be inserted or removed.

A.3 Concepts and Models

We currently do not specify concepts for the underlying triangulation type. Models that work for a basic alpha-shape are the classes *CGAL::Delaunay_triangulation_2* and *CGAL::Triangulation_hierarchy_2* templatized with a Delaunay triangulation. A model that works for a weighted alpha-shape is the class *CGAL::Regular_triangulation_2*.

The triangulation needs a geometric traits class as argument. The requirements of this class are described in the concept *CGAL::AlphaShapeTraits_2* for which the classes *CGAL::Alpha_shape_euclidean_traits_2* and *CGAL::Weighted_alpha_shape_euclidean_traits_2* are models.

There are no requirements on the triangulation data structure. However it must be parameterized with vertex and face classes, which are model of the concepts *AlphaShapeVertex_2* and *AlphaShapeFace_2*, by default the classes *CGAL::Alpha_shape_vertex_base_2<Gt>* and *CGAL::Alpha_shape_face_base_2<Tf>*.

A.4 Examples

A.4.1 Example for Basic Alpha-Shapes

The basic alpha shape needs a Delaunay triangulation as underlying triangulation *Dt*. The Delaunay triangulation class is parameterized with a geometric and a triangulation data structure traits.

For the geometric traits class we can use the *CGAL::Alpha_shape_traits_2* class, which in turn is parameterized with a kernel.

For the triangulation data structure traits, we have to choose the vertex and face classes needed for alpha shapes, namely *CGAL::Alpha_shape_vertex_base_2<Gt, Dv>* and *CGAL::Alpha_shape_face_base_2<Gt,Df>*. As default vertex and face type they use *CGAL::Triangulation_vertex_base_2<Gt>* and *CGAL::Triangulation_face_base_2<Gt>* respectively.

The following code sniplet shows how to obtain a basic alpha shape type.

```
typedef CGAL::Cartesian<double> K;
typedef CGAL::Alpha_shape_euclidean_traits_2<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_2<Gt> Av;

typedef CGAL::Triangulation_face_base_2<Gt> Tf;
typedef CGAL::Alpha_shape_face_base_2<Gt,Tf> Af;
```

```
typedef CGAL::Triangulation_default_data_structure_2<Gt,Av,Af> Tds;
typedef CGAL::Delaunay_triangulation_2<Gt,Tds> Dt;
typedef CGAL::Alpha_shape_2<Dt> Alpha_shape_2;
```

A.4.2 Example for Basic Alpha-Shapes with Many Points

When the input data set is huge, say more than 10.000 points, it pays off to use a triangulation hierarchy. It has the same API as the Delaunay triangulation and differs only in the types of the vertices and faces.Therefore, the only part that changes are the typedefs in the beginning.

```
typedef CGAL::Cartesian<double> K;
typedef CGAL::Alpha_shape_euclidean_traits_2<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_2<Gt> Avb;
typedef CGAL::Triangulation_hierarchy_vertex_base_2<Avb> Av;

typedef CGAL::Triangulation_face_base_2<Gt> Tf;
typedef CGAL::Alpha_shape_face_base_2<Gt,Tf> Af;

typedef CGAL::Triangulation_default_data_structure_2<Gt,Av,Af> Tds;
typedef CGAL::Delaunay_triangulation_2<Gt,Tds> Dt;
typedef CGAL::Triangulation_hierarchy_2<Dt> Ht;
typedef CGAL::Alpha_shape_2<Ht> Alpha_shape_2;
```

A.4.3 Example for Weighted Alpha-Shapes

A weighted alpha shape, needs a regular triangulation as underlying triangulation Dt, and it needs a particular face class, namely *CGAL::Regular_triangulation_face_base_2<Gt>*. Note that there is no special weighted alpha shape class.

```
typedef CGAL::Cartesian<double> K;
typedef CGAL::Weighted_alpha_shape_euclidean_traits_2<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_2<Gt> Av;

typedef CGAL::Regular_triangulation_face_base_2<Gt> Rf;
typedef CGAL::Alpha_shape_face_base_2<Gt,Rf>  Af;

typedef CGAL::Triangulation_default_data_structure_2<Gt,Av,Af> Tds;
typedef CGAL::Regular_triangulation_2<Gt,Tds> Rt;
typedef CGAL::Alpha_shape_2<Rt> Alpha_shape_2;
```

2D Alpha Shapes (reference manual)

This chapter presents a framework for alpha shapes. The description is based on the articles [EM94, Ede92]. Alpha shapes are the generalization of the convex hull of a point set. Let S be a finite set of points in $\mathbb{R}^d$, $d = 2, 3$ and α a parameter with $0 \leq \alpha \leq \infty$. For $\alpha = \infty$, the α-shape is the convex hull of S. As α decreases, the α-shape shrinks and develops cavities, as soon as a sphere of radius $\sqrt{\alpha}$ can be put inside. Finally, for $\alpha = 0$, the α-shape is the set S itself.

We distinguish two versions of alpha shapes, one is based on the Delaunay triangulation and the other on its generalization, the regular triangulation, replacing the natural distance by the power to weighted points. The metric used determines an underlying triangulation of the alpha shape and thus, the version computed. The *classic alpha shape* (cf. A.4.1) is associated with the Delaunay triangulation. The *weighted alpha shape* (cf. A.4.3) is associated with the regular triangulation.

There is a close connection between alpha shapes and the underlying triangulations. More precisely, the α-complex of S is a subcomplex of this triangulation of S, containing the α-exposed k-simplices, $0 \leq k \leq d$. A simplex is α-exposed, if there is an open disk (resp. ball) of radius $\sqrt{\alpha}$ through the vertices of the simplex that does not contain any other point of S, for the metric used in the computation of the underlying triangulation. The corresponding α-shape is defined as the underlying interior space of the α-complex.

In general, an α-complex is a non-connected and non-pure polytope, it means, that one k-simplex, $0 \leq k \leq d-1$ is not necessary adjacent to a $(k+1)$-simplex.

The α-shapes of S form a discrete family, even though they are defined for all real numbers α with $0 \leq \alpha \leq \infty$. Thus, we can represent the entire family of α-shapes of S by the underlying triangulation of S. In this representation each k-simplex of the underlying triangulation is associated with an interval that specifies for which values of α the k-simplex belongs to the α-shape. Relying on this result, the family of α-shapes can be computed efficiently and relatively easily. Furthermore, we can select an appropriate α-shape from a finite number of different α-shapes and corresponding α-values.

Concepts

Classes

CGAL::Alpha_shape_2<Dt>

Class

Definition

The class *Alpha_shape_2<Dt>* represents the family of α-shapes of points in a plane for *all* positive α. It maintains the underlying triangulation *Dt* which represents connectivity and order among its faces. Each k-dimensional face of the *Dt* is associated with an interval that specifies for which values of α the face belongs to the α-shape. There are links between the intervals and the k-dimensional faces of the triangulation.

Note that this class is at the same time used for *basic* and for *weighted* Alpha Shapes.

Inherits From

Dt

This class is the underlying triangulation class.

The modifying functions *insert* and *remove* will overwrite the inherited functions. At the moment, only the static version is implemented.

Types

Alpha_shape_2<Dt>:: Gt the alpha shape traits type.

it has to derive from a triangulation traits class. For example *Dt::Point* is a Point class.

typedef typename Gt::Coord_type

Coord_type ;; the number type for computation.

Alpha_shape_2<Dt>:: Alpha_iterator ;

A bidirectional and non-mutable iterator that allow to traverse the increasing sequence of different α-values.
Precondition: Its *value_type* is *Coord_type*

Alpha_shape_2<Dt>:: Alpha_shape_vertices_iterator ;

A bidirectional and non-mutable iterator that allow to traverse the vertices which belongs to the α-shape for the current α.
Precondition: Its *value_type* is *Dt : :Vertex_handle*

Alpha_shape_2<Dt>:: Alpha_shape_edges_iterator;

A bidirectional and non-mutable iterator that allow to traverse the edges which belongs to the α-shape for the current α.
Precondition: Its *value_type* is *Dt : :Edge*.

enum Classification_type { EXTERIOR, SINGULAR, REGULAR, INTERIOR};

Distinguishes the different cases for classifying a k-dimensional face of the underlying triangulation of the α-shape.
EXTERIOR if the face does not belong to the α-complex.
SINGULAR if the face belongs to the boundary of the α-shape, but is not incident to any 2-dimensional face of the α-complex
REGULAR if the face belongs to the boundary of the α-shape and is incident to a 2-dimensional face of the α-complex
INTERIOR if the face belongs to the α-complex, but does not belong to the boundary of the α-shape.

enum Mode { GENERAL, REGULARIZED};

In general, an alpha shape can be disconnected and contain many singular edges or vertices. Its regularized version is formed by the set of regular edges and their vertices.

Creation

Alpha_shape_2<Dt> A(Coord_type alpha = 0, Mode m = GENERAL);

Introduces an empty α-shape A for a positive α-value *alpha*.
Precondition: *alpha* ≥ 0.

```
template < class InputIterator >
Alpha_shape_2<Dt>  A( InputIterator first,
                      InputIterator last,
                      Coord_type alpha = 0,
                      Mode m = GENERAL)
```

Initializes the family of alpha-shapes with the points in the range $[first, last)$ and introduces an α-shape A for a positive α-value *alpha*.
Precondition: The *value_type* of *first* and *last* is *Point*.
alpha ≥ 0.

Operations

template < class InputIterator >
int *A.make_alpha_shape(InputIterator first, InputIterator last)*

Initialize the family of alpha-shapes with the points in the range [*first*, *last*). Returns the number of inserted points. If the function is applied to an non-empty family of alpha-shape, it is cleared before initialization.
Precondition: The *value_type* of *first* and *last* is *Point*.

void *A.clear()* Clears the structure.

Coord_type *A.set_alpha(Coord_type alpha)*

Sets the α-value to *alpha*. Returns the previous α-value.
Precondition: *alpha* ≥ 0.

Coord_type *A.get_alpha(void)*

Returns the current α-value.

Coord_type *A.get_nth_alpha(int n)*

Returns the n-th alpha-value, sorted in an increasing order.
Precondition: $n <$ number of alphas.

int *A.number_of_alphas()*

Returns the number of different alpha-values.

Mode *A.set_mode(Mode m = GENERAL)*

Sets A to its general or regularized version. Returns the previous mode.

Mode *A.get_mode(void)*

Returns whether A is general or regularized.

Alpha_shape_vertices_iterator

A.alpha_shape_vertices_begin()

Starts at an arbitrary finite vertex which belongs to the α-shape for the current α.

Alpha_shape_vertices_iterator

A.alpha_shape_vertices_end()

Past-the-end iterator.

Alpha_shape_edges_iterator

A.alpha_shape_edges_begin()

Starts at an arbitrary finite edge which belongs to the α-shape for the current α. In regularised mode, edges are represented as a pair (f,i), where f is an interior face of the α-shape.

Alpha_shape_edges_iterator

A.alpha_shape_edges_end()

Past-the-end iterator.

Predicates

Classification_type

A.classify(Point p, Coord_type alpha = get_alpha())

Locates a point p in the underlying triangulation and Classifies the associated k-face with respect to A.

Classification_type

A.classify(Face_handle f, Coord_type alpha = get_alpha())

Classifies the face f of the underlying triangulation with respect to A.

Classification_type

A.classify(Edge e, Coord_type alpha = get_alpha())

Classifies the edge e of the underlying triangulation with respect to A.

Classification_type

A.classify(Face_handle f, int i, Coord_type alpha = get_alpha())

Classifies the edge of the face f opposite to the vertex with index i of the underlying triangulation with respect to A.

Classification_type

A.classify(Vertex_handle v, Coord_type alpha = get_alpha())

Classifies the vertex v of the underlying triangulation with respect to A.

Traversal of the α-Values

Alpha_iterator

A.alpha_begin()

Returns an iterator that allows to traverse the sorted sequence of α-values of the family of alpha shapes.

Alpha_iterator

A.alpha_end() Returns the corresponding past-the-end iterator.

Alpha_iterator

A.alpha_find(Coord_type alpha)

Returns an iterator pointing to an element with α-value *alpha*, or the corresponding past-the-end iterator if such an element is not found.

Alpha_iterator

A.alpha_lower_bound(Coord_type alpha)

Returns an iterator pointing to the first element with α-value not less than *alpha*.

Alpha_iterator

A.alpha_upper_bound(Coord_type alpha)

Returns an iterator pointing to the first element with α-value greater than *alpha*.

Operations

int *A.number_of_solid_components(Coord_type alpha = get_alpha())*

Returns the number of solid components of A, that is, the number of components of its regularized version.

Alpha_iterator

A.find_optimal_alpha(int nb_components)

Returns an iterator pointing to the first element with α-value such that A satisfies the following two properties :
nb_components equals the number of solid components and all data points are either on the boundary or in the interior of the regularized version of A.
If no such value is found, the iterator points to the first element with α-value such that A satisfies the second property.

I/O

The I/O operators are defined for *iostream*, and for the window stream provided by CGAL. The format for the iostream is an internal format.

#include <CGAL/IO/io.h>

ostream& *ostream& os << A*

Inserts the alpha shape A for the current α-value into the stream *os*.
Precondition: The insert operator must be defined for *Point*.

#include <CGAL/IO/Window_stream.h>

#include <CGAL/IO/alpha_shapes_2_window_stream.h>

Window_stream&

Window_stream& W << A

Inserts the alpha shape A for the current α-value into the window stream W.
Precondition: The insert operator must be defined for *Point* and *Segment*.

Implementation

The set of intervals associated with the k-dimensional faces of the underlying triangulation are stored in *multimaps*.

The cross links between the intervals and the k-dimensional faces of the triangulation are realized using methods in the k-dimensional faces themselves.

A.alpha find uses linear search, while *A.alpha lower bound* and *A.alpha upper bound* use binary search. *A.number of solid components* performs a graph traversal and takes time linear in the number of faces of the underlying triangulation. *A.find optimal alpha* uses binary search and takes time $O(\ n\ \log\ n\ )$, where n is the number of points.

Concept

AlphaShapeFace_2

Definition

Refines

TriangulationFaceBase_2.

Types

AlphaShapeFace_2:: Interval_3 — A container type to get (and put) the three special values $(\alpha_1, \alpha_2, \alpha_3)$ associated with an alpha shape edge.

AlphaShapeFace_2:: Coord_type

A type to hold a coordinate type class. The type must provide a copy constructor, assignment, comparison operators, negation, multiplication, division and allow the declaration and initialization with a small integer constant (cf. requirements for number types). An obvious choice would be coordinate type of the point class.
Precondition: Let's us precise that *Coord_type* has to be the same as the one used by *Gt*, already used by the underlying triangulation *Dt*, and by her face base *Df*.

Creation

advanced

AlphaShapeFace_2 f; — default constructor.

AlphaShapeFace_2 f(void v0, void* v1, void* v2);*

constructor setting the incident vertices.

AlphaShapeFace_2 f(void v0, void* v1, void* v2, void* n0, void* n1, void* n2);*

constructor setting the incident vertices and the neighboring faces.

advanced

Access Functions

Interval_3 *f.get_ranges(int i)*

returns the interval associated with the edge indexed with i, which contains three alpha values $\alpha_1 \leq \alpha_2 \leq \alpha_3$, such as for α between α_1 and α_2, the edge indexed with i is attached but singular, for α between α_2 and α_3, the edge is regular, and for α greater than α_3, the edge is interior.

Coord_type *f.get_alpha()* return the alpha value, under which the alpha shape contains the face.

Modifiers

advanced

void *f.set_ranges(int i, Interval_3 V)*

sets the interval associated with the edge indexed with i, which contains three alpha values $\alpha_1 \leq \alpha_2 \leq \alpha_3$, such as for α between α_1 and α_2, the edge indexed with i is attached but singular, for α between α_2 and α_3, the edge is regular, and for α greater than α_3, the edge is interior.

void *f.set_alpha(Coord_type A)*

sets the alpha value, under which the alpha shape contains the face.

advanced

Class

CGAL::Alpha_shape_face_base_2<AlphaShapeTraits_2, TriangulationFaceBase_2>

Definition

The class *Alpha_shape_face_base_2<AlphaShapeTraits_2, TriangulationFaceBase_2>* is the default model for the concept *AlphaShapeFace_2*.

#include <Alpha_shape_face_base_2.h>

Is Model for the Concept

AlphaShapeFace_2

Inherits From

TriangulationFaceBase_2

AlphaShapeTraits_2

Concept

Definition

A model of the concept AlphaShapeTraits_2 must provide the following predicate and operations in addition to the requirements for the underlying triangulation traits class. It means, the metric has to be euclidean for Delaunay triangulation or the power metric for regular triangulation.

Refines

TriangulationTraits_2

Types

AlphaShapeTraits_2:: Coord_type

A type to hold a coordinate type class. The type must provide a copy constructor, assignment, comparison operators, negation, multiplication, division and allow the declaration and initialization with a small integer constant (cf. requirements for number types).
Precondition: An obvious choice would be coordinate type of the point class.

Creation

Only a default constructor is required. Note that further constructors can be provided.

AlphaShapeTraits_2 t; A default constructor.

Constructions by function objects

Compute_squared_radius_2

t.compute_squared_radius_2_object()

Returns an object, which has to be able to compute the squared radius of the circle of the points *p0, p1, p2* or the squared radius of smallest circle of the points *p0, p1*, as *Coord_type* associated with *the metric used by Dt*.

Predicate by function object

Side_of_bounded_circle_2

t.side_of_bounded_circle_2_object()

Returns an object, which has to be able to compute the relative position of point *test* to the smallest circle of the points *p0, p1*, using *the same metric as Dt*.

CGAL::Alpha_shape_euclidean_traits_2<K>

Class

Definition

The class *Alpha_shape_euclidean_traits_2<K>* is the default model for the concept *AlphaShapeTraits_2* for the basic version of Alpha Shapes. *K* must be a kernel.

#include <Alpha_shape_euclidean_traits_2.h>

Refines

Triangulation_euclidean_traits_2<K, typename K::FT>

Is Model for the Concept

AlphaShapeTraits_2

Class

CGAL::Weighted_alpha_shape_euclidean_traits_2<K>

Definition

The class *Weighted_alpha_shape_euclidean_traits_2<K>* is the default model for the concept *AlphaShapeTraits_2* for the regular version of Alpha Shapes. *K* must be a kernel.

#include <Weighted_alpha_shape_euclidean_traits_2.h>

Refines

Regular_triangulation_euclidean_traits_2<K, typename K::FT>

Is Model for the Concept

AlphaShapeTraits_2

AlphaShapeVertex_2

Concept

Definition

Refines

TriangulationVertexBase_2.

Types

AlphaShapeVertex_2:: Coord_type

A type to hold a coordinate type class. The type must provide a copy constructor, assignment, comparison operators, negation, multiplication, division and allow the declaration and initialization with a small integer constant (cf. requirements for number types). An obvious choice would be coordinate type of the point class.
Precondition: *Coord_type* has to be the same as the one used by the *Gt*, already used by underlying triangulation *Dt*.

Creation

advanced

AlphaShapeVertex_2 v; default constructor.

AlphaShapeVertex_2 v(Point p);

constructor setting the point.

AlphaShapeVertex_2 v(Point p, void f);*

constructor setting the point associated to and an incident face.

advanced

Access Functions

std::pair< Coord_type, Coord_type >

v.get_range() returns two alpha values $\alpha_1 \leq \alpha_2$, such as for α between α_1 and α_2, the vertex is attached but singular, and for α upper α_2, the vertex is regular.

Modifiers

advanced

void *v.set_range(std::pair< Coord_type, Coord_type > I)*

sets the alpha values $\alpha_1 \leq \alpha_2$, such as for α between α_1 and α_2, the vertex is attached but singular, and for α upper α_2, the vertex is regular.

advanced

CGAL::Alpha_shape_vertex_base_2<AlphaShapeTraits_2>

Class

Definition

The class *Alpha_shape_vertex_base_2<AlphaShapeTraits_2>* is the default model for the concept *AlphaShapeVertex_2*.

#include <Alpha_shape_vertex_base_2.h>

Is Model for the Concept

AlphaShapeVertex_2

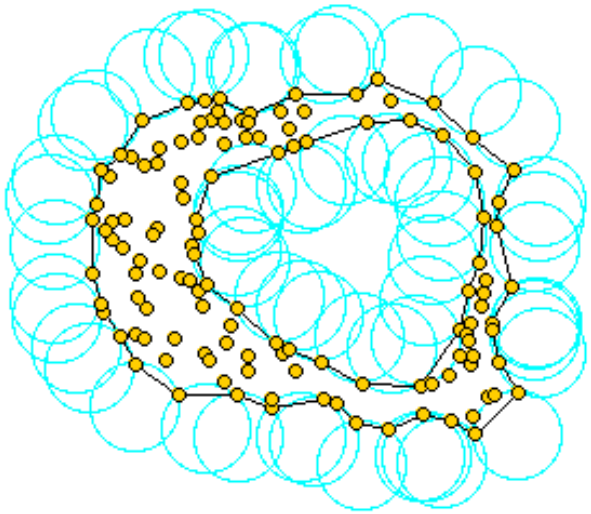
Inherits From

TriangulationVertexBase_2

Annexe B

Documentation CGAL : *Alpha-formes* 3D

3D Alpha Shapes (user manual)



Assume we are given a set S of points in 2D or 3D and we'd like to have something like "the shape formed by these points." This is quite a vague notion and there are probably many possible interpretations, the α-shape being one of them. Alpha shapes can be used for shape reconstruction from a dense unorganized set of data points. Indeed, an α-shape is demarcated by a frontier, which is a linear approximation of the original shape [BB97].

As mentionned in Edelsbrunner's and Mücke's paper [EM94], one can intuitively think of an α-shape as the following. Imagine a huge mass of ice-cream making up the space $\mathbb{R}^3$ and containing the points as "hard" chocolate pieces. Using one of these sphere-formed ice-cream spoons we carve out all parts of the ice-cream block we can reach without bumping into chocolate pieces, thereby even carving out holes in the inside (eg. parts not reachable by simply moving the spoon from the outside). We will eventually end up with a (not necessarily convex) object bounded by caps, arcs and points. If we now straighten all "round" faces to triangles and line segments, we have an intuitive description of what is called the α-shape of S. Here's an example for this process in 2D (where our ice-cream spoon is simply a circle) :

And what is α in the game ? α is the squared radius of the carving spoon. A very small value will allow us to eat up all of the ice-cream except the chocolate points themselves. Thus we already see that the α-shape degenerates to the point-set S for $\alpha \to 0$. On the other hand, a huge value of α will prevent us even from moving the spoon between two points since it's way too large. So we will never spoon up ice-cream lying in the inside of the convex hull of S, and hence the α-shape for $\alpha \to \infty$ is the convex hull of S.

B.1 Definitions

We distinguish two versions of alpha shapes. *Basic alpha shapes* are based on the Delaunay triangulation. *Weighted alpha shapes* are based on its generalization, the regular triangulation, replacing the euclidean distance by the power to weighted points.

There is a close connection between alpha shapes and the underlying triangulations. More precisely, the α-complex of S is a subcomplex of this triangulation of S, containing the α-exposed k-simplices, $0 \leq k \leq d$. A simplex is α-exposed, if there is an open disk (resp. ball) of radius $\sqrt{\alpha}$ through the vertices of the simplex that does not contain any other point of S, for the metric used in the computation of the underlying triangulation. The corresponding α-shape is defined as the underlying interior space of the α-complex (see [EM94]).

In general, an α-complex is a non-connected and non-pure polytope, it means, that one k-simplex, $0 \leq k \leq d-1$ is not necessary adjacent to a $(k+1)$-simplex.

The α-shapes of S form a discrete family, even though they are defined for all real numbers α with $0 \leq \alpha \leq \infty$. Thus, we can represent the entire family of α-shapes of S by the underlying triangulation of S. In this representation each k-simplex of the underlying triangulation is associated with an interval that specifies for which values of α the k-simplex belongs to the α-shape. Relying on this fact, the family of α-shapes can be computed efficiently and relatively easily. Furthermore, we can select an appropriate α-shape from a finite number of different α-shapes and corresponding α-values.

B.2 Functionality

The class *CGAL::Alpha_shape_3<Dt>* represents the family of α-shapes of points in a plane for *all* positive α. It maintains the underlying triangulation *Dt* which represents connectivity and order among squared radius of its faces. Each k-dimensional face of the *Dt* is associated with an interval that specifies for which values of α the cell belongs to the α-shape. There are links between the intervals and the k-dimensional faces of the triangulation.

The class *CGAL::Alpha_shape_3<Dt>* provides functions to set and get the current α-value, as well as an iterator that enumerates the α-values where the α-shape changes.

It provides iterators to enumerate the vertices and facets that are in the α-shape, and functions that allow to classify vertices, edges, facets and cells with respect to the α-shape. They can be in the interior of a cell that belongs or does not belong to the α-shape. They can be singular/regular, that is be on the boundary of the α-shape, but not incident/incident to a tetrahedron of the α-complex.

Finally, it provides a function to determine the α-value such that the α-shape satisfies the following two properties, or at least the second one if there is no such α that both are satisfied :

(i) The number of components equals the number of your choice and
(ii) all data points are either on the boundary or in the interior of the regularized version of the α-shape (no singular facets and edges).

The current implementation is static, that is after its construction points cannot be inserted or removed.

B.3 Concepts and Models

We currently do not specify concepts for the underlying triangulation type. Models that work for a basic alpha-shape are the classes *CGAL::Delaunay_triangulation_3* and *CGAL::Triangulation_hierarchy_3* templatized with a Delaunay triangulation. A model that works for a weighted alpha-shape is the class *CGAL::Regular_triangulation_3*.

The triangulation needs a geometric traits class as argument. The requirements of this class are described in the concept *CGAL::AlphaShapeTraits_3* for which the classes *CGAL::Alpha_shape_euclidean_traits_3* and *CGAL::Weighted_alpha_shape_euclidean_traits_3* are models.

There are no requirements on the triangulation data structure. However it must be parameterized with vertex and cell classes, which are model of the concepts *AlphaShapeVertex_3* and *AlphaShapeCell_3*, by default the classes *CGAL::Alpha_shape_vertex_base_3<Gt>* and *CGAL::Alpha_shape_Cell_base_3<Tf>*.

B.4 Examples

B.4.1 Example for Basic Alpha-Shapes

The basic alpha shape needs a Delaunay triangulation as underlying triangulation *Dt*. The Delaunay triangulation class is parameterized with a geometric and a triangulation data structure traits.

For the geometric traits class we can use the *CGAL::Alpha_shape_traits_3* class, which in turn is parameterized with a kernel.

For the triangulation data structure traits, we have to choose the vertex and cell classes needed for alpha shapes, namely *CGAL::Alpha_shape_vertex_base_3<Gt, Dv>* and *CGAL::Alpha_shape_cell_base_3<Gt,Df>*. As default vertex and cell type they use *CGAL::Triangulation_vertex_base_3<Gt>* and *CGAL::Triangulation_cell_base_3<Gt>* respectively.

The following code sniplet shows how to obtain a basic alpha shape type.

```
typedef CGAL::Cartesian<double> K;
typedef CGAL::Alpha_shape_euclidean_traits_3<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_3<Gt> Av;

typedef CGAL::Triangulation_cell_base_3<Gt> Tc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Tc> Ac;
```

```
typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Delaunay_triangulation_3<Gt,Tds> Dt;
typedef CGAL::Alpha_shape_3<Dt> Alpha_shape_3;
```

B.4.2 Example for Basic Alpha-Shapes with Many Points

When the input data set is huge, say more than 10.000 points, it pays off to use a triangulation hierarchy. It has the same API as the Delaunay triangulation and differs only in the types of the vertices and cells.Therefore, the only part that changes are the typedefs in the beginning.

```
typedef CGAL::Cartesian<double> K;
typedef CGAL::Alpha_shape_euclidean_traits_3<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_3<Gt> Avb;
typedef CGAL::Triangulation_hierarchy_vertex_base_3<Avb> Av;

typedef CGAL::Triangulation_cell_base_3<Gt> Tc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Tc> Ac;

typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Delaunay_triangulation_3<Gt,Tds> Dt;
typedef CGAL::Triangulation_hierarchy_3<Dt> Ht;
typedef CGAL::Alpha_shape_3<Ht> Alpha_shape_3;
```

B.4.3 Example for Weighted Alpha-Shapes

A weighted alpha shape, needs a regular triangulation as underlying triangulation Dt, and it needs a particular cell class, namely *CGAL::Regular_triangulation_cell_base_3<Gt>*. Note that there is no special weighted alpha shape class.

```
typedef CGAL::Cartesian<double> K;
typedef CGAL::Weighted_alpha_shape_euclidean_traits_3<K> Gt;

typedef CGAL::Alpha_shape_vertex_base_3<Gt> Av;

typedef CGAL::Regular_triangulation_cell_base_3<Gt> Rc;
typedef CGAL::Alpha_shape_cell_base_3<Gt,Rc>  Ac;

typedef CGAL::Triangulation_data_structure_3<Gt,Av,Ac> Tds;
typedef CGAL::Regular_triangulation_3<Gt,Tds> Rt;
typedef CGAL::Alpha_shape_3<Rt> Alpha_shape_3;
```

3D Alpha Shapes (reference manual)

This chapter presents a framework for alpha shapes. The description is based on the articles [EM94, Ede92]. Alpha shapes are the generalization of the convex hull of a point set. Let S be a finite set of points in $\mathbb{R}^d$, $d = 2, 3$ and α a parameter with $0 \leq \alpha \leq \infty$. For $\alpha = \infty$, the α-shape is the convex hull of S. As α decreases, the α-shape shrinks and develops cavities, as soon as a sphere of radius $\sqrt{\alpha}$ can be put inside. Finally, for $\alpha = 0$, the α-shape is the set S itself.

We distinguish two versions of alpha shapes, one is based on the Delaunay triangulation and the other on its generalization, the regular triangulation, replacing the natural distance by the power to weighted points. The metric used determines an underlying triangulation of the alpha shape and thus, the version computed. In one hand, there is the *classic alpha shapes* (cf. B.4.1) associated with the Delaunay triangulations, in the other hand, the *weighted alpha shapes* (cf. B.4.3) associated with the regular triangulations.

There is a close connection between alpha shapes and the underlying triangulations. More precisely, the α-complex of S is a subcomplex of this triangulation of S, containing the α-exposed k-simplices, $0 \leq k \leq d$. A simplex is α-exposed, if there is an open disk (resp. ball) of radius $\sqrt{\alpha}$ through the vertices of the simplex that does not contain any other point of S, for the metric used in the computation of the underlying triangulation. The corresponding α-shape is defined as the underlying interior space of the α-complex.

In general, an α-complex is a non-connected and non-pure polytope, it means, that one k-simplice, $0 \leq k \leq d-1$ is not necessary adjacent to a $(k+1)$-simplice.

The α-shapes of S form a discrete family, even though they are defined for all real numbers α with $0 \leq \alpha \leq \infty$. Thus, we can represent the entire family of α-shapes of S by the underlying triangulation of S. In this representation each k-simplex of the underlying triangulation is associated with an interval that specifies for which values of α the k-simplex belongs to the α-shape. Relying on this result, the family of α-shapes can be computed efficiently and relatively easily. Furthermore, we can select an appropriate α-shape from a finite number of different α-shapes and corresponding α-values.

Concepts

Classes

CGAL::Alpha_shape_3<Dt>

Class

Definition

The class *Alpha_shape_3<Dt>* represents the family of α-shapes of points in the 3D space for *all* positive α. It maintains the underlying triangulation *Dt* which represents connectivity and order among its faces. Each k-dimensional face of the *Dt* is associated with an interval that specifies for which values of α the face belongs to the α-shape. There are links between the intervals and the k-dimensional faces of the triangulation.

Note that this class is at the same time used for *basic* and for *weighted* Alpha Shapes.

Inherits From

Dt

This class is the underlying triangulation class.

The modifying functions *insert* and *remove* will overwrite the inherited functions. At the moment, only the static version is implemented.

Types

Alpha_shape_3<Dt>:: Gt the alpha shape traits type.

it has to derive from a triangulation traits class. For example *Dt::Point* is a Point class.

typedef typename Gt::Coord_type

Coord_type ;; the number type for computation.

Alpha_shape_3<Dt>:: Alpha_iterator ;

A bidirectional and non-mutable iterator that allow to traverse the increasing sequence of different α-values.
Precondition: Its *value_type* is *Coord_type*

Alpha_shape_3<Dt>:: Alpha_shape_vertices_iterator ;

A bidirectional and non-mutable iterator that allow to traverse the vertices which belongs to the α-shape for a fixed α.
Precondition: Its *value_type* is *Dt : :Vertex_handle*

Alpha_shape_3<Dt>:: Alpha_shape_facets_iterator;

A bidirectional and non-mutable iterator that allow to traverse the facets which belongs to the α-shape for a fixed α.
Precondition: Its *value_type* is *Dt : :Facet*

enum Classification_type { EXTERIOR, SINGULAR, REGULAR, INTERIOR};

Distinguishes the different cases for classifying a k-dimensional face of the underlying triangulation of the α-shape.
EXTERIOR if the face does not belong to the α-complex.
SINGULAR if the face belongs to the boundary of the α-shape, but is not incident to any 3-dimensional face of the α-complex
REGULAR if the face belongs to the boundary of the α-shape and is incident to a 3-dimensional face of the α-complex
INTERIOR if the face belongs to the α-complex, but does not belong to the boundary of the α-shape

enum Mode { GENERAL, REGULARIZED};

In general, an alpha shape can be disconnected and contain many singular facets or vertices. Its regularized version is formed by the set of regular facets and their vertices

Creation

Alpha_shape_3<Dt> A(Coord_type alpha = 0, Mode m = REGULARIZED);

Introduces an empty α-shape A for a positive α-value *alpha*.
Precondition: *alpha* ≥ 0.

```
template < class InputIterator >
Alpha_shape_3<Dt>  A( InputIterator first,
                      InputIterator last,
                      Coord_type alpha = 0,
                      Mode m = REGULARIZED)
```

Initializes the family of alpha-shapes with the points in the range [*first*, *last*) and introduces an α-shape A for a positive α-value *alpha*.
Precondition: The *value_type* of *first* and *last* is *Point*.
alpha ≥ 0.

Operations

template < class InputIterator >
int *A.make_alpha_shape(InputIterator first, InputIterator last)*

Initialize the family of alpha-shapes with the points in the range [*first*, *last*). Returns the number of inserted points. If the function is applied to an non-empty family of alpha-shape, it is cleared before initialization.
Precondition: The *value_type* of *first* and *last* is *Point*.

void *A.clear()* Clears the structure.

Coord_type *A.set_alpha(Coord_type alpha)*

Sets the α-value to *alpha*. Returns the previous α-value.
Precondition: *alpha* ≥ 0.

Coord_type *A.get_alpha(void)*

Returns the current α-value.

Coord_type *A.get_nth_alpha(int n)*

Returns the n-th alpha-value, sorted in an increasing order.
Precondition: $n <$ number of alphas.

int *A.number_of_alphas()*

Returns the number of different alpha-values.

Mode *A.set_mode(Mode m = REGULARIZED)*

Sets A to its general or regularized version. Returns the previous mode.

Mode *A.get_mode(void)*

Returns whether A is general or regularized.

Alpha_shape_vertices_iterator

A.alpha_shape_vertices_begin()

Starts at an arbitrary finite vertex which belongs to the α-shape for a fixed α.

Alpha_shape_vertices_iterator

A.alpha_shape_vertices_end()

Past-the-end iterator.

Alpha_shape_facets_iterator

A.alpha_shape_facets_begin()

Starts at an arbitrary finite facet which belongs to the α-shape for a fixed α. In regularised mode, facets are represented as a pair (f,i), where f is an interior cell of the α-shape.

Alpha_shape_facets_iterator

A.alpha_shape_facets_end()

Past-the-end iterator.

Predicates

Classification_type

A.classify(Point p, Coord_type alpha = get_alpha())

Locates a point p in the underlying triangulation and Classifies the associated k-face with respect to A.

Classification_type

A.classify(Cell_handle f, Coord_type alpha = get_alpha())

Classifies the cell f of the underlying triangulation with respect to A.

Classification_type

A.classify(Facet f, Coord_type alpha = get_alpha())

Classifies the facet e of the underlying triangulation with respect to A.

Classification_type

A.classify(Cell_handle f, int i, Coord_type alpha = get_alpha())

Classifies the facet of the cell f opposite to the vertex with index i of the underlying triangulation with respect to A.

Classification_type

A.classify(Vertex_handle v, Coord_type alpha = get_alpha())

Classifies the vertex v of the underlying triangulation with respect to A.

Traversal of the α-Values

Alpha_iterator

A.alpha_begin()

Returns an iterator that allows to traverse the sorted sequence of α-values of the family of alpha shapes.

Alpha_iterator

A.alpha_end() Returns the corresponding past-the-end iterator.

Alpha_iterator

A.alpha_find(Coord_type alpha)

Returns an iterator pointing to an element with α-value *alpha*, or the corresponding past-the-end iterator if such an element is not found.

Alpha_iterator

A.alpha_lower_bound(Coord_type alpha)

Returns an iterator pointing to the first element with α-value not less than *alpha*.

Alpha_iterator

A.alpha_upper_bound(Coord_type alpha)

Returns an iterator pointing to the first element with α-value greater than *alpha*.

Operations

int *A.number_of_solid_components(Coord_type alpha = get_alpha())*

Returns the number of solid components of A, that is, the number of components of its regularized version.

Alpha_iterator

A.find_optimal_alpha(int nb_components)

Returns an iterator pointing to the first element with α-value such that A satisfies the following two properties :
nb_components equals the number of solid components and all data points are either on the boundary or in the interior of the regularized version of A.
If no such value is found, the iterator points to the first element with α-value such that A satisfies the second property.

I/O

The I/O operators are defined for *iostream*, and for the window stream provided by CGAL. The format for the iostream is an internal format.

#include <CGAL/IO/io.h>

ostream& *ostream& os << A*

Inserts the alpha shape A for the current α-value into the stream *os*.
Precondition: The insert operator must be defined for *Point*.

#include <CGAL/IO/Geomview_stream.h>

#include <CGAL/IO/alpha_shape_geomview_ostream_3.h>

Geomview_stream&

Geomview_stream& W << A

Inserts the alpha shape A for the current α-value into the Geomview stream W.
Precondition: The insert operator must be defined for *Point* and *Triangle*.

Implementation

Note that this implementation do not provide functions to deal with edges, first because of single edges are not in any reconstruction, second because dealing with edges would slow down the wall initialization process.

The set of intervals associated with the k-dimensional faces of the underlying triangulation are stored in *multimaps*.

The cross links between the intervals and the k-dimensional faces of the triangulation are actually realized using methods in the k-dimensional faces themselves.

A.alpha find uses linear search, while *A.alpha lower bound* and *A.alpha upper bound* use binary search. *A.number of solid components* performs a graph traversal and takes time linear in the number of cells of the underlying triangulation. *A.find of optimal alpha* uses binary search and takes time $O(\ n\ \log\ n\ )$, where n is the number of points.

Concept

AlphaShapeCell_3

Definition

Refines

TriangulationCellBase_3.

Types

AlphaShapeCell_3:: Interval_3	A container type to get (and put) the three special values $(\alpha_1, \alpha_2, \alpha_3)$ associated with an alpha shape facet.
AlphaShapeCell_3:: Coord_type	A type to hold a coordinate type class. The type must provide a copy constructor, assignment, comparison operators, negation, multiplication, division and allow the declaration and initialization with a small integer constant (cf. requirements for number types). An obvious choice would be coordinate type of the point class. *Precondition*: Let's us precise that *Coord_type* has to be the same as the one used by *Gt*, already used by the underlying triangulation *Dt*, and by her cell base *Df*.

Creation

advanced

AlphaShapeCell_3 f; default constructor.

AlphaShapeCell_3 f(void v0, void* v1, void* v2, void* v3);*

constructor setting the incident vertices.

```
AlphaShapeCell_3 f( void* v0,
                    void* v1,
                    void* v2,
                    void* v3,
                    void* n0,
                    void* n1,
                    void* n2,
                    void* n3)
```

constructor setting the incident vertices and the neighboring cells.

advanced

Access Functions

Interval_3 *f.get_ranges(int i)*

returns the interval associated with the facet indexed with i, which contains three alpha values $\alpha_1 \leq \alpha_2 \leq \alpha_3$, such as for α between α_1 and α_2, the facet indexed with i is attached but singular, for α between α_2 and α_3, the facet is regular, and for α greater than α_3, the facet is interior.

Coord_type *f.get_alpha()* return the alpha value, under which the alpha shape contains the cell.

Modifiers

advanced

void *f.set_ranges(int i, Interval_3 V)*

sets the interval associated with the facet indexed with i, which contains three alpha values $\alpha_1 \leq \alpha_2 \leq \alpha_3$, such as for α between α_1 and α_2, the facet indexed with i is attached but singular, for α between α_2 and α_3, the facet is regular, and for α greater than α_3, the facet is interior.

void *f.set_alpha(Coord_type A)*

sets the alpha value, under which the alpha shape contains the cell.

advanced

Class

CGAL::Alpha_shape_cell_base_3<AlphaShapeTraits_3, TriangulationCellBase_3>

Definition

The class *Alpha_shape_cell_base_3<AlphaShapeTraits_3, TriangulationCellBase_3>* is the default model for the concept *AlphaShapeCell_3*.

#include <Alpha_shape_cell_base_3.h>

Is Model for the Concept

AlphaShapeCell_3

Inherits From

TriangulationCellBase_3

AlphaShapeTraits_3

Concept

Definition

A model of the concept AlphaShapeTraits_3 must provide the following predicate and operations in addition to the requirements for the underlying triangulation traits class. It means, the metric has to be euclidean for Delaunay triangulation or the power metric for regular triangulation.

Refines

TriangulationTraits_3

Types

AlphaShapeTraits_3:: Coord_type

A type to hold a coordinate type class. The type must provide a copy constructor, assignment, comparison operators, negation, multiplication, division and allow the declaration and initialization with a small integer constant (cf. requirements for number types).
Precondition: An obvious choice would be coordinate type of the point class.

Creation

Only a default constructor is required. Note that further constructors can be provided.

AlphaShapeTraits_3 t; A default constructor.

Constructions by function objects

Compute_squared_radius_3

t.compute_squared_radius_3_object()

Returns an object, which has to be able to compute the squared radius of the circle of the points *p0, p1, p2, p3* or the squared radius of the smallest circle of the points *p0, p1, p2* or the squared radius of smallest circle of the points *p0, p1*, as *Coord_type* associated with the *metric used by Dt*.

Predicate by function object

Side_of_bounded_circle_3

t.side_of_bounded_circle_3_object()

Returns an object, which has to be able to compute the relative position of point *test* to the smallest circle of the points *p0, p1, p2* or to the smallest circle of the points *p0, p1*, using *the same metric as Dt*.

CGAL::Alpha_shape_euclidean_traits_3<K>

Class

Definition

The class *Alpha_shape_euclidean_traits_3<K>* is the default model for the concept *AlphaShapeTraits_3* for the basic version of Alpha Shapes. *K* must be a kernel.

#include <Alpha_shape_euclidean_traits_3.h>

Refines

Triangulation_euclidean_traits_3<K, typename K::FT>

Is Model for the Concept

AlphaShapeTraits_3

Class

CGAL::Weighted_alpha_shape_euclidean_traits_3<K>

Definition

The class *Weighted_alpha_shape_euclidean_traits_3<K>* is the default model for the concept *AlphaShapeTraits_3* for the regular version of Alpha Shapes. *K* must be a kernel.

#include <Weighted_alpha_shape_euclidean_traits_3.h>

Refines

Regular_triangulation_euclidean_traits_3<K, typename K::FT>

Is Model for the Concept

AlphaShapeTraits_3

AlphaShapeVertex_3

Concept

Definition

Refines

TriangulationVertexBase_3.

Types

AlphaShapeVertex_3:: Coord_type

A type to hold a coordinate type class. The type must provide a copy constructor, assignment, comparison operators, negation, multiplication, division and allow the declaration and initialization with a small integer constant (cf. requirements for number types). An obvious choice would be coordinate type of the point class.
Precondition: Let's us precise that *Coord_type* has to be the same as the one used by the *Gt*, already used by underlying triangulation *Dt*.

Creation

advanced

AlphaShapeVertex_3 v; default constructor.

AlphaShapeVertex_3 v(Point p);

constructor setting the point associated to.

AlphaShapeVertex_3 v(Point p, void c);*

constructor setting the point associated to and an incident cell.

advanced

Access Functions

std::pair< Coord_type, Coord_type >

v.get_range() returns two alpha values $\alpha_1 \leq \alpha_2$, such as for α between α_1 and α_2, the vertex is attached but singular, and for α upper α_2, the vertex is regular.

Modifiers

advanced

void *v.set_range(std::pair< Coord_type, Coord_type > I)*

sets the alpha values $\alpha_1 \leq \alpha_2$, such as for α between α_1 and α_2, the vertex is attached but singular, and for α upper α_2, the vertex is regular.

advanced

CGAL::Alpha_shape_vertex_base_3<AlphaShapeTraits_3>

Class

Definition

The class *Alpha_shape_vertex_base_3<AlphaShapeTraits_3>* is the default model for the concept *AlphaShapeVertex_3*.

#include <Alpha_shape_vertex_base_3.h>

Is Model for the Concept

AlphaShapeVertex_3

Inherits From

TriangulationVertexBase_3

Bibliographie

[AB99] Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. *Discrete Comput. Geom.*, 22(4) :481–504, 1999.

[ABE98] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the β-skeleton : Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60 :125–135, 1998.

[ACDL00] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 213–222, 2000.

[AGJ00] Udo Adamy, Joachim Giesen, and Matthias John. The λ-complex and surface reconstruction. In *Abstracts 16th European Workshop Comput. Geom.*, pages 14–17. Ben-Gurion University of the Negev, 2000.

[Alf89] P. Alfeld. Scattered data interpolation in three or more variables. *Mathematical Methods in Computer Aided Geometric Design*, pages 1–34, 1989.

[Ame97] N. Amenta. Computational geometry software. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 52, pages 951–960. CRC Press LLC, Boca Raton, FL, 1997.

[Att98] D. Attali. r-regular shape reconstruction from unorganized points. *Comput. Geom. Theory Appl.*, 10 :239–247, 1998.

[BB97] Fausto Bernardini and Chandrajit L. Bajaj. Sampling and reconstructing manifolds using alpha–shapes. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 193–198, 1997.

[BBCS99] F. Bernardini, C. L. Bajaj, J. Chen, and D. R. Schikore. Automatic reconstruction of 3D CAD models from digital scans. *Internat. J. Comput. Geom. Appl.*, 9 :327–369, 1999.

[BBX97] C. L. Bajaj, F. Bernadini, and G. Xu. Reconstructing surfaces and functions on surfaces from unorganized three-dimensional data. *Algorithmica*, 19(1–2) :243–261, September 1997.

[BC00] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 223–232, 2000.

[BC01] Jean-Daniel Boissonnat and Frédéric Cazals. Coarse-to-fine surface simplification with geometric guarantees. In A. Chalmers and T.-M. Rhyne, editors, *Eurographics'01*, Manchester, 2001. Blackwell.

[BCD+99] Jean-Daniel Boissonnat, Frédéric Cazals, Frank Da, Olivier Devillers, Sylvain Pion, François Rebufat, Monique Teillaud, and Mariette Yvinec. Programming with cgal : The example of triangulations. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 421–423, 1999.

[BDTY00] Jean-Daniel Boissonnat, Olivier Devillers, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 11–18, 2000.

[BMR+99] F. Bernardini, J. Mittleman, H. Rushmeir, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4), 1999.

[Boi84] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4) :266–286, 1984.

[Boi88] Jean-Daniel Boissonnat. Shape reconstruction from planar cross-sections. *Comput. Vision Graph. Image Process.*, 44(1) :1–29, October 1988.

[Bro97] J. L. Brown. Systems of coordinates associated with points scattered in the plane. *Comput. Aided Design*, 14 :547–559, 1997.

[BS96] G. Barequet and Micha Sharir. Piecewise-linear interpolation between polygonal slices. *Comput. Vision Graph. Image Process.*, 63, 1996. in press.

[BY95] Jean-Daniel Boissonnat and Mariette Yvinec. *Géométrie Algorithmique*. Ediscience international, Paris, 1995.

[CD99] S. W. Cheng and T. K. Dey. Improved constructions of delaunay based contour surfaces. In *Proc. Symposium on Solid Modeling and CAD/CAM Applications*, pages 322–323. ACM/SIGGRAPH, 1999.

[CGA01] The CGAL reference manual, Aout 2001. Release 2.3.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH 96*, pages 303–312, 1996.

[CSYL88] B. K. Choi, H. Y. Shin, Y. I. Yoon, and J. W. Lee. Triangulation of scattered data in 3d space. *Comput. Aided Design*, 20(5) :239–248, June 1988.

[DMR00] T. K. Dey, K. Mehlhorn, and E. A. Ramos. Curve reconstruction : Connecting dots with good reason. *Comput. Geom. Theory Appl.*, 15 :229–244, 2000.

[Ede92] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992.

[EKS83] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29 :551–559, 1983.

[EM94] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1) :43–72, January 1994.

[Far90] G. Farin. Surfaces over Dirichlet tesselations. *Comput. Aided Geom. Design*, 7 :281–292, 1990.

[FKU77] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20 :693–702, 1977.

[Gei93] Bernhard Geiger. *Three-dimensional modeling of human organs and its application to diagnosis and surgical planning*. Thèse de doctorat en sciences, Ecole Nationale Supérieure des Mines de Paris, France, 1993.

[Gei95] Bernhard Geiger. 3D Modeling using the Delaunay triangulation. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages V11–V12, 1995.

[GKS00] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized Delaunay triangulation. In *Eurographics*, 2000.

[GOS96] C. Gitlin, J. O'Rourke, and V. Subramanian. On reconstructing polyhedra from parallel slices. *Internat. J. Comput. Geom. Appl.*, 6(1) :103–122, 1996.

[GTHB92] Jingsheng Zheng Gabor T. Herman and Carolyn A. Bucholtz. Shape-based interpolation. *IEEE Computer Graphics & Applications*, pages 69–79, May 1992.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Comput. Graphics*, 26(2) :71–78, 1992. Proc. SIGGRAPH '92.

[HS00] Hisamoto Hiyoshi and Kokichi Sugihara. Voronoi-based interpolation with higher continuity. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 242–250, 2000.

[JSV97] Fabrice Jaillet, Behzad Shariat, and Denis Vandorpe. Periodic B–spline surface skinning of anatomic shapes. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 199–204, 1997.

[Ket98] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 146–154, 1998.

[Ket99] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Comput. Geom. Theory Appl.*, 13 :65–90, 1999.

[LC87] W. Lorensen and H. Cline. Marching cubes : a high resolution 3d surface construction algorithm. *Comput. Graph.*, 21(4) :163–170, 1987.

[Lev86] David Levin. Multidimensionnal reconstruction by set-valued approximations. *IMA Journal of Numerical Analysis*, 6 :173–184, 1986.

[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. In ACM, editor, *Proc. SIGGRAPH*, 2000. http ://graphics.Stanford.EDU/projects/mich/.

[Mel97] M. Melkemi. A-shapes and their derivatives. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 367–369, 1997.

[Mey94] D. Meyers. Multiresolutional tiling. *Comput. Graph. Forum*, 13(5) :325–340, December 1994.

[MSS92] David Meyers, Shelley Skinner, and Kenneth Sloan. Surfaces from contours. *ACM Trans. Graph.*, 11(3) :228–258, July 1992.

[Nul98] Stéphane Nullans. *Reconstruction tridimensionnelle de structures géologiques à partir de données hétérogènes.* Thèse de doctorat en sciences, université de Nice-Sophia Antipolis, France, 1998.

[oM86] US Nationale Library of Medecine, editor. *Visible human project*, 1986. http ://www.nlm.nih.gov/research/visible/visible_human.html.

[OPC96] J. M. Oliva, M. Perrin, and S. Coquillart. 3d reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram. *Comput. Graph. Forum*, 15(3) :397–408, 1996.

[PB01] Sylvain Petitjean and Edmond Boyer. Regular and non-regular point sets : properties and reconstruction. *Comput. Geom. Theory Appl.*, 19 :101–126, 2001.

[Pio99] Sylvain Pion. *De la géométrie algorithmique au calcul géométrique.* Thèse de doctorat en sciences, Université de Nice-Sophia Antipolis, France, 1999. TU-0619.

[Sch98] Stefan Schirra. Precision and robustness issues in geometric computation. In *Handbook of Computational Geometry*. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[Set96] J. A. Sethian. *Level Set Methods*. Cambridge University Press, 1996.

[She96] J. R. Shewchuk. Triangle : Engineering a 2d quality mesh generator and Delaunay triangulator. In *First Workshop on Applied Computational Geometry*. Association for Computing Machinery, May 1996.

[She00] Jonathan R. Shewchuk. Mesh generation for domains with small angles. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 1–10, 2000.

[Sib81] R. Sibson. A brief description of natural neighbour interpolation. In Vic Barnet, editor, *Interpreting Multivariate Data*, pages 21–36. John Wiley & Sons, Chichester, 1981.

[Str97] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, 3rd edition, 1997.

[TKDH01] J. Giesen T. K. Dey and J. Hudson. Delaunay based shape reconstruction from large data. In *Proc. IEEE Symp. in Parallel and Large Data Visu. and Graphics*, pages 19–27, 2001.

[Vel92] R. C. Veltkamp. *Closed object boundaries from scattered points*. PhD thesis, Center for Mathematics and Computer Science, Amsterdam, 1992.

[Vel97] Remco C. Veltkamp. Generic programming in CGAL, the Computational Geometry Algorithms Library. In F. Arbab and Ph. Slusallek, editors, *Proceedings of the 6th Eurographics Workshop on Programming Paradigms in Graphics, Budapest, Hungary, 8 September 1997*, pages 127–138, 1997.

[Wat92] Davis F. Watson. *Contouring : A Guide to the Analysis and Display of Spatial Data*. Pergamon, 1992.

[Wat94] D. F. Watson. *nngridr : An implementation of natural neighbor interpolation*. David Watson, 1994.

[ZJH87] M. J. Zyda, A. R. Jones, and P. G. Hogan. Surface construction from planar contours. *Computers and Graphics*, 11 :393–408, 1987.

[ZOMK00] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit nonparametric shape reconstruction from unorganized points using a variational level set method. *Computer Vision and Image Understanding*, 80(3), 2000.

Résumé

Pour de nombreuses applications informatiques, il est nécessaire d'interpréter des données échantillonnées et de fournir une représentation aussi correcte que possible des objets dont elles proviennent. Entre autres, on peut penser à l'imagerie médicale, au *reverse engineering*, à des applications de réalité virtuelle ou encore aux effets spéciaux pour le cinéma. Le problème étudié dans le cadre de cette thèse peut être formulé ainsi : à partir d'un ensemble $\mathcal{S}$ de points 3D échantillonnés sur un objet $\mathcal{O}$, il s'agit de fournir un modèle géométrique de la surface délimitant $\mathcal{O}$.

Dans un premier temps, on détaille l'implantation d'une solution classique en Géométrie algorithmique dans le cadre du progiciel CGAL (`http://www.cgal.org/`). Les modules développés, Alpha-formes en dimensions 2 et 3, sont dorénavant partie intégrante de la libraire et distribués avec la version 2.3. Ensuite, on présente une nouvelle approche pour la reconstruction 3D à partir de nuages de points, dont le principe est de *déployer* une surface orientable sur les données. Cette méthode se révèle être très efficace, et surtout capable de fournir des réponses dans des cas difficiles. Elle offre, en outre, d'excellentes performances et permet de traiter de gros jeux de données. Enfin, on décrit une nouvelle méthode de reconstruction 3D pour des points organisés en sections. Il s'agit d'une méthode d'interpolation reposant sur les *voisins naturels*, un système de coordonnées barycentriques locales. Elle réunit deux grandes tendances : elle propose une définition fonctionnelle, $\mathscr{C}^1$ presque partout, de l'objet reconstruit tout en ne considérant que des structures géométriques discrètes de type triangulation de Delaunay. L'interpolation de coupes parallèles permet, de surcroît, une solution efficace, grâce à des calculs uniquement réalisés en dimension 2.

Mots-clés: reconstruction 3D, modélisation géométrique, nuages de points, sections, ensemble de contours, simplification d'échantillons, imagerie médicale, *reverse-engineering*, CAO.

Abstract

Computing a correct representation from a set of sample points is essential in various contexts, such as in applications including, for instance, medical imaging, reverse engineering, virtual reality, or special effects for cinema. The problem, addressed in this thesis, can be described as follows : given $\mathcal{S}$, a set of points sampled on a three dimensional object $\mathcal{O}$, reconstruct a geometric model of the surface bounding $\mathcal{O}$. The contributions are threefold.

First, we describe the implementation of a classic solution in Computational Geometry as a part of the CGAL library (`http://www.cgal.org/`). The packages, Alpha-shapes in 2 and 3D, are now available in the version 2.3 of the CGAL basic library. Second, we present a new approach to reconstruct orientable surfaces from unorganized point sets. The main idea is to extend incrementally a surface by attaching triangles at its current boundary. Our method is very effective and is able to reconstruct a correct surface for very large and challenging data sets. Moreover, the implementation is fast and memory efficient. Third, we describe a new interpolation method for cross-sections. This approach uses the *natural neighbors* barycentric coordinates. The reconstructed surface is smooth almost everywhere and is defined only with discrete geometric structures like Delaunay triangulation. Futhermore, the reconstruction is very efficient since all the calculations are performed in 2D.

Keywords: 3D reconstruction, geometric modeling, point clouds, cross-sections, sample simplification, medical imaging, reverse-engineering, CAD.