Journées du Groupe de Travail en Modélisation Géométrique

Equipe Informatique Géométrique et Graphique
Strasbourg 2012

Table des Matières

Rubans géodésiques pour la segmentation et la reconstruction de surfaces développables

Mathieu Huard[1], Nathalie Sprynski[1], Nicolas Szafran[2] et Luc Biard[2]

[1]CEA-LETI, MINATEC Campus, Grenoble, France
[2]Laboratoire Jean Kuntzmann, Université Joseph Fourier, Grenoble, France

Résumé
Nous considérons le problème général de l'acquisition et de la reconstruction de surfaces physiques 3D développables de classe de continuité G1 à l'aide de rubans équipés de microaccéléromètres et micromagnétomètres suivant des courbes géodésiques sur la surface. La méthode consiste à travailler dans l'espace des plans tangents fournis par les courbes géodésiques, permettant de caractériser les composantes canoniques d'une surface développable, puis de la reconstruire comme faisceau de droites s'appuyant sur les géodésiques acquises.

This paper deals with the acquisition and reconstruction of physical developable G1 surfaces by using a ribbon equipped with micro-accelerometers and micro-magnetometers, providing geodesic curves running on the surface. The method consists in working with the space of tangent planes provided by the geodesics, where the different components of the surface can be identified, and then reconstructed as beams of lines going through the acquired geodesics.

Mots-clés : Modélisation géométrique, Surface développable, Courbe géodésique, Capteurs d'orientation

1. Introduction

Nous considérons le problème général de l'acquisition et de la reconstruction de surfaces tridimensionnelles équipées de rubans de micro-capteurs. Les rubans *Morphosense*, développés par le département Leti du CEA de Grenoble, sont équipés de noeuds de capteurs constitués de magnétomètres et d'accéléromètres suivant les 3 axes d'un repère orthonormé (figure 1). Les capteurs permettent d'obtenir en chaque noeud l'orientation du repère de Serret-Frenet à la courbe suivie par le ruban. Lors de l'application de ce ruban sur une surface physique, la nature semi-rigide de ces rubans les contraint à suivre des courbes géodésiques de la surface [SSLB08]. Des méthodes d'interpolation spécifiques [Hua11, Spr07] permettent alors de reconstruire les courbes géodésiques sous-jacentes aux rubans.

Dans le cadre d'une ERC (Equipe de Recherche Commune) CEA-Leti/UJF-LJK, une table de contrôle métrologique est en cours de développement. Cette table, constituée d'une feuille de parablond (caoutchouc déformable) permet de générer une grande variété de formes. En pratique, ces formes sont de nature développable ou quasi-développable en raison de la faible elasticité du parablond. Cette table, équipée par ailleurs d'un système de contrôle externe (scanner), permet donc de tester et de valider des méthodes de reconstruction (en statique et en dynamique) basées sur la capture "Morphosense".

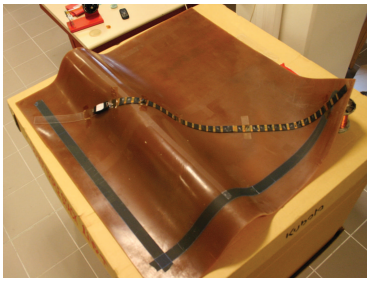


Figure 1: *Parablond équipé d'un ruban Morphosense*

L'identification de la forme du parablond avec une surface développable pose la question de l'ordre de continuité

de notre modèle. La nature restrictive des surfaces développables de classe $C2$ ainsi que la nature lisse des surfaces "parablond" générées nous conduisent à considérer des surfaces de classe $G1$. On considère donc ici le problème de l'identification et de la reconstruction d'une surface développable $G1$ $\mathbf{s}$ à partir d'une ou plusieurs courbes géodésiques tracées sur $\mathbf{s}$.

Après une certain nombre de rappels sur les surfaces développables dans la deuxième partie, on expose ensuite la problématique précise de notre problème puis notre approche d'identification et de reconstruction dans les parties suivantes. On présente alors des résultats de la méthode appliquée à des surfaces développables théoriques. Dans la dernière partie, on discute enfin des perspectives d'application au cadre plus pratique des surfaces quasi-développables.

2. Surfaces Développables

Les points de $\mathbb{R}^3$ et $\mathbb{R}^4$ sont respectivement définis par leurs coordonnées cartésiennes $\mathbf{u} = (x,y,z)$ et $\mathbf{U} = (u_0,u_1,u_2,u_3)$. L'ensemble des droites vectorielles de $\mathbb{R}^4$ définit l'espace projectif $\mathcal{P}^3(\mathbb{R})$. Précisément, le point $\mathbf{U}$ de $\mathbb{R}^4$ représente le point

$$\mathbf{u} = (\frac{u_1}{u_0}, \frac{u_2}{u_0}, \frac{u_3}{u_0}),$$

si $u_0 \neq 0$, ou bien le point à l'infini dans la direction (u_1,u_2,u_3) si $u_0 = 0$, définissant ainsi les coordonnées homogènes (u_0,u_1,u_2,u_3) d'un point de $\mathbb{R}^3$.

Approche cartésienne. – Une surface $\mathbf{s}$ de $\mathbb{R}^3$ est dite *réglée* si elle admet une paramétrisation de la forme

$$\mathbf{s}(u,v) = \mathbf{c}(u) + v\,\mathbf{e}(u), \tag{1}$$

où $\mathbf{c}(u)$ est une courbes de $\mathbb{R}^3$ appelée *directrice* de la surface et $\mathbf{e}(u)$ est un champ de vecteurs. La surface est ainsi engendrée par la famille des droites $L(u) = \{\mathbf{c}(u), \mathbf{e}(u)\}$, appelées *génératrices* de la surface.

Une telle paramétrisation est dite *normalisée* [DoC76] si

$$|\mathbf{e}(u)| \equiv 1 \quad \text{et} \quad \mathbf{e}'(u)\cdot\mathbf{c}'(u) \equiv 0.$$

La directrice $\mathbf{c}(u)$ est alors la *courbe de régression* et concentre l'ensemble des (éventuels) points singuliers. Notons qu'une autre normalisation consiste à choisir une directrice orthogonale à l'ensemble des génératrices [Hos71].

Une surface *développable* est une surface que l'on peut "déplier" (*développer*) sur le plan par une transformation isométrique. Une surface réglée est développable si le plan tangent est constant le long de chaque droite génératrice, ce qui se traduit par la condition

$$\det(\mathbf{c}'(u), \mathbf{e}(u), \mathbf{e}'(u)) \equiv 0, \tag{2}$$

exprimant la coplanarité de ces trois vecteurs. Ainsi, la direction du vecteur normal

$$\mathbf{s}'_u \wedge \mathbf{s}'_v = \mathbf{c}'(u) \wedge \mathbf{e}(u) + v\,\mathbf{e}'(u) \wedge \mathbf{e}(u) \tag{3}$$

est indépendante de v.

Pour une paramétrisation normalisée, on déduit $\mathbf{e}'(u)\cdot\mathbf{e}(u) \equiv 0$, ce qui permet de classer les surfaces réglées développables $C2$ dans l'une des trois catégories suivantes.

◇ Les *développées tangentielles*, définies comme l'enveloppe des tangentes à la courbe de régression (dont tous les points sont singuliers sur la surface développable) :

$$\mathbf{s}_0(u,v) = \mathbf{c}(u) + v\,\mathbf{c}'(u). \tag{4}$$

◇ Les *cônes généralisés*, pour lesquels la courbe de régression dégénère en un unique point, le sommet du cône.

◇ Les *cylindres généralisés*, pour lesquels la courbe de régression dégénère en un unique point à l'infini. La direction $\mathbf{e}$ du cylindre est alors la direction commune à toutes ses génératrices : $\mathbf{e}(u) = \mathbf{e}$.

Pour une paramétrisation quelconque (1) d'une surface réglée développable, la courbe de régression $\mathbf{b}(u)$ est définie par

$$\mathbf{b}(u) = \mathbf{c}(u) + v(u)\,\mathbf{e}(u) \text{ avec } v(u) = -\frac{(\mathbf{c}'\times\mathbf{e})\cdot(\mathbf{e}'\times\mathbf{e})}{(\mathbf{e}'\times\mathbf{e})^2}(u). \tag{5}$$

Notons enfin que la courbure gaussienne des surfaces réglées développables de classe $C2$ est nulle en tout point. Nous considérons ici des surfaces développables $G1$, composées de développées tangentielles, de cônes généralisés et de cylindres généralisés, ayant un contact $G1$ le long de droites génératrices communes.

Approche duale. – Toute surface développable peut également être définie comme l'enveloppe d'une famille à un paramètre de plans :

$$U(t):\ n_1(t)\,u_1 + n_2(t)\,u_2 + n_3(t)\,u_3 + d(t) = 0, \tag{6}$$

avec la convention de Hesse $n_1^2 + n_2^2 + n_3^2 = 1$, où $\mathbf{n} = (n_1,n_2,n_3)$ est le vecteur normal unitaire au plan et d sa distance à l'origine. La surface est ainsi associée à la courbe $\mathbf{U}(t) = (d(t), \mathbf{n}(t))$ contenue dans le sous-espace de $\mathbb{R}^4$ défini par l'équation

$$u_1^2 + u_2^2 + u_3^2 = 1, \tag{7}$$

appelé *cylindre de Blaschke*, chaque vecteur $\mathbf{U}(t)$ représentant le plan tangent $U(t)$ à la surface. Les droites génératrices $L(t)$ de la surface développable sont alors définies par l'intersection

$$L(t) = U(t) \cap \dot{U}(t). \tag{8}$$

La courbe de régression $\mathbf{B}(t) = (b_0(t), b_1(t), b_2(t), b_3(t))$ est définie en coordonnées homogènes par la double intersection :

$$\mathbf{B}(t) = \mathbf{U}(t) \wedge \mathbf{U}'(t) \wedge \mathbf{U}''(t). \tag{9}$$

3. Objectifs et enjeux

On considère une surface développable **s** de régularité $G1$ sur laquelle on se donne une (ou plusieurs) courbes géodésiques **r**. Cela correspond à une modélisation du problème de reconstruction de la table métrologique (**s**) par les rubans Morphosense (courbe **r**).

On veut alors reconstruire la surface **s** en exploitant la courbe géodésique **r**. L'approche présentée ici s'appuie sur le résultat suivant.

Une courbe géodésique sur une surface développable identifie complètement la portion de la surface dont les génératrices intersectent cette géodésique. Précisément, la surface développable est l'enveloppe des plans rectifiants de la courbe géodésique [Str61].

En effet, le repère de Serret-Frenet d'une géodésique coïncide en tout point avec le repère de Darboux de la surface développable. En particulier, la normale du repère de Serret-Frenet

$$\mathbf{n} = \mathbf{r}' \times (\mathbf{r}' \times \mathbf{r}''),$$

coïncide le long de la courbe avec la normale à la surface sous-jacente. Ainsi, connaître la géodésique sur une portion de la surface revient à connaître la famille des plans tangents associés à cette portion, et permet donc de la reconstruire précisément.

Cependant, deux questions essentielles sont à prendre en considération pour la mise en oeuvre de cette stratégie.

◇ **L'absence d'expression analytique** exacte pour les géodésiques considérées. En effet, en pratique les géodésiques sont reconstruites numériquement à partir des informations capteurs. L'approche duale, qui implique l'utilisation des dérivées secondes, ne peut donc être appliquée en l'état.

◇ **La segmentation**. Par notre choix d'un modèle de surfaces développables $G1$, la géodésique calculée traversera successivement des portions de cône, de cylindre et de développées tangentielles. L'un des enjeux est donc de localiser et d'identifier chacune des portions de la géodésique associées à ces différentes composantes canoniques.

Dans la suite, les courbes géodésiques sont donc assimilées à une suite de points $\mathbf{p}_i, i = 1..n$ de l'espace euclidien $\mathbb{R}^3$ associés à des normales $\mathbf{n}_i$ définissant le plan tangent à la surface en ces points.

4. Reconstruction

La reconstruction globale des surfaces développables $G1$ que nous étudions se fait au travers d'une première étape de segmentation de la surface, puis par la reconstruction individuelle de chacune des portions de la surface. Cela nécessite, dans le cas des développées tangentielles, une approximation continue des données.

4.1. Segmentation à l'aide du cylindre de Blaschke

L'idée de cette approche est d'étudier directement les données dans l'espace des normales pour caractériser la surface sous-jacente.

Pour un cylindre généralisé $\mathbf{s}(u,v)$ de direction $\mathbf{e}$, on déduit de l'expression (3) que $\langle \mathbf{s}'_u \wedge \mathbf{s}'_v, \mathbf{e} \rangle \equiv 0$, ce qui prouve que la famille des normales $\mathbf{n}_i$ décrivent un grand cercle de S^2, à savoir l'intersection de S^2 avec le plan vectoriel de normale $\mathbf{e}$. Cette propriété permet d'identifier un cylindre généralisé uniquement à partir des normales.

L'opération est plus compliquée pour les deux autres classes de surfaces développables. En particulier, le lemme suivant montre que l'ensemble des normales ne suffit pas à identifier les autres catégories.

Lemme 4.1 Soit $\mathbf{c}(u)$ une courbe régulière tracée sur la sphère unité S^2 ($\mathbf{c}(u)$ non tangente à un grand cercle). Alors il existe un cône généralisé et une développée tangentielle dont la famille des normales $\mathbf{n}(u)$ coïncide avec $\mathbf{c}(u)$.

Preuve Considérons le cône $\mathbf{s}(u,v) = v\,\varphi(u)$ avec $\varphi(u) = \mathbf{c}(u) \times \mathbf{c}'(u)$. La courbe $\mathbf{c}(u)$ étant régulière, $\varphi(u)$ ne s'annule pas et le cône est donc bien défini. On déduit ensuite de la relation (3) :

$$\begin{aligned} \mathbf{s}'_u \wedge \mathbf{s}'_v &= v\,\varphi'(u) \wedge \varphi(u)\,, \\ &= (\mathbf{c} \wedge \mathbf{c}'') \wedge (\mathbf{c} \wedge \mathbf{c}')\,, \\ &= -\det(\mathbf{c}, \mathbf{c}', \mathbf{c}'')\,\mathbf{c}\,. \end{aligned}$$

Ainsi, quitte à réorienter la surface, la famille des normales unitaires au cône $\mathbf{s}(u,v)$ coïncide avec $\mathbf{c}(u)$.

La preuve pour les développées tangentielles est donnée dans [DoC76] (section 3-5). ∎

L'idée est alors de travailler là encore avec les points $\mathbf{U}_i = (d_i, \mathbf{n_i})$ du cylindre de Blaschke, où $d_i = -\mathbf{p}_i \cdot \mathbf{n}_i$. On a vu que les surfaces développables étaient duales de courbes $\mathbf{U}(t)$ sur le cylindre de Blaschke (cf 4.2). On peut caractériser plus précisément ces courbes selon les différentes classes de surfaces développables :

(1) Dans le cas des *cylindres généralisés*, tous les plans tangents $\mathbf{U}(t) = (d(t), \mathbf{n}(t))$ sont parallèles à un vecteur direction $\mathbf{d}$, et vérifient donc la relation $\mathbf{n} \cdot \mathbf{d} = 0$. Ainsi la courbe $\mathbf{U}$ est contenue dans l'hyperplan de $\mathbb{R}^4$:

$$d_1\,u_1 + d_2\,u_2 + d_3\,u_3 = 0,$$

(2) Dans le cas des *cônes généralisés*, tous les plans tangents $\mathbf{U}(t)$ passent par un même point $\mathbf{B} = (1, \mathbf{b})$. Cela implique que la courbe $\mathbf{U}$ est contenue dans le sous-espace affine de dimension 3 décrit par la relation d'incidence :

$$u_0 + b_1\,u_1 + b_2\,u_2 + b_3\,u_3 = 0,$$

(3) Dans le cas des *développées tangentielles*, la courbe ne pourra pas être restreinte à ce type d'espaces.

Cette classification permet de déterminer à quel type de

surface correspondent les points $\mathbf{U}_i$, sans avoir recours à un paramétrage de $\mathbf{U}$. On se base pour cela sur la méthode développée dans [Pet04]. Etant donné un ensemble de points $\mathbf{U}_i$, cette méthode nous permet de déterminer s'il existe un espace de type (1) ou (2) les contenant tous, à l'aide d'une analyse en composantes principales. Dans le cas des surfaces $G1$ que nous étudions, on cherche à partitionner l'ensemble des points $\mathbf{U}_i$ pour retrouver les différentes portions de la surface considérée. Il s'agit donc d'appliquer la méthode à des parties de l'ensemble des points $\mathbf{U}_i$ afin de les partitionner selon la catégorie de surface à laquelle ils appartiennent.

Les points $\mathbf{U}_i$ étant extraits d'une courbe géodésique, ils sont ordonnés. Autrement dit si la surface couverte par la géodésique est composée de m portions de classe $C2$, il existe $m+1$ indices

$$i_0 = 0 \leq i_1 \leq ... \leq i_{m-1} \leq i_m = n$$

tels que les sous-ensembles de points

$$\{\mathbf{U}_{i_k+1}, \mathbf{U}_{i_k+1}, ..., \mathbf{U}_{i_{k+1}}\}, k = 0...m$$

sont entièrement contenus dans l'une de ces portions.

Pour déterminer si l'on est sur un cône ou un cylindre, quatre points suffisent puisqu'ils déterminent entièrement les espaces de dimension 3 décrits en (1) et en (2). On procède donc à l'aide d'une fenêtre glissante de quatre points sur lesquels on opère l'analyse en composantes principales.

Tant que les points testés appartiennent à des espaces de type (1) ou (2), la portion de surface décrite correspond à un cône ou à un cylindre dont on peut déterminer respectivement le sommet ou la direction. On peut alors reconstruire ces portions comme faisceau de droites sécantes au sommet du cône ou parallèles à la direction du cylindre qui passent par les points $\mathbf{p}_i$ correspondants.

Les points qui ne satisfont pas les cas (1) et (2) sont issus d'une portion de développée tangentielle. Ce cas est plus complexe et nécessite un traitement différent décrit dans la section suivante.

4.2. Approche continue pour les développées tangentielles

Pour reconstruire les portions de surface correspondant à des développées tangentielles, on a besoin d'approcher la courbe de régression $\mathbf{b}(t)$ correspondante. L'idée est d'interpoler les points $\mathbf{U}_i$ que l'on a identifié dans cette portion selon un paramètre t, afin de construire une approximation de l'enveloppe $\mathbf{U}(t)$ des plans tangents à la surface. On peut alors appliquer directement le cadre théorique de la représentation duale. En effet, l'équation (9) permet alors d'obtenir la courbe de régression $\mathbf{B}(t)$ sous sa forme homogène ([PW99]). On peut alors reconstruire $\mathbf{s}$ sur $[a,b]$ comme l'enveloppe des tangentes de $\mathbf{b}$.

Remarque On vérifie que la géodésique $\mathbf{r}$ (l'ensemble des points $\mathbf{p}_i$) est contenue dans la surface reconstruite, mais contrairement aux cas du cône et du cylindre, on n'a plus besoin de $\mathbf{r}$ au moment de la reconstruction.

5. Résultats

On présente ici un exemple qui détaille les étapes du processus de reconstruction. On génère les surfaces développables $G1$ comme surfaces de Bézier. Il est aisé de générer des cônes et des cylindres généralisés par cette méthode. Les développées tangentielles peuvent également s'écrire sous forme de Bézier (produit tensoriel), que l'on trouve par un calcul élémentaire en utilisant l'équation (4).

En effet, si l'on écrit $\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{x}_i B_i^n(u)$ la courbe de régression, et en posant

$$\begin{cases} \mathbf{d}_0 = \mathbf{x}_1 - \mathbf{x}_0,\ \mathbf{d}_n = \mathbf{x}_n - \mathbf{x}_{n-1} \\ \mathbf{d}_i = i\,(\mathbf{x}_i - \mathbf{x}_{i-1}) + (n-i)(\mathbf{x}_{i+1} - \mathbf{x}_i),\ i = 1..n-1, \end{cases}$$

on obtient la forme suivante

$$\mathbf{s}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{1} \mathbf{q}_{i,j} B_i^n(u) B_j^1(v), \tag{10}$$

avec

$$\begin{cases} \mathbf{q}_{i,0} = \mathbf{x}_i \\ \mathbf{q}_{i,1} = \mathbf{x}_i + \mathbf{d}_i \end{cases},\ i = 0...n.$$

On peut ainsi générer les surfaces développables $G1$ en raccordant des patches Bézier de chacune des trois catégories. Cette méthode permet d'obtenir une grande variété de surfaces. De plus, ce type de surfaces de test permettra par la suite d'étendre ces travaux. En effet, par élévation de degré, on pourra imposer aux surfaces générées des petites perturbations pour les étendre au cas quasi-développable.

L'exemple présenté dans les figures 2-3-4 détaille les étapes de la méthode d'acquisition, de segmentation et de reconstruction. A partir d'une surface développable quelconque (a) sur laquelle on pose un ruban pour acquérir une courbe géodésique (b), on segmente la surface en effectuant l'analyse en composantes principales. On identifie alors dans un premier temps les portions de cône et de cylindre (c) pour les reconstruire individuellement (d). Les portions restantes, correspondant à des développées tangentielles, sont alors reconstruites par l'intermédiaire de leur courbe de régression, identifiée par dualité dans l'espace des plans tangents (e). La surface dont les droites génératrices sont couvertes par la géodésique est alors entièrement identifiée et reconstruite (f).

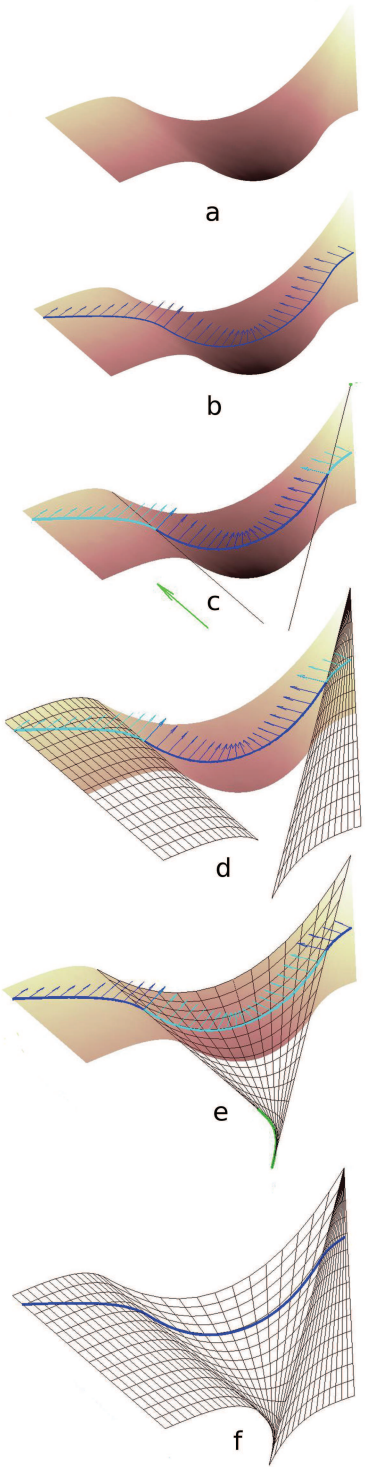


Figure 2: *Etapes du processus de reconstruction (vue 1)*

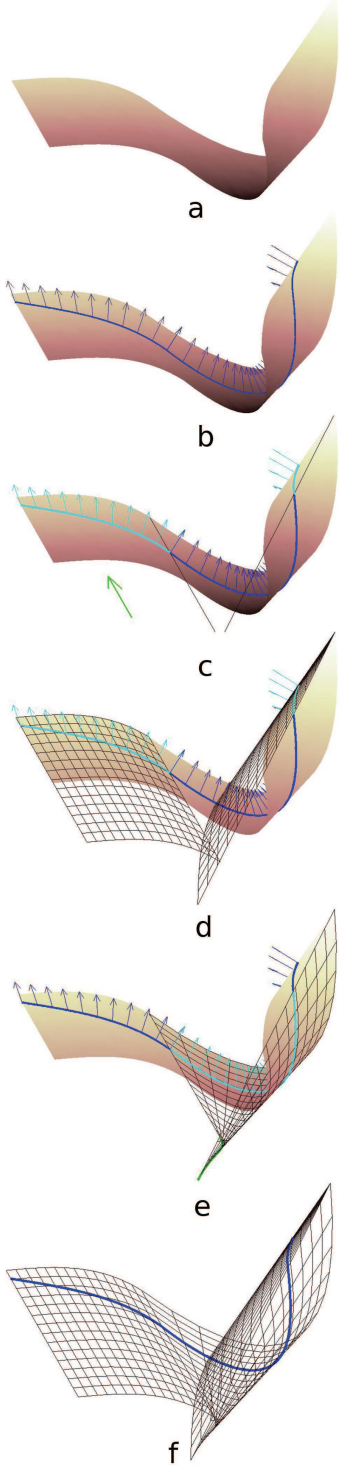


Figure 3: *Etapes du processus de reconstruction (vue 2)*

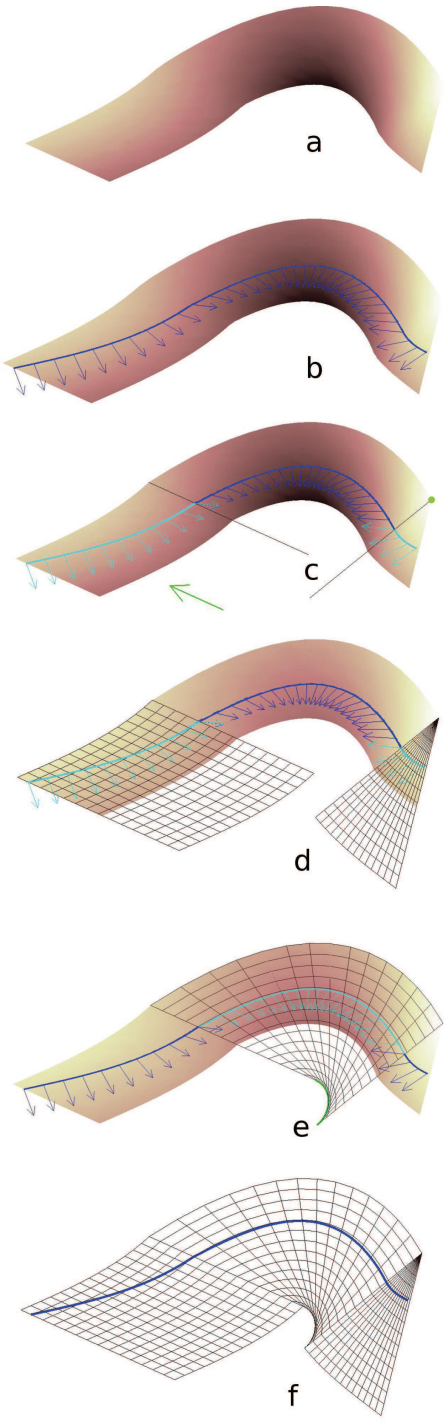


Figure 4: *Etapes du processus de reconstruction (vue 3)*

6. Conclusion

On a présenté dans ce travail une méthode de reconstruction pour les surfaces développables de regularité $G1$. La méthode permet de reconstruire exactement les portions de surface issues de cônes et de cylindres généralisés, et avec une bonne précision les portions correspondant à des développées tangentielles.

Les surfaces développables $G1$ étudiées sont moins restrictives et plus souples pour une modélisation réaliste des surfaces à reconstruire. Cependant, pour être appliquée en pratique, la méthode devra être étendue au cas des surfaces quasi-développables. Notre méthode de génération de surfaces permet d'introduire des perturbations aux surfaces de test, afin de les étendre à ce cadre et de tester la robustesse de la méthode. Cette démarche est actuellement en cours d'étude.

Références

[DoC76] DOCARMO M. P. : *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[DS06] DAVID D., SPRYNSKI N. : Method and device for acquisition of a geometric shape, 2006.

[Far02] FARIN G. : *Curves and Surfaces for CAGD (5th Edition)*. Academic Press, 2002.

[FSB10] FAROUKI R., SZAFRAN N., BIARD L. : Construction and smoothing of triangular coons patches with geodesic boundary curves. *Computer Aided Geometric Design. Vol. 27* (2010).

[Hos71] HOSCHEK J. : *LinienGeometrie*. Zürich : Bibliographisches Institut, 1971.

[HSSB12] HUARD M., SPRYNSKI N., SZAFRAN N., BIARD L. : Reconstruction of quasi developable surfaces from ribbon curves. Soumis à Numerical Algorithms, 2012.

[Hua11] HUARD M. : Serret-frenet frame interpolation under length constraints for surface reconstruction. In *Actes des journées du GTMG* (2011).

[Pet04] PETERNELL M. : Developable surface fitting to point clouds. *Computer Aided Geometric Design. Vol. 21* (2004).

[PW99] POTTMANN H., WALLNER J. : Approximation algorithms for developable surfaces. *Computer Aided Geometric Design. Vol. 16* (1999).

[PW01] POTTMANN H., WALLNER J. : *Computational Line Geometry*. Springer Verlag, 2001.

[Spr07] SPRYNSKI N. : *Reconstruction de Courbes et Surfaces à partir de données tangentielles*. PhD thesis, Université Joseph Fourier, 2007.

[SSLB08] SPRYNSKI N., SZAFRAN N., LACOLLE B., BIARD L. : Surface reconstruction via geodesic interpolation. *Computer-Aided Design. Vol. 40* (2008).

[Str61] STRUIK D. J. : *Lectures on Classical Differential Geometry*. Dover Publications, Inc, 1961.

Reconstruction de Modèles CAO de scènes industrielles étant données des connaissances *a priori*

Aurélien Bey[1,2,3], Raphaëlle Chaine[1,2], Raphaël Marc[3], Guillaume Thibault[3,4]

[1]Université de Lyon, CNRS
[2]Université Lyon 1, LIRIS, UMR5205, 69622
[3]Électricité De France Recherche & Développement, 92140
[4]CNRS, Laboratoire de Physiologie de la Perception et de l'Action, UMR7152, 75231

Résumé

Nous abordons dans cet article le problème de la reconstruction de modèles CAO à partir de nuages de points acquis en environnement industriel, en nous appuyant sur des modèles 3D préexistants ainsi que sur des connaissances métier quant à la composition des environnements traités. Ces diverses connaissances a priori peuvent être utilisées pour guider la reconstruction afin d'obtenir des modèles CAO fiables correspondant aux nuages de points. Nous concentrons plus particulièrement notre travail sur le traitement des cylindres, plans et tores. Nous proposons de formuler le problème de la reconstruction comme la recherche de la configuration la plus probable vis-à-vis de multiples contraintes. Le problème d'optimisation ainsi défini est résolu à l'aide d'une méthode d'exploration stochastique de l'espace des solutions, basée sur l'ajout d'éléments dans la configuration en cours de construction et la gestion gloutonne des conflits pouvant survenir, de manière à améliorer efficacement la configuration à chaque étape. Nous montrons que la méthode proposée permet la reconstruction de modèles fiables en présentant quelques résultats obtenus sur une scène industrielle.

Mots-clés : Nuage de points, Reconstruction, Reconnaissance de formes, CAO, Bayésien

1. Introduction

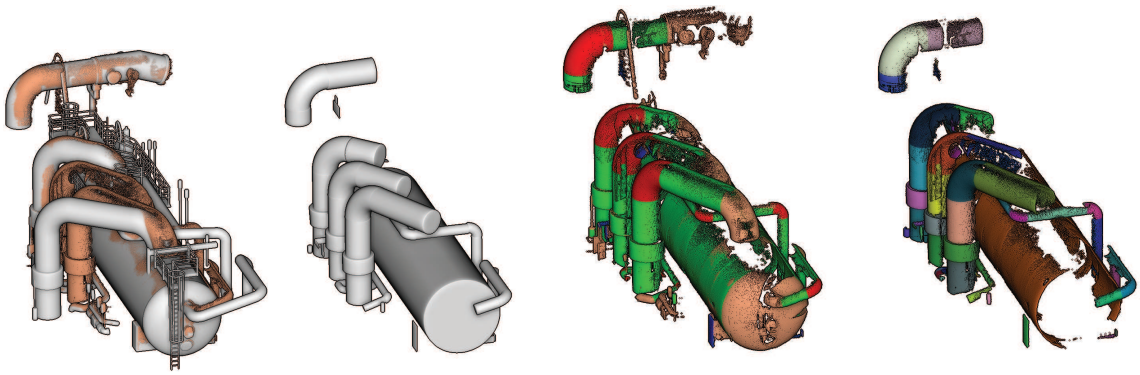
1.1. Contexte

Les modèles 3D d'installations industrielles telles que les centrales électriques, raffineries ou usines pharmaceutiques servent à différentes fins : la simulation d'opérations de maintenance, la formation, la radioprotection etc. Certaines de ces applications nécessitent de disposer de modèles 3D décrivant précisément l'état réel des scènes qu'ils représentent (précision de l'ordre du centimètre). De tels modèles sont nommés modèles "Tel Que Construit" (TQC), et sont habituellement produits à partir de nuages de points issus de relevés laser sur site.

Nous nous intéressons dans nos travaux aux modèles CAO décrits comme des assemblages de primitives géométriques simples telles que les plans, cônes, cylindres et tores. Chacune de ces primitives est un objet géométrique entièrement caractérisé par une position et une orientation, ainsi qu'un certain nombre de paramètres géométriques inhérents au type de la primitive.

En plus du nuage de points $\mathscr{P}$, nous considérons que nous disposons d'un modèle CAO *a priori* $\mathcal{M}_0$. Alors que le nuage de points définit les observations, et donc l'état réel de la scène à traiter, le modèle *a priori* représente l'état théorique de l'installation : il s'agit d'une estimation grossière de la solution que l'on cherche à reconstruire (cf. Figure 1a). Il peut exister des différences significatives entre $\mathcal{M}_0$ et l'état réel de la scène représentée par $\mathscr{P}$: certains éléments peuvent être déplacés, réorientés, voire même déformés. De plus, certains composants peuvent être absents ou dupliqués d'une scène à l'autre.
Concernant la provenance du modèle *a priori*, il peut par exemple s'agir d'une représentation générique du site à traiter (modèle 3D décrivant une centrale électrique type), ou de la reconstruction TQC d'un autre site semblable.

Dans cet article, nous nous intéressons plus particulièrement au traitement des cylindres et des plans, puisqu'ils



(a) Nuage de points $\mathscr{P}$ et modèle *a priori* $\mathcal{M}_0$

(b) Modèle reconstruit $\mathcal{X}$

(c) Type de primitive reconnu (bleu :plan, vert :cylindre, rouge :tore, saumon : non reconnu)

(d) Segmentation

Figure 1: Résultats de la reconstruction sur un jeu de données industriel. Le modèle *a priori* (en gris dans la Figure (a)) contient 626 primitives, dont 391 cylindres, et présente quelques différences évidentes avec le nuage de points (en saumon : 990323 points). La plupart des cylindres de l'*a priori* n'ont aucune correspondance dans le nuage de points. La configuration calculée (Figure (b)) contient 68 cylindres, et la segmentation associée est présentée en Figure (d) : chaque segment y est coloré aléatoirement. La Figure (c) présente les segments en fonction du type de la primitive reconnue.

représentent une part importante des composants industriels. Les cylindres correspondent par exemple aux lignes de tuyauterie, aux cuves, certaines machineries, etc. Quant aux plans, ils représentent principalement les sols et murs. De plus, les cylindres correctement détectés pourront par la suite être utilisés comme contraintes pour permettre la reconstruction fiable des tores, qui quant à eux servent essentiellement à décrire les composants de transition (coudes) dans les tuyauteries, et peuvent donc être calculés dans un second temps.

Ainsi, étant donnés $\mathscr{P}$ et $\mathcal{M}_0$, nous cherchons à construire un modèle CAO $\mathcal{X}$ à base de cylindres, plans et tores tel que :

- $\mathcal{X}$ s'ajuste bien sur le nuage de points
- chaque primitive de $\mathcal{X}$ correspond bien à une primitive de $\mathcal{M}_0$. Nous considérons en effet que, pour fournir des résultats fiables, les éléments détectés doivent être validés par mise en correspondance avec les éléments de $\mathcal{M}_0$
- $\mathcal{X}$ est consistant : les connexions entre éléments doivent être pertinentes vis-à-vis des règles métier d'assemblage des composants. Ces contraintes d'assemblage font donc partie des connaissances *a priori* dont on dispose au sujet de la scène traitée, au même titre que le modèle $\mathcal{M}_0$.

Nos principales contributions sont la formulation probabiliste du problème de reconstruction étant données des connaissances *a priori*, ainsi que la mise en place d'une nouvelle méthode permettant de résoudre le problème d'optimisation ainsi posé. Notre démarche repose sur l'utilisation du modèle disponible pour guider la reconstruction, et permet d'assurer la consistance du modèle CAO reconstruit. Non seulement le résultat s'ajuste bien sur le nuage de points, mais il est aussi pertinent vis-à-vis des contraintes d'assemblage que l'on se donne et sa ressemblance au modèle *a priori* peut en outre être assurée.

1.2. État de l'art

À notre connaissance, les travaux de [Bos10] sont les seuls proposant d'utiliser un modèle CAO *a priori* en tant qu'estimation initiale du nuage de points que l'on souhaite reconstruire. L'auteur propose de recaler individuellement les éléments du modèle en utilisant l'algorithme ICP. Toutefois, cet algorithme est connu pour ne fournir de résultats satisfaisants que lorsque les données à recaler sont initialement très proches l'une de l'autre. Cette approche ne peut donc pas traiter de changements significatifs.

Les auteurs de [SP10] utilisent aussi des informations *a priori*, telles que les tailles de fenêtres, pour segmenter les façades de bâtiments, mais ces travaux sont très spécifiques au problème du traitement de façades, bien que certaines idées intéressantes puissent être utilisées dans un contexte différent. D'autre part, il existe de nombreux travaux traitant de la reconstruction de nuages de points à partir d'*a priori* [PMG*05, JWB*06, GSH*07], mais ceux-ci abordent la reconstruction de maillages surfaciques alors que nous nous intéressons à la reconstruction de modèles par

assemblage de primitives géométriques, permettant d'obtenir une représentation de la scène plus adaptée à nos besoins.

En matière de reconnaissance de formes, deux approches sont très utilisées : les Transformées de Hough Généralisées (THG) et l'algorithme RANdom SAmple Consensus (RANSAC).
Les THG sont des méthodes à base de vote : chaque point du nuage s'exprime en faveur d'un ensemble de formes dans un espace paramétrique discrétisé. Les formes dont la description paramétrique correspond à un pic de votes sont alors validées et considérées comme reconnues dans le nuage de points. Le principal inconvénient de cette méthode, lorsqu'elle est appliquée telle quelle, est sa complexité prohibitive, puisqu'elle requiert le parcours de l'espace paramétrique (5 dimensions pour le cylindre, par exemple) pour chaque vote. Cette complexité peut être significativement amoindrie dans le cas du cylindre, en considérant successivement différents paramètres, permettant ainsi de séparer le problème en THG de dimensions inférieures [RV05]. Cette méthode semble fournir des résultats intéressants, mais la détection de cylindres de tailles variées dans de grandes scènes reste problématique.
L'algorithme RANSAC [FB87] quant à lui propose une méthode stochastique, dans laquelle des ensembles minimaux de points sont tirés aléatoirement, et des formes candidates sont générées à partir de ces ensembles minimaux. Si le tirage des points au sein du nuage est effectué de manière purement uniforme, la grande majorité des formes candidates construites ne sera pas pertinente, et le nombre de tentatives nécessaires à la reconnaissance d'une forme peut être énorme. Dans [SWK07], les auteurs proposent une méthode astucieuse permettant de diminuer significativement le nombre de tentatives permettant d'assurer le succès de la reconnaissance. De plus, les auteurs proposent aussi une méthode efficace d'évaluation des formes candidates. Cet algorithme permet d'obtenir des résultats corrects sur les jeux de données que nous avons testés, bien qu'il reste quelques problèmes conduisant à l'apparition de formes non pertinentes dans les résultats.

1.3. Vue d'ensemble de la démarche

La disponibilité d'un *a priori* nous conduit naturellement à formuler le problème sous forme Bayésienne [DTB06], où la reconstruction s'exprime comme la recherche du modèle CAO le plus probable (cf. Section 2.2). Pour ce faire, nous définissons la probabilité en question de manière à ce qu'elle concentre nos attentes en matière de qualité de la reconstruction. La probabilité que nous proposons embarque donc les différentes contraintes (cf. Section 2.3) que sont la qualité d'ajustement sur le nuage, la ressemblance au modèle *a priori* et la cohérence des connexions entre éléments composant le modèle reconstruit. Une fois la probabilité définie, nous devons adopter une méthode permettant de trouver efficacement la configuration optimale parmi l'ensemble immense des solutions possibles. Nous proposons pour cela un nouvel algorithme d'optimisation itératif basé sur l'exploration stochastique de l'espace des solutions (cf. Section 3.1), ne traitant que les cylindres et plans. À chaque étape, une primitive candidate est générée aléatoirement puis on tente de l'insérer dans la configuration en cours de construction. Comme cette insertion est susceptible de diminuer la probabilité de la configuration courante, nous proposons un algorithme de gestion de conflits avec les primitives déjà détectées : le candidat n'est accepté que s'il est en mesure d'augmenter la probabilité de la configuration après qu'il y ait été inséré et que les conflits aient été supprimés. Nous présentons de plus un algorithme de génération de primitives favorisant l'apparition de formes pertinentes au regard du nuage de points et du modèle *a priori* (cf. Section 3.2). Le modèle ainsi construit à base de cylindres et plans uniquement est finalement complété en joignant les cylindres par des tores lorsque cette opération améliore la description de la scène (cf. Section 4).

2. Une approche probabiliste

2.1. Introduction aux données et notations

On se donne $\mathscr{P} = \{(\mathbf{p}_0, \mathbf{n}_0), \ldots, (\mathbf{p}_{n-1}, \mathbf{n}_{n-1})\}$ un nuage de points représentant l'état réel de l'installation à traiter. Chaque paire $\left(\mathbf{p}_j, \mathbf{n}_j\right) \in \mathscr{P}$ représente un point $\mathbf{p}_j$ ainsi que le vecteur normal $\mathbf{n}_j$ qui lui est associé, pouvant être fourni avec les données d'acquisition ou calculé à l'aide d'ajustement de plans locaux [MN04]. Le modèle CAO *a priori* $\mathcal{M}_0 = \left\{\mathcal{S}^0, \ldots, \mathcal{S}^{m-1}\right\}$ est un ensemble de primitives parmi lesquelles nous ne considérons que les cylindres droits et les plans. La figure 2 illustre la manière dont les primitives géométriques que nous utilisons sont exprimées.

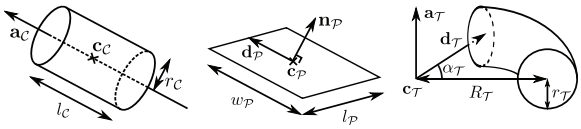


Figure 2: Définition des primitives géométriques paramétrées.

Nous cherchons à trouver la configuration $\mathcal{X} = \{\mathcal{S}_0, \ldots, \mathcal{S}_{x-1}\}$ composée de de cylindres, plans et tores qui soit la plus probable.

2.2. La reconstruction vue comme problème d'optimisation

Si nous considérons la reconstruction d'un point de vue probabiliste, nous cherchons la configuration maximisant une probabilité *a posteriori* $\pi(.)$ qui, selon la règle de Bayes, est proportionnelle (à une constante de normalisation près)

au produit d'un terme d'attache aux données P_D et d'un terme *a priori* P_P :

$$\pi(\mathcal{X}|\mathscr{P},\mathcal{M}_0) \propto P_D(\mathscr{P}|\mathcal{X},\mathcal{M}_0) \times P_P(\mathcal{X}|\mathcal{M}_0) \quad (1)$$

"$\propto$" représente la relation de proportionnalité : la constante de normalisation peut être ignorée car elle n'intervient pas dans le processus d'optimisation.
$P_D(\mathscr{P}|\mathcal{X},\mathcal{M}_0)$ est en fait indépendant de $\mathcal{M}_0$ dans la mesure où $\mathscr{P}$ est lui même indépendant de $\mathcal{M}_0$. Cette mesure quantifie la qualité de l'ajustement de $\mathscr{P}$ sur $\mathcal{X}$. P_P quant à elle définit une densité *a priori* sur l'ensemble de toutes les configurations possibles. Nous devons donc faire en sorte que P_P favorise les configurations consistantes, et tienne aussi compte de la ressemblance à $\mathcal{M}_0$.
Si nous utilisons des densités de la forme :

$$P_D(\mathscr{P}|\mathcal{X}) \propto \mathrm{e}^{-\lambda_D.E_D(\mathcal{X},\mathscr{P})} \quad (2)$$

$$P_P(\mathcal{X}|\mathcal{M}_0) \propto \mathrm{e}^{-E_P(\mathcal{X},\mathcal{M}_0)} \quad (3)$$

nous devons donc trouver la configuration $\mathcal{X}$ minimisant :

$$\begin{aligned} E(\mathcal{X},\mathscr{P},\mathcal{M}_0) &= -\log(\pi(\mathcal{X}|\mathscr{P},\mathcal{M}_0)) \\ &= \lambda_D.E_D(\mathcal{X},\mathscr{P}) + E_P(\mathcal{X},\mathcal{M}_0) \end{aligned} \quad (4)$$

où λ_D spécifie l'importance relative du terme d'attache aux données E_D par rapport au terme *a priori* E_P.

D'après les explications précédentes et les équations 2 et 3, il apparait que l'énergie d'attache aux données E_D mesure la qualité d'ajustement de $\mathscr{P}$ sur $\mathcal{X}$, alors que l'énergie *a priori* E_P mesure la conformité de $\mathcal{X}$ vis-à-vis des attentes *a priori*.

2.3. Formulation des énergies

2.3.1. Terme d'attache aux données

Nous définissons l'énergie d'attache aux données E_D comme la somme des contributions apportées par chaque point du nuage :

$$E_D(\mathcal{X},\mathscr{P}) = \frac{1}{n} \sum_{\mathcal{S}_i \in \mathcal{X}} \sum_{\mathbf{p} \in \mathscr{P}} e_D(\mathcal{S}_i, \mathbf{p}) \quad (5)$$

rappelons que n est le nombre de points dans $\mathscr{P}$.
Soit $d(\mathcal{S}_i, \mathbf{p})$ (noté $d^i_{\mathbf{p}}$) la distance séparant $\mathbf{p}$ de la primitive $\mathcal{S}_i \in \mathcal{X}$ dont il est le plus proche, $\mathbf{n}$ le vecteur normal associé à $\mathbf{p}$ et $\mathbf{v}$ le vecteur normale à $\mathcal{S}$ en son point le plus proche de $\mathbf{p}$. Nous définissons e_D comme suit :

$$e_D(\mathcal{S}_i, \mathbf{p}) = \begin{cases} 0 \text{ si } d^i_{\mathbf{p}} > 3\varepsilon \\ \left(\dfrac{d^i_{\mathbf{p}}}{\varepsilon}\right)^2 + \left(\dfrac{\arccos(|\mathbf{n}\cdot\mathbf{v}|)}{\eta}\right)^2 - 1 \text{ sinon} \end{cases} \quad (6)$$

"$\cdot$" étant le produit scalaire dans $\mathbb{R}^3$. ε est un seuil estimant le bruit du nuage de points, tandis que η est un seuil angulaire spécifiant la tolérance à l'erreur d'ajustement des normales.

La contribution énergétique apportée par un point $\mathbf{p}$ diminue lorsque ce point s'approche de la forme dont il est le plus proche et lorsque la normale qui lui est associée s'aligne avec la normale théorique $\mathbf{v}$. Toutefois, la primitive $\mathcal{S}$ est considérée comme mal représentée par les points appartenant à un voisinage trop distant (entre ε et 3ε ici) ou ceux dont la normale est trop éloignée de la normale à la surface. De tels points augmentent ainsi l'énergie de par leur contribution positive. Les points étant trop manifestement trop éloignés de toute primitive de $\mathcal{X}$ (au delà de 3ε) ne contribuent pas à l'énergie, puisqu'ils ne décrivent tout simplement pas le modèle.

2.3.2. Terme *a priori*

La densité *a priori* quantifie la pertinence de $\mathcal{X}$ indépendamment de $\mathscr{P}$. Elle définit la pertinence de toute configuration uniquement sur la base de critères *a priori*. Les deux critères que nous proposons dans le processus de reconstruction sont la qualité de la connectivité entre éléments et la ressemblance au modèle CAO *a priori* étant données des tolérances σ_A, σ_R et σ_C vis à vis (respectivement) des changements de position, d'orientation et de géométrie.
Il semble donc tout à fait naturel de séparer l'énergie E_P en deux termes distincts : une énergie topologique E_T basée sur la connectivité des éléments de $\mathcal{X}$, une énergie géométrique E_G basée sur la ressemblance entre $\mathcal{X}$ et $\mathcal{M}_0$:

$$E_P(\mathcal{X},\mathcal{M}_0) = \lambda_T.E_T(\mathcal{X}) + \lambda_G.E_G(\mathcal{X},\mathcal{M}_0) \quad (7)$$

où les coefficients λ_T et λ_G modulent l'importance relative de chacun des termes.

2.3.2.1. Énergie géométrique Nous définissons E_G comme la somme des contributions de chaque primitive de $\mathcal{X}$: chacune est comparée avec la forme de $\mathcal{M}_0$ qui semble lui être la plus similaire, et apporte l'énergie correspondante :

$$E_G(\mathcal{X},\mathcal{M}_0) = \sum_{\mathcal{S}_i \in \mathcal{X}} \min_{\mathcal{S}^j \in \mathcal{M}_0} e_G(\mathcal{S}_i, \mathcal{S}^j) \quad (8)$$

L'expression de e_G dépend du type des primitives considérées. Mais cette contribution est supposée infinie lorsque le type de $\mathcal{S}_i$ et $\mathcal{S}^j$ diffèrent.

Cylindres : Soient $\mathcal{C}_i \in \mathcal{X}$ et $\mathcal{C}^j \in \mathcal{M}_0$ deux cylindres. e_G est séparé en trois termes (un pour chaque paramètre principal définissant les cylindres) :

$$e_G(\mathcal{C}_i,\mathcal{C}^j) = e^A_G(\mathcal{C}_i,\mathcal{C}^j) + e^R_G(\mathcal{C}_i,\mathcal{C}^j) + e^C_G(\mathcal{C}_i,\mathcal{C}^j) \quad (9)$$

Le terme angulaire e^A_G diminue lorsque les axes des cylindres s'alignent :

$$e^A_G(\mathcal{C}_i,\mathcal{C}^j) = \left(\frac{\arccos(|\mathbf{a}_{\mathcal{C}_i}\cdot\mathbf{a}_{\mathcal{C}^j}|)}{\sigma_A}\right)^2 \quad (10)$$

La terme radial e^R_G diminue lorsque les valeurs des rayons se

rapprochent :

$$e_G^R(\mathcal{C}_i,\mathcal{C}^j) = \left(\frac{r_{\mathcal{C}_i} - r_{\mathcal{C}^j}}{\sigma_R . r_{\mathcal{C}^j}}\right)^2 \quad (11)$$

Le terme d'éloignement e_G^C diminue lorsque la position de $\mathcal{C}_i$ s'approche de l'axe de $\mathcal{C}^j$:

$$e_G^C(\mathcal{C}_i,\mathcal{C}^j) = \left(\frac{\|(\mathbf{c}_{\mathcal{C}_i} - \mathbf{c}_{\mathcal{C}^j}) \times \mathbf{a}_{\mathcal{C}^j}\|}{\sigma_C}\right)^2 \quad (12)$$

"$\times$" et "$\|\ \|$" correspondent respectivement au produit vectoriel et à la norme euclidienne dans $\mathbb{R}^3$.

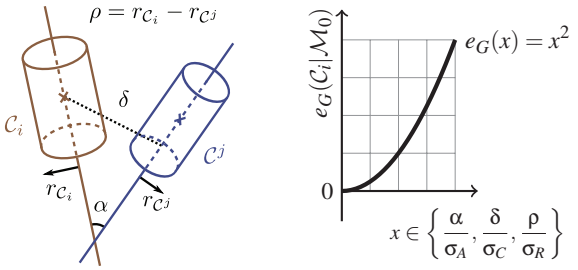


Figure 3: Contribution d'un cylindre $\mathcal{C}_i \in \mathcal{X}$ à l'énergie géométrique E_G. $\mathcal{C}_i$ est confronté au cylindre $\mathcal{C}^j$ de $\mathcal{M}_0$ dont il est le plus semblable.

Plans : Soient $\mathcal{P}_i \in \mathcal{X}$ et $\mathcal{P}^j \in \mathcal{M}_0$ deux plans. e_G est séparé en deux termes :

$$e_G(\mathcal{P}_i,\mathcal{P}^j) = e_G^A(\mathcal{P}_i,\mathcal{P}^j) + e_G^C(\mathcal{P}_i,\mathcal{P}^j) \quad (13)$$

Le terme angulaire e_G^A diminue lorsque les vecteur normaux aux plans s'alignent. On réutilisera l'expression 10 en substituant aux axes de cylindres les vecteurs $\mathbf{n}_{\mathcal{P}_i}$ et $\mathbf{n}_{\mathcal{P}^j}$ normaux aux plans $\mathcal{P}_i$ et $\mathcal{P}^j$.

Quant au terme d'éloignement e_G^C, il diminue lorsque la position de $\mathcal{P}_i$ s'éloigne du plan $\mathcal{P}^j$

$$e_G^C(\mathcal{P}_i,\mathcal{P}^j) = \left(\frac{(\mathbf{c}_{\mathcal{P}_i} - \mathbf{c}_{\mathcal{P}^j}) \cdot \mathbf{n}_{\mathcal{P}^j}}{\sigma_C}\right)^2 \quad (14)$$

2.3.2.2. Énergie topologique Concernant la définition de l'énergie topologique E_T, nous nous basons sur les hypothèses suivantes inspirées des connaissances métier d'assemblage des composants dans des environnement industriels fortement structurés :

- Cylindre - Cylindre :
  - les axes des cylindres connectés doivent se couper
  - les cylindres ne doivent pas se chevaucher : le volume d'intersection de deux cylindres doit être négligeable par rapport au volume des cylindres eux-mêmes
  - l'angle entre deux cylindres connectés doit être droit (90°) ou plat (0°),
  - les connexions impliquant deux cylindres de même rayon sont pertinentes, bien qu'il ne soit pas impossible de rencontrer des connexions entre cylindres ayant des rayons très différents (e.g. piquages).
- Plan - Plan
  - les vecteurs normaux de plans connectés doivent être orthogonaux
- Cylindre - Plan
  - le vecteur normal à un plan connecté à un cylindre doit être aligné avec l'axe de ce cylindre

Nous définissons l'énergie E_T comme la somme des énergies à chaque connexion :

$$E_T(\mathcal{X}) = \sum_{(\mathcal{S}_i \in \mathcal{X}, \mathcal{S}_j \in \mathcal{X}), i \neq j} e_T(\mathcal{S}_i,\mathcal{S}_j) \quad (15)$$

Cylindre - Cylindre

$$e_T(\mathcal{C}_i,\mathcal{C}_j) = \begin{cases} 0 & \text{si } \mathcal{C}_i \cap \mathcal{C}_j = \emptyset \\ g_\tau(\mathcal{C}_i,\mathcal{C}_j) & \text{si } 0 < \mathcal{V}(\mathcal{C}_i \cap \mathcal{C}_j) < \frac{1}{2} \\ \omega_T & \text{si } \frac{1}{2} \leq \mathcal{V}(\mathcal{C}_i \cap \mathcal{C}_j) \end{cases} \quad (16)$$

$\mathcal{V}(\mathcal{C}_i \cap \mathcal{C}_j)$ est le maximum entre la proportion du volume de $\mathcal{C}_i$ intersectant $\mathcal{C}_j$ et la proportion du volume de $\mathcal{C}_j$ intersectant $\mathcal{C}_i$. Cette quantité peut être grossièrement approximée en utilisant une méthode de type Monte Carlo : quelques centaines de points sont tirés aléatoirement dans chaque cylindre, et il suffit ensuite de compter la proportion de ces points se trouvant simultanément dans les deux cylindres (nous ne conservons que le maximum des deux proportions). De cette manière, les cylindres se chevauchant peuvent être fortement pénalisés à l'aide de la constante positive ω_T. Les cylindres qui ne sont en contact avec aucun autre cylindre n'interviennent pas dans le calcul de l'énergie topologique. Dans le cas général, nous définissons une énergie favorisant les connexions satisfaisants les attentes mentionnées ci-avant (intersection des axes, angles droits ou plats, rayons identiques) :

$$g_\tau(\mathcal{C}_i,\mathcal{C}_j) = g_\tau^L(\mathcal{C}_i,\mathcal{C}_j) + g_\tau^R(\mathcal{C}_i,\mathcal{C}_j) + g_\tau^A(\mathcal{C}_i,\mathcal{C}_j) \quad (17)$$

Cette fois encore, nous proposons trois termes distincts, représentant chacun une contrainte. Puisque le centre $\mathbf{c}_{\mathcal{C}_i}$ du cylindre $\mathcal{C}_i$ (resp. $\mathbf{c}_{\mathcal{C}_j}$ le centre de $\mathcal{C}_j$) et la direction de son axe $\mathbf{a}_{\mathcal{C}_i}$ (resp. $\mathbf{a}_{\mathcal{C}_j}$) définissent une droite dans $\mathbb{R}^3$, et étant donnée une fonction $d\left((\mathbf{c}_{\mathcal{C}_i}, \mathbf{a}_{\mathcal{C}_i}), (\mathbf{c}_{\mathcal{C}_j}, \mathbf{a}_{\mathcal{C}_j})\right)$ de distance entre deux droites, la contrainte d'intersection des axes est quantifiée par l'énergie :

$$g_\tau^L(\mathcal{C}_i,\mathcal{C}_j) = 3 - 4\left(\frac{3}{4}\right)\left(\frac{d\left((\mathbf{c}_{\mathcal{C}_i}, \mathbf{a}_{\mathcal{C}_i}), (\mathbf{c}_{\mathcal{C}_j}, \mathbf{a}_{\mathcal{C}_j})\right)}{\tau \times \min\left(r_{\mathcal{C}_i}, r_{\mathcal{C}_j}\right)}\right)^2 \quad (18)$$

où τ est une tolérance spécifiée par l'utilisateur.

Le terme angulaire g_τ^A favorise les connexions à angle droit ou plat (énergie négative), tout en pénalisant les autres

angles (énergie positive) :

$$g_\tau^A(\mathcal{C}_i,\mathcal{C}_j) = 3 - 4\left(\frac{3}{4}\right)\left(\frac{\arccos(|\mathbf{a}_{\mathcal{C}_i}\cdot\mathbf{a}_{\mathcal{C}_j}|)}{a}\right)^2 - 4\left(\frac{3}{4}\right)\left(\frac{\arccos(|\mathbf{a}_{\mathcal{C}_i}\cdot\mathbf{a}_{\mathcal{C}_j}|)-\frac{\pi}{2}}{a}\right)^2 \quad (19)$$

a étant une tolérance spécifiée par l'utilisateur.

Pour finir, le terme radial g_τ^R favorise les connexions impliquant des cylindres de même rayon. Toutefois, cette énergie est tout à fait neutre vis-à-vis des situations où cette attente n'est pas satisfaite. Pour être considérés comme égaux, la différence des rayons ne doit pas être significativement plus grande que le paramètre de bruit ε (cf. Section 2.3.1) :

$$g_\tau^R(\mathcal{C}_i,\mathcal{C}_j) = -\mathrm{e}^{-\frac{1}{2}\left(\frac{r_{\mathcal{C}_i}-r_{\mathcal{C}_j}}{\varepsilon}\right)^2} \quad (20)$$

Les termes g_τ^L et g_τ^A définissent des contraintes dures : lorsque l'une d'entre elles n'est pas satisfaite, l'énergie correspondante tend vers son maximum (en l'occurrence 3), et la somme g_τ devient automatiquement une pénalité positive puisque chacun des deux autres termes est supérieur à -1. Ceci justifie le recours aux valeurs 3 et 4 dans les Équations 18 et 19, bien que l'on puisse toutefois leur substituer toute autre paire $(\gamma, \gamma+1)$, tant que $\gamma > 2$.

Plan - Plan :

La seule contrainte applicable aux connections entre deux plans $\mathcal{P}_i$ et $\mathcal{P}_j$ concerne leur orthogonalité. On pose donc :

$$g_\tau(\mathcal{P}_i,\mathcal{P}_j) = 3 - 4\left(\frac{3}{4}\right)\left(\frac{\arccos(\mathbf{n}_{\mathcal{P}_i}\cdot\mathbf{n}_{\mathcal{P}_j})-\frac{\pi}{2}}{a}\right)^2 \quad (21)$$

Cylindre - Plan : De même, la seule contrainte concernant les connections entre un cylindre $\mathcal{C}_i$ et un plan $\mathcal{P}_j$ nous conduit à poser :

$$g_\tau(\mathcal{C}_i,\mathcal{P}_j) = 3 - 4\left(\frac{3}{4}\right)\left(\frac{\arccos(\mathbf{a}_{\mathcal{C}_i}\cdot\mathbf{n}_{\mathcal{P}_j})}{a}\right)^2 \quad (22)$$

3. Résolution du problème d'optimisation

3.1. Optimisation gloutonne

Nous cherchons à générer une configuration $\mathcal{X}$ minimisant l'énergie E (cf. Section 2.2), mais nous ne savons pas à l'avance combien de primitives exactement composent $\mathcal{X}$. Les outils habituels d'optimisation numérique ne permettent donc pas vraiment de traiter ce problème.
Dans [DMZ08, LGD10], les auteurs résolvent un problème similaire à l'aide d'approches basées sur les chaînes de Markov réversibles couplées à un recuit simulé. En effet, l'espace des configurations peut être exploré en "sautant" aléatoirement d'une configuration à une autre, chaque saut étant une opération d'ajout ou de retrait d'un élément (méthode "birth and death"). Ces transitions sont ensuite acceptées aléatoirement, selon l'évolution de l'énergie. Ce processus décrit une chaine de Markov réversible dont la distribution stationnaire correspond à l'énergie que l'on souhaite minimiser, assurant ainsi la convergence théorique de la résolution vers le minimum global.

Pour notre part, nous pensons que la génération d'un élément devrait plutôt être guidée à l'aide de l'ensemble des données disponibles, à savoir $\mathscr{P}$ et $\mathcal{M}_0$ (cf. Section 3.2). En effet, la génération aléatoire naïve d'éléments, en proposant leurs paramètres selon des lois uniformes, conduirait à un temps de résolution prohibitif puisqu'il est très peu probable qu'une telle primitive améliore la configuration courante au sens de l'énergie E. En introduisant un biais dans le noyau de naissance (génération de nouveaux éléments), nous pouvons favoriser l'apparition d'éléments pertinents mais ne pouvons plus garantir la réversibilité du processus de Markov. De sorte qu'on ne peut plus assurer la convergence théorique du processus.
De plus, le schéma de refroidissement du recuit simulé adopté joue un rôle crucial dans la convergence du processus, et requiert de la part de l'utilisateur de spécifier certains nouveaux paramètres très difficiles à appréhender.

Algorithme 1 : Algorithme d'optimisation glouton

Données : $\mathscr{P}$: nuage de points à reconstruire
$\mathcal{M}_0$: modèle CAO approximant grossièrement $\mathscr{P}$
Résultat : $\mathcal{X}$ s'ajuste sur $\mathscr{P}$, "ressemble" à $\mathcal{M}_0$ et est consistante

$\mathcal{X} \leftarrow \emptyset$;
Répéter
  Pour tout $\mathcal{S}^i \in \mathcal{M}_0$ **faire**
    Générer "aléatoirement" une primitive $\mathcal{S}$ à partir de $\mathscr{P}$ et $\mathcal{S}^i$;
    $\mathcal{X}^* \leftarrow \mathcal{X} \cup \{\mathcal{S}\}, \quad E^0 \leftarrow E(\mathcal{X}^*, \mathscr{P}, \mathcal{M}_0)$;
    Calculer l'ensemble $\mathcal{K}$ des primitives de $\mathcal{X}$ en contact avec $\mathcal{S}$;
    Trier $\mathcal{K}$ par $E(\mathcal{X}^* \setminus \{\mathcal{S}_j \in \mathcal{K}\}, \mathscr{P}, \mathcal{M}_0)$ croissant;
    $\mathcal{K}^* \leftarrow \emptyset, \quad \mathcal{Q} \leftarrow \emptyset$;
    Pour tout $\mathcal{S}_j \in \mathcal{K}$ **faire**
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathcal{S}_j\}, \quad E^j \leftarrow E(\mathcal{X}^* \setminus \mathcal{Q}, \mathscr{P}, \mathcal{M}_0)$;
      Si $E^j < E^0$ **alors**
        $E^0 \leftarrow E^j, \quad \mathcal{K}^* \leftarrow \mathcal{Q}$;
    Si $E^0 < E(\mathcal{X}, \mathscr{P}, \mathcal{M}_0)$ **alors**
      $\mathcal{X} \leftarrow \mathcal{X}^* - \mathcal{K}^*$;
jusqu'à $\mathcal{X}$ *n'a pas été modifié* ;
Retourner $\mathcal{X}$

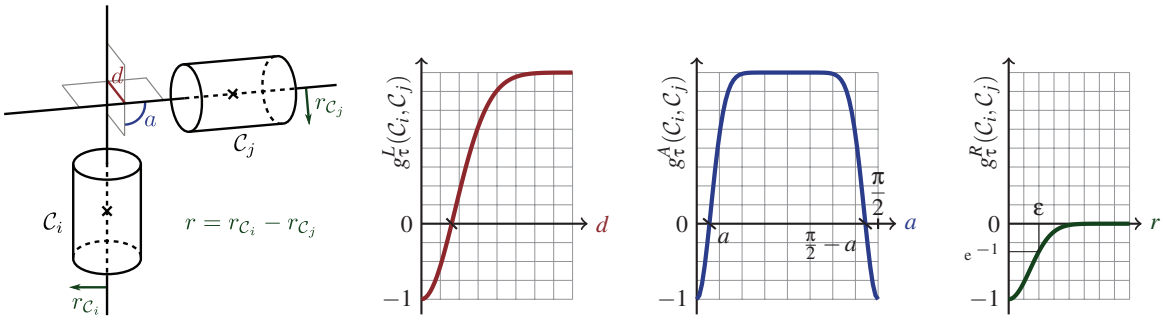


Figure 4: Contribution de toute paire de cylindres connectés $(\mathcal{C}_i,\mathcal{C}_j) \in \mathcal{X}^2$ à l'énergie topologique E_T.

C'est pourquoi, au lieu d'utiliser un algorithme probabiliste, nous proposons d'utiliser une méthode gloutonne (Algorithme 1) reposant sur des mécanismes similaires d'exploration stochastique de l'espace des solutions. La décision portant sur l'acceptation ou non d'une transition est prise de manière déterministe en vue de minimiser systématiquement E à chaque pas. Lorsqu'un nouveau candidat $\mathcal{S}$ est généré (cf. Section 3.2), nous tentons de l'insérer dans la configuration $\mathcal{X}$. Il se peut que cette insertion augmente l'énergie bien que $\mathcal{S}$ soit un candidat pertinent, à cause de conflits avec d'autres primitives de $\mathcal{X}$. Avant toute prise de décision, nous devons donc résoudre ces conflits. Soit $\mathcal{K} \subset \mathcal{X}$ l'ensemble des primitives de $\mathcal{X}$ en contact avec $\mathcal{S}$. Il nous faut trouver le sous-ensemble $\mathcal{K}^* \subset \mathcal{K}$ qui minimise $E(\mathcal{X} \setminus \mathcal{K}^* \cup \{\mathcal{S}\}, \mathscr{P}, \mathcal{M}_0)$, i.e. la suppression optimale de conflits. En théorie, nous devrions tester toute les parties de l'ensemble $\mathcal{K}$ pour trouver $\mathcal{K}^*$. Mais pour éviter une explosion combinatoire, nous proposons d'utiliser une heuristique gloutonne permettant d'approximer $\mathcal{K}^*$: les primitives $\mathcal{S}_j$ de $\mathcal{K}$ dont la suppression conduit à l'énergie la plus basse sont supprimées en premier. En supprimant les éléments conflictuels dans cet ordre, nous pouvons garder une trace du sous-ensemble atteignant l'énergie minimale en un temps proportionnel à $|\mathcal{K}^*|$ au lieu de $2^{|\mathcal{K}^*|}$. Finalement, une fois les conflits résolus, le cylindre $\mathcal{C}$ est accepté et la suppression des conflits validée si et seulement si l'énergie atteinte est moins élevée.

3.2. Génération de primitives candidates

Dans notre cas, nous disposons d'une estimation de la solution que nous recherchons et cette connaissance *a priori* $\mathcal{M}_0$ peut être utilisée pour générer des primitives pertinentes dans le nuage de points, et ainsi améliorer la convergence du processus d'optimisation. Nous proposons tout d'abord une méthode pour la génération de cylindres. Cette méthode pourra ensuite être étendue à la génération de plans.

3.2.0.3. Génération de cylindres Étant donné un cylindre $\mathcal{C}^i \in \mathcal{M}_0$, nous proposons de construire un cylindre candidat en utilisant une approche selon laquelle les primitives sont créées à partir de points du nuage, à la manière de l'algorithme RANSAC. Dans un premier temps, nous localisons la zone dans le nuage de points où le candidat va être créé, de manière à favoriser l'apparition de candidats qui soient proches de $\mathcal{C}^i$ et dont l'orientation soit cohérente avec celle de $\mathcal{C}^i$. Pour ce faire, nous définissons en tout point $(\mathbf{p},\mathbf{n})$ du nuage une probabilité :

$$P((\mathbf{p},\mathbf{n})|\mathcal{C}^i) = \mathrm{e}^{-\left(\frac{d(\mathcal{C}^i,\mathbf{p})}{\sigma_C}\right)^2 - \left(\frac{\arccos(|\mathbf{n}\cdot\mathbf{a}_{\mathcal{C}^i}|) - \frac{\pi}{2}}{\sigma_A}\right)^2} \quad (23)$$

Rappelons que $d(\mathcal{C},\mathbf{p})$ est la distance séparant $\mathbf{p}$ de $\mathcal{C}^i$, et σ_C ainsi que σ_A ont été introduits dans les définition de l'énergie géométrique E_G. Nous sélectionnons ensuite un premier point qui semble pertinent vis à vis de $\mathcal{C}^i$ de part sa distance et la normale qui lui est associée, à l'aide d'une méthode d'acceptation/rejet simulant la probabilité ci-dessus.

Nous choisissons ensuite un voisinage sphérique de ce premier point retenu, dont la taille est estimée en nous basant sur le rayon de $\mathcal{C}^i$. En fait, le centre de ce voisinage est obtenu en coulissant le premier point sélectionné le long de sa normale, de manière à positionner le centre du voisinage à l'intérieur de la forme échantillonnée par le nuage de points. Parmi les points figurant dans ce voisinage, nous n'en retenons qu'un certain nombre susceptibles de ce trouver sur la même surface que le premier point sélectionné, en nous basant sur un ajustement grossier de cylindre sur tous les points du voisinage (détecté par RANSAC). Pour finir, le cylindre candidat est calculé comme étant la forme s'ajustant au mieux sur les points retenus, par minimisation au sens des moindres carrés (initialisée à l'aide de l'estimation locale grossière mentionnée ci-avant).

Cet algorithme génère aléatoirement des cylindres pertinents tendant à ressembler au cylindre *a priori* $\mathcal{C}^i$, et s'ajustent bien dans le nuage de points $\mathscr{P}$. En effet, nous utilisons les paramètres de $\mathcal{C}^i$ pour guider la sélection des points. Et puisque cette sélection s'effectue selon un processus fiable, nous pouvons générer le cylindre candidat à partir d'une grande quantité de points (quelques centaines), alors

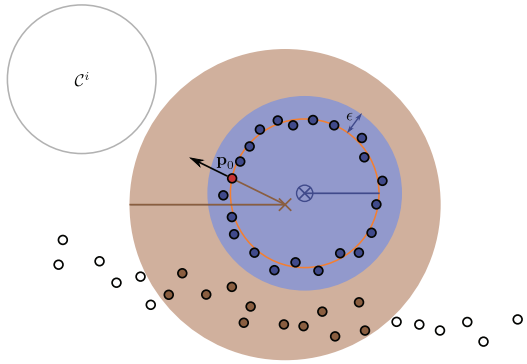


Figure 5: Sélection des points pour la génération d'un cylindre. On choisit un premier point $\mathbf{p}_0$ à partir de l'*a priori* $\mathcal{C}^i$. Puis on considère un premier voisinage (marron) excentré à partir de $\mathbf{p}_0$ de manière à positionner le centre à l'intérieur du cylindre échantillonné. Après avoir détecté un cylindre (cercle orange) parmi les points de ce voisinage, on ne conserve que les points suffisamment proches (bleu) pour construire le candidat final.

que les méthodes RANSAC habituelles ne peuvent fonctionner qu'avec des ensembles minimaux de points (2 à 7). Cette méthode rend l'étape de génération plus précise et robuste envers le bruit et les incertitudes sur les normales. La probabilité pour un cylindre d'être généré augmente donc lorsque les énergies E_D et E_G correspondantes diminuent.
De plus, l'ensemble des cylindres pouvant être construits par cette méthode est fini et dénombrable, et le cardinal de cet ensemble est inférieur ou égal au nombre de parties de $\mathscr{P}$ possédant κ points. En particulier, les cylindres n'étant pas supportés par un nombre suffisant de points ont une probabilité nulle d'apparaître. La méthode que nous proposons définit donc un espace probabilisé sur l'espace des cylindres, ayant un ensemble fini dénombrable d'événements de probabilité non nulle, et cette méthode peut donc être utilisée pour l'exploration efficace de l'espace des solutions en vue de résoudre le problème Bayésien formulé dans cet article.

3.2.0.4. Génération de plans La méthode de génération de cylindre peut être utilisée pour générer des plans étant donné un *a priori* $\mathcal{P}^i \in \mathcal{M}_0$. Il suffit pour cela de redéfinir la probabilité en tout point du nuage pour l'adapter au plan :

$$P((\mathbf{p},\mathbf{n})|\mathcal{P}^i) = \mathrm{e}^{-\left(\frac{d(\mathcal{P}^i,\mathbf{p})}{\sigma_C}\right)^2 - \left(\frac{\arccos(|\mathbf{n}\cdot\mathbf{n}_{\mathcal{P}^i}|)}{\sigma_A}\right)^2} \tag{24}$$

4. Reconstruction des jonctions toriques

Une fois les cylindres convenablement détectés et connectés, nous proposons de finaliser le reconstruction en modélisant les coudes de tuyauterie par des tores lorsque c'est nécessaire. Pour ce faire, nous considérons toutes les connections entre paires de cylindres, et construisons une section de tore circulaire qui relie les cylindres et s'ajuste sur le nuage de points au sens des moindres carrés. Si ce nouvel élément diminue l'énergie d'attache aux données E_D de la configuration, il est alors intégré à la configuration. Dans le cas contraire, il est refusé.

5. Résultats

5.1. Quelques mots au sujet des paramètres

Nous avons présenté plusieurs paramètres tout au long de cet article. Certains d'entre eux doivent être spécifiés par l'utilisateur, puisqu'ils apportent une connaissance au sujet du problème. Ainsi, ε et η liés au bruit dans le nuage de points $\mathscr{P}$, ou σ_A, σ_R et σ_C spécifiant l'incertitude au sujet de l'approximation initiale $\mathcal{M}_0$, ou encore les coefficients de pondération λ_D, λ_G et λ_T, doivent-ils être fournis par l'utilisateur en entrée de la méthode. Cependant, il n'est pas simple d'appréhender les paramètres λ_D, λ_G et λ_T, et une approche estimant automatiquement ces valeurs pourrait être très utile. Pour simplifier légèrement ce paramétrage, nous imposons $\lambda_D + \lambda_G + \lambda_T = 1$.

D'autres paramètres peuvent être estimés indépendamment des données en entrée. C'est par exemple le cas de la tolérance topologique τ, pour laquelle nous préconisons une valeur de 0.1 : la distance entre les axes de deux cylindres connectés ne devrait en aucun cas excéder un dixième du rayon du plus petit cylindre (cf. Équation 18). De même, nous fixons ici $a = 5^\circ$ (cf. Équation 19). Pour finir, le fait de poser $\omega_T = 100$ permet l'interdiction du chevauchement entre cylindres dans la plupart des cas testés.

5.2. Résultats sur une scène industrielle

Nous présentons ici quelques uns des résultats que nous avons obtenus sur le jeu de données présenté en Figure 1, afin de montrer la capacité de notre méthode à créer des modèles CAO consistant à partir de données laser. Nous utilisons dans cet exemple un modèle CAO *a priori* grossièrement recalé avec le nuage de points (les erreurs de recalage sont de l'ordre de 10 cm), et présentant quelques différences non négligeables avec le nuage de points : certains tuyaux sont absents et d'autres ont été significativement modifiés (réorientés et/ou déplacés). Nous utilisons donc des tolérances au changement relativement permissives : $\sigma_A = 30^\circ$, $\sigma_R = 0.2$ et $\sigma_C = 3.5$ m. La configuration reconstruite contient bien les tuyaux qui ne figuraient pourtant pas dans le modèle *a priori*, mais qui ont été détectés à cause de leur similarité avec d'autres composants voisins. De même, les tuyaux déplacés et réorientés ont été relativement bien gérés. De plus, nous pouvons constater que le modèle obtenu est consistant : ces cylindres sont bien connectés, et nous n'y voyons aucun chevauchement indésirable. Le modèle s'ajuste aussi très bien sur le nuage de points. Un autre aspect intéressant de cet algorithme tient

au fait qu'il permet la reconnaissance d'une grande variété d'objets, puisqu'il permet de détecter des cylindres dont les rayons varient entre 10 mm et 2300 mm environ. Toutefois, la détection des plus petits éléments est bien moins précise puisque leur rayon est inférieur au bruit du nuage spécifié (nous utilisons ε=30 mm).

L'évolution des énergies au cours de la résolution est présentée en Figure 6. Le processus nécessite environ 200000 itérations (une itération correspondant à la génération d'un cylindre candidat) soit environ 15 minutes de calcul sur un processeur 2.4 GHz.

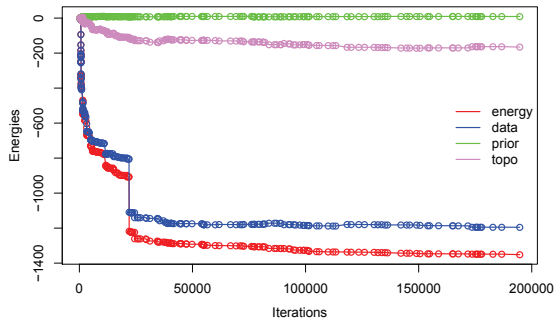


Figure 6: Évolution de l'énergie au cours du processus de reconstruction. On note que l'énergie géométrique (vert) joue un rôle de terme de contrôle : elle est positive et permet d'éviter la reconstruction d'un modèle trop éloigné de l'*a priori*, alors que les autres énergies sont décroissantes et souvent négatives.

Nous comparons aussi notre approche avec un algorithme RANSAC de référence [SWK07] permettant la détection de tores, cylindres, plans et sphères, et n'utilisant aucune information *a priori*. Cet algorithme est plus rapide que le nôtre (une trentaine de secondes). Toutefois les résultats de segmentation montrent que notre approche parvient à détecter correctement des primitives indépendantes bien connectés, alors que le RANSAC tend à détecter plusieurs fragments de formes se chevauchant, et peine à faire la distinction entre les différents types de primitive (tores à la place de cylindres, etc.).
La table ci-après présente quelques éléments de comparaison entre l'algorithme RANSAC et notre méthode. Nous pouvons constater que notre approche fournit de meilleurs résultats en matière d'ajustement des cylindres sur les points (deuxième ligne). Elle permet aussi d'éviter les conflits et chevauchements entre cylindres (quatrième ligne) et optimise la distribution des points à la surface des cylindres, signe vraisemblablement d'une meilleure qualité d'ajustement (troisième ligne).

	RANSAC	**Bayésien**
Quantité de cylindres détectés	42	68
Distance moyenne aux voisins (ε-inliers)	14.58 mm	10.12 mm
Nombre moyen de points par unité de surface	21,18 points/dm^2	25,81 points/dm^2
Points simultanément voisins de plus d'un cylindre	5.67%	0.85%

6. Conclusion

Nous avons posé le problème de la reconstruction de nuages de points comme la recherche d'un modèle CAO satisfaisant plusieurs contraintes basées sur des connaissances *a priori*. Nous avons plus particulièrement concentré nos travaux sur l'utilisation d'un modèle CAO existant décrivant grossièrement l'état de la scène à traiter. Nous utilisons cette connaissance *a priori* afin de permettre une reconstruction de modèles CAO de qualité. En effet, les approches de reconstruction existantes ne traitent, pour la plupart, le problème de reconstruction que comme un problème d'ajustement sur le nuage de points, négligeant la qualité du modèle résultant, alors que la formulation Bayésienne proposée permet d'embarquer différentes contraintes telles que la consistance du résultat ou sa conformité vis-à-vis d'un modèle *a priori* étant par hypothèse une approximation du résultat recherché. Nous proposons de plus d'utiliser le modèle *a priori* comme un outil permettant de cibler la reconstruction sur les parties de la scène dont nous savons qu'elles sont pertinentes.

La probabilité *a posteriori* que nous proposons définit nos trois attentes vis-à-vis du résultat. De plus, nous proposons une nouvelle approche qui permet la recherche de la solution maximisant cette probabilité. Cette méthode fournit des résultats satisfaisants sur le jeu de données testé : la résolution permet effectivement d'obtenir un modèle CAO ayant une énergie faible (donc une probabilité élevée), s'ajustant bien sur le nuage de points, composé de formes convenablement connectées et dont les éléments correspondent effectivement au modèle *a priori* que l'on se donne.

La méthode, de par son aspect incrémental, permet à l'utilisateur d'évaluer la solution en cours de construction et d'arrêter à tout moment le processus lorsqu'il estime qu'il est arrivé à son terme, sans avoir à attendre la convergence et l'arrêt complet du calcul.

Références

[BCM*11] BEY A., CHAINE R., MARC R., THIBAULT G., AKKOUCHE S. : Reconstruction of consistent 3d cad models from point cloud data using a priori cad models. In *ISPRS Workshop on Laser Scanning* (2011).

[Bos10] BOSCHÉ F. : Automated Recognition of 3D CAD Model Objects and Calculation of As-built Dimensions for Dimensional Compliance Control in Construc-

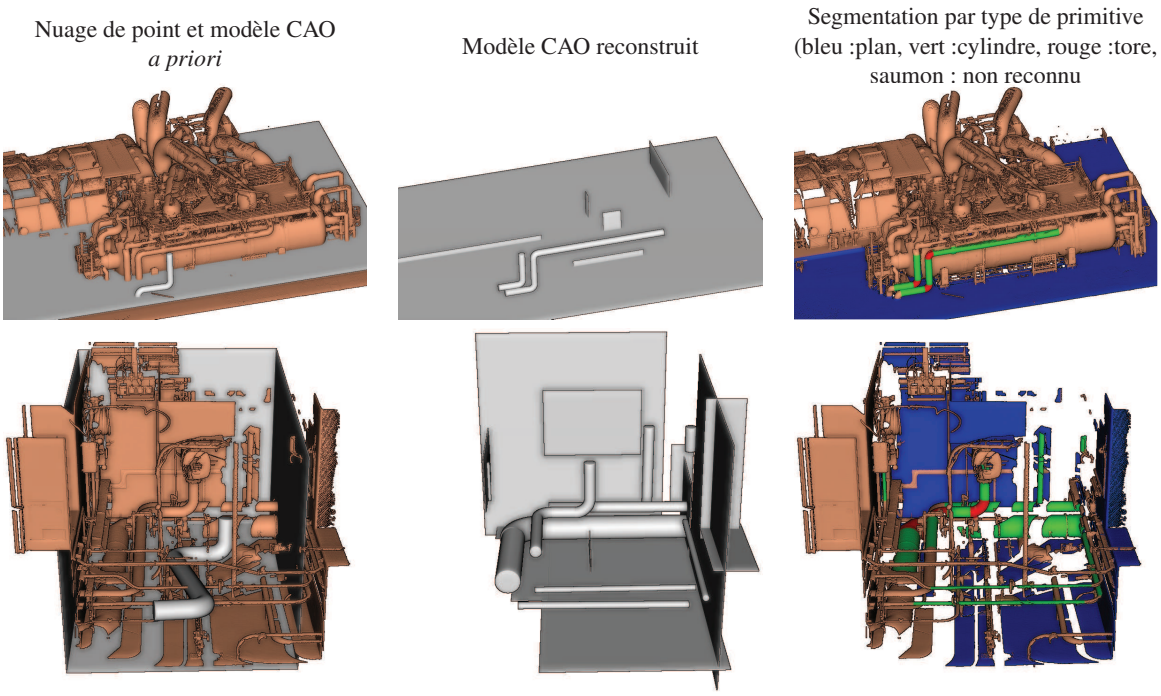


Figure 7: Quelques résultats sur d'autres scènes, en utilisant des modèles CAO partiels. L'algorithme ne reconstruit que les parties correspondant effectivement au modèle CAO *a priori*, à certaines tolérances près.

tion. *Elsevier Journal of Advanced Engineering Informatics* (2010).

[DMZ08] DESCOMBES X., MINLOS R., ZHIZHINA E. : Object Extraction Using a Stochastic Birth-and-Death Dynamics in Continuum. *Journal of Mathematical Imaging and Vision. Vol. 33*, Num. 3 (octobre 2008), 347–359.

[DTB06] DIEBEL J., THRUN S., BR\"UNIG M. : A Bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics (TOG). Vol. 25*, Num. 1 (2006), 39–59.

[FB87] FISCHLER M. A., BOLLES R. C. : Random Sample Consensus : a Paradigm for Model Fitting with applications to Image Analysis and Automated Cartography. *Readings in computer vision : issues, problems, principles, and paradigms* (1987), 726–740.

[GSH*07] GAL R., SHAMIR A., HASSNER T., PAULY M., COHEN-OR D. : Surface Reconstruction using Local Shape Priors. *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), 253–262.

[JWB*06] JENKE P., WAND M., BOKELOH M., SCHILLING A., STRASS ER W. : Bayesian point cloud reconstruction. *Computer Graphics Forum. Vol. 25*, Num. 3 (2006), 379–388.

[LGD10] LAFARGE F., GIMEL'FARB G., DESCOMBES X. : Geometric feature extraction by a multimarked point process. *IEEE transactions on pattern analysis and machine intelligence. Vol. 32*, Num. 9 (septembre 2010), 1597–609.

[MN04] MITRA N. J., NGUYEN A. : Estimating Surface Normals in Noisy Point Cloud Data. *Special Issue of International Journal of Computational Geometry and Applications. Vol. 14* (2004), 261–276.

[PMG*05] PAULY M., MITRA N., GIESEN J., GROSS M., GUIBAS L. : Example-based 3D scan completion. *Proceedings of the third Eurographics symposium on Geometry processing* (2005), 23–es.

[RV05] RABBANI T., VAN DEN HEUVEL F. : Efficient hough transform for automatic detection of cylinders in point clouds. *ISPRS WG III/3, III/4. Vol. 3* (2005), 60–65.

[SP10] SCHMITTWILKEN J., PLÜMER L. : Model-based reconstruction and classification of facade parts in 3d point clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 38*, Num. 1 (2010), 269–274.

[SWK07] SCHNABEL R., WAHL R., KLEIN R. : Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum. Vol. 26*, Num. 2 (2007), 214–226.

Modélisation géométrique d'organes pelviens par une approche offset

Thierry Bay*, Romain Raffin*, Marc Daniel*

*E-mails: (thierry.bay, romain.raffin, marc.daniel)@lsis.org
Aix-Marseille Université, LSIS UMR CNRS 7296, Domaine Universitaire de Saint-Jérôme,
Avenue Escadrille Normandie-Niemen, 13397 MARSEILLE CEDEX 20

Résumé

Dans l'optique de concevoir un simulateur patient-spécifique des organes pelviens le projet MoDyPe considère les organes en tant que surfaces épaisses. Partant d'une surface paramétrique fermée pour l'enveloppe externe, une approche offset est appliquée. Cependant, respecter d'une part l'épaisseur en tout point et d'autre part garantir l'absence d'auto-intersection est impossible si la forme présente des courbures locales trop importantes. Deux approches itératives sont présentées. La première méthode est basée sur la représentation continue paramétrique, alors que la seconde s'appuie sur une représentation discrète du modèle.

In order to design a patient-specific simulator of pelvic organs, MoDyPe project considers the organs as thick surfaces. Starting from a closed parametric surface for the outer hull, an offset approach is applied. However, respecting the thickness on the surface and ensuring the absence of self-intersection is impossible if the shape has too important local curvatures. Two iterative approaches are presented. The first method is based on continuous parametric representation, while the second presents a discrete representation to define the model.

Mots-clés : Approche offset, ajustement paramétrique, B-spline, système masses-ressorts.

1. Introduction

Les études effectuées sur la simulation d'organes cherchent à mieux comprendre des troubles mal connus de certains patients. La problématique qui nous intéresse fait référence à la statique pelvienne, dont les organes ont subi un déséquilibre dans leur configuration spatiale. Bien que la chirurgie puisse être employée pour y remédier, estimer l'impact fonctionnel d'un acte chirurgical est difficile. De nombreux outils ont été développés à cet effet, nécessitant une virtualisation de l'environnement d'intérêt [Mol97, Gro98].

La modélisation géométrique des organes pelviens présentée par la suite s'insère dans un processus préopératoire et patient-spécifique plus large (cf. [BCR*11, BCR*12]). Elle s'appuie sur la réalité physiologique en travaillant sur des surfaces épaisses : avec une approche surfacique, l'épaisseur non-négligeable des membranes ne serait pas prise en compte, alors que certaines cavités internes ont presque disparu (e.g. l'utérus) ; de même, un maillage volumique n'est pas suffisant pour les organes présentant une différence de densité entre l'intérieur et la paroi (e.g. l'urine et la paroi vésicale). L'épaisseur est ajoutée avec une approche offset. En partant d'une B-spline ajustée aux données pour l'enveloppe externe des organes, la littérature présente de nombreuses méthodes pour construire une seconde surface située à une distance donnée de la surface initiale. Après un tour d'horizon de l'existant, nous présentons une méthode développée pour les B-splines. Une seconde méthode est ensuite introduite, construisant un offset discret puisqu'un maillage hexaédrique est à fournir en sortie. Une comparaison qualitative des deux approches est proposée.

2. Etat de l'art sur l'approche offset paramétrique

Un état de l'art résumant différentes méthodes de calcul de surfaces offsets ouvertes est présenté par [KN02]. Les techniques applicables pour des surfaces peuvent être re-

groupées en différentes catégories. La première concerne les méthodes décomposant la surface globale en carreaux de Bézier, puis calculant l'offset de chaque carreau pour finalement fusionner tous les résultats. La continuité est généralement C^0 ou C^1. Nous pouvons citer la méthode de Hoschek [HSW89] avec une interpolation hermitiennne, celle de Farouki [Far86] avec des surfaces de degré 5, ou encore celle des moindres carrés appliqués sur chaque carreau. La seconde catégorie traite des méthodes travaillant sur la surface entière : une méthode basée sur les moindres carrés sans décomposition de la surface permet d'aboutir à une continuité C^2 pour les points à l'intérieur (à différencier des points sur les frontières). Finalement, le dernier groupe fait référence aux méthodes géométriques, basées par exemple sur l'offset du réseau de contrôle [TH84] amenant à une continuité C^2.

Le problème connu des offsets est celui des auto-intersections locales et globales lorsque la courbure est trop importante par rapport à la distance-offset. Les méthodes se veulent très différentes pour apporter une réponse à cette question, travaillant soit sur le nuage-offset, soit directement sur la surface paramétrique. Dans [KSP02], un nuage ordonné est nécessaire. Avant d'être interpolé, les échantillons situés sur une ligne par exemple sont reliés en une seule polyligne pour vérifier la présence d'intersections. Le travail est fait sur chaque ligne et chaque colonne. Puisque les points créant des intersections sont supprimés, il faut en ajouter dans les zones nettoyées jusqu'à satisfaire le nombre de points initial, impliquant une densité importante dans les zones à forte courbure (difficulté au niveau de l'interpolation et temps de calcul important). Dans [SNL04], un repositionnement itératif des points de contrôle est effectué pour diminuer la courbure dans les zones sujettes aux intersections locales. Après un échantillonnage basé sur la dérivée seconde, la courbure est calculée en chaque point afin de tester la présence d'intersections, et d'identifier les points de contrôle exerçant une influence dans cette zone. La méthode est cependant dite plus efficace si l'intersection locale est faite dans les directions paramétriques u et v, ce qui n'est pas toujours le cas. Cette méthode ne peut pas servir aux intersections globales. Seong [SEK06] se base sur une carte de distance entre la surface initiale et la surface-offset. Une fonction quadrivariée (basée sur la somme des deux espaces bivariées individuels) est formée. La projection des zéros de cette fonction sur l'espace paramétrique de l'offset permet de déterminer les zones d'intersection avec la surface-offset. La limitation de l'algorithme se situe au niveau des petites intersections qui peuvent ne pas être décelées. Perkerman [PEK08] présente une méthode de résolution d'intersections en déterminant les points antipodaux des boucles dans les surfaces (points appartenant à la surface, partageant la même normale mais de direction opposée). Une fonction d'erreurs caractérisant la distance entre ces points est définie puis réduite jusqu'à disparition des intersections.

Déterminer un offset avec une représentation paramétrique nécessite de se débarrasser des intersections. Pour cela, trois choix sont possibles : modifier l'espace de départ, i.e. les points à choisir pour la discrétisation de la surface initiale susceptibles a priori de créer des intersections, agir au niveau de la création du nuage-offset en empêchant la génération de points pouvant créer des problèmes (mais non nécessairement leur suppression), ou modifier la surface après création de l'offset pour se débarrasser a posteriori des intersections. Les méthodes agissent généralement à ce troisième niveau.

Nous présentons au paragraphe 3 un autre algorithme à classer dans cette catégorie. Il a l'avantage d'être basé sur les moindres carrés comme le processus d'ajustement paramétrique utilisé pour la construction de la surface.

3. Une épaisseur par une approche paramétrique

L'opérateur offset dans cette partie repose sur l'ajustement paramétrique présentée dans [BCR*12] : une fonction d'énergie bidirectionnelle décrit l'ensemble des distances des données décrivant les organes de la patiente vers l'échantillonnage de la surface et vice-versa, puis est minimisée par une descente de gradient.

Afin d'améliorer les tests à effectuer sur le maillage lors de l'application des lois de comportement, les hexaèdres du maillage en sortie doivent être uniformes, voire quasi-uniformes. Or, l'épaisseur d'un organe comme la vessie peut atteindre 10% de sa taille [SSS*10], nécessitant plusieurs couches offsets dans la membrane si la discrétisation est dense. Plusieurs paramètres sont alors à définir afin de créer l'épaisseur : la distance-offset d, le sens de l'offset (choisi vers l'intérieur), le nombre de couches-offsets N_{lay} (générant $N_{lay}-1$ couches d'hexaèdres) et la répartition des couches dans l'épaisseur.

3.1. L'opérateur offset

Soit S_0 la surface bivariée issue de l'ajustement aux données de la patiente. La construction de la surface-offset S_1 située à une distance h_0 de S_0 consiste en quatre étapes majeures :

1. construire l'ensemble $\left\{S_0(u_i,v_j)\right\}_{i,j=0}^{M,N}$, échantillonnage uniforme de S_0 ;
2. calculer la normale en chaque point $S_0(u_i,v_j)$:

$$n_0(u_i,v_j) = \left.\frac{\frac{\partial S_0(u,v)}{\partial u}\times\frac{\partial S_0(u,v)}{\partial v}}{\left\|\frac{\partial S_0(u,v)}{\partial u}\times\frac{\partial S_0(u,v)}{\partial v}\right\|}\right|_{(u_i,v_j)} \quad (1)$$

3. créer le nuage-offset D_o à partir de l'échantillonnage de S_0 et des normales :

$$D_o = \left\{S_0(u_i,v_j) + h_0.n_0(u_i,v_j)\right\}_{i,j=0}^{M,N} \quad (2)$$

4. ajuster itérativement une surface paramétrique fermée S_1 sur l'ensemble D_o pour obtenir la surface-offset (cf. [BCR*12]).

Le même processus est répété jusqu'à obtention du nombre de couches N_{lay}, si ce n'est qu'à chaque nouvelle surface les données considérées proviennent de la discrétisation de la surface précédente. Soit $\{S_l\}_{l=1}^{N_{lay}}$ l'ensemble des surfaces-offsets. Chaque surface S_l est décalée de la surface S_{l-1} d'une distance h_{l-1}. Afin de respecter finalement la distance-offset, la somme des épaisseurs entre deux couches doit vérifier :

$$\sum_{l=0}^{N_{lay}-1} h_l = d \tag{3}$$

Afin d'avoir un maillage final conforme, la carte paramétrique est fixée au départ. Les quadrangles entre deux surfaces-offsets successives pour un même couple paramétrique sont alors très proches.

3.2. Résultats

Les simulations sont obtenues avec un Intel Core i7 M620 (2.67 GHz, 4 GB RAM). La B-spline initiale est d'ordre 4 dans les deux directions paramétriques. Le réseau de contrôle est de 16×7. Un ajustement paramétrique est d'abord appliqué sur les formes sur 10 itérations [BCR*12].

L'algorithme est en premier lieu utilisé sur des superellipsoïdes [CJB03]. Il s'agit d'une famille de formes bien connues, issues du produit sphérique de deux superquadriques. La cardinalité de chaque ensemble de points est de 10000.

L'offset d'une sphère est tout d'abord calculé. Puisque la forme 1(a) présente peu d'irrégularités (présentes à cause de la formulation B-spline), il n'est pas étonnant que la figure 1 ne présente aucun problème dans l'épaisseur à moins de prendre une épaisseur avoisinant le rayon de la sphère.

L'histogramme de la figure 1(c) concernant la création des couches internes illustre les erreurs pour chaque sous-fonctionnelle et la distance maximale entre la surface et les données.

Dans un second exemple, une carambole est reconstruite. Bien que la surface de la figure 2(a) ne présente pas de repliement visuellement observable, l'histogramme de la figure 2(c) met en évidence des problèmes liés aux courbures de la forme. La figure 2(b) montre en effet les normales qui seront suivies par les points de la discrétisation de la première surface pour construire les nuages-offsets. Sans une méthode appropriée, il ne sera pas possible de reconstruire des formes présentant des arêtes vives comme celle-ci.

La méthode est maintenant appliquée sur les organes d'intérêt, avec des données de l'ordre de 40K. Puisque l'épaisseur est non-négligeable, la connaissance de ces valeurs est essentielle. Le tableau 1 reporte les valeurs des épaisseurs

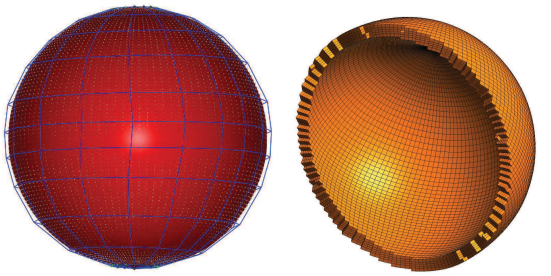
Table 1: *Epaisseurs (mm) prises pour les organes pelviens - proportion (en %) par rapport aux autres dimensions.*

	Organes			
	vessie	rectum	utérus	vagin
min	2 (2,9%)	5 (16,1%)	20 (30%)	3 (12%)
max	5 (7,5%)	6 (18,8%)	30 (44,7%)	4 (15%)

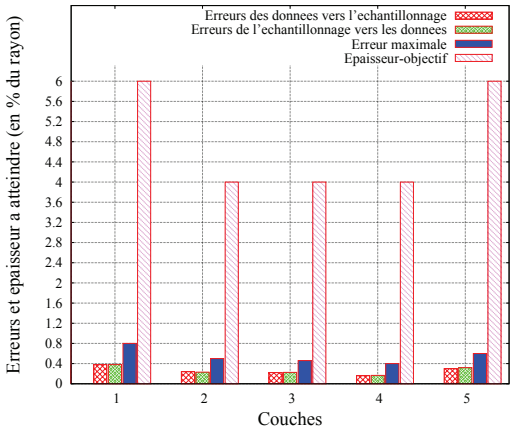
considérées pour les organes [SSS*10]. Les normales sont orientées vers l'intérieur des formes, puisque les données décrivent l'extérieur des organes.

Ces valeurs varient beaucoup d'une femme à l'autre. Les épaisseurs choisies sont la moyenne des valeurs min et max pour chaque organe : 3,5 mm pour la vessie, 5,5 mm pour le rectum et 25 mm pour l'utérus. Le vagin n'est pas considéré, car la segmentation du vagin ne peut être séparée de celle de l'utérus. Les résultats mis en évidence par la suite proviennent de la vessie et du rectum.

Concernant la vessie, la figure 3(a) illustre la surface

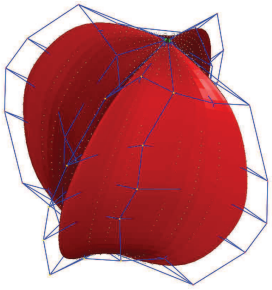


(a) Reconstruction surfacique (b) Maillage avec 5 couches

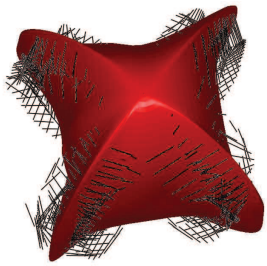


(c) Erreurs maximales et quadratiques moyennes par couche

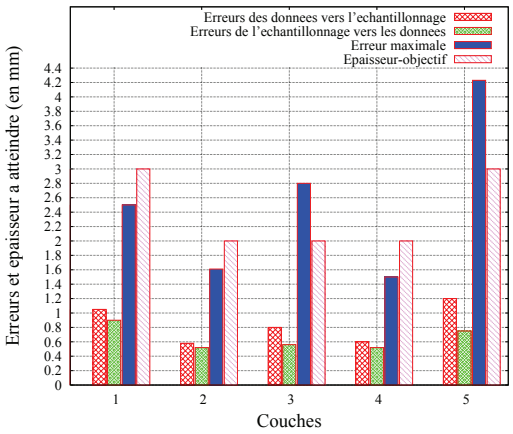
Figure 1: *Offset d'une superellipsoïde de type sphère de rayon* 50 *unités de l'extérieur vers l'intérieur, avec une distance-offset de* 12 *unités.*



(a) Reconstruction surfacique et réseau de contrôle

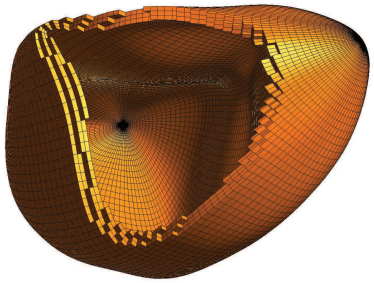


(b) Normales associées

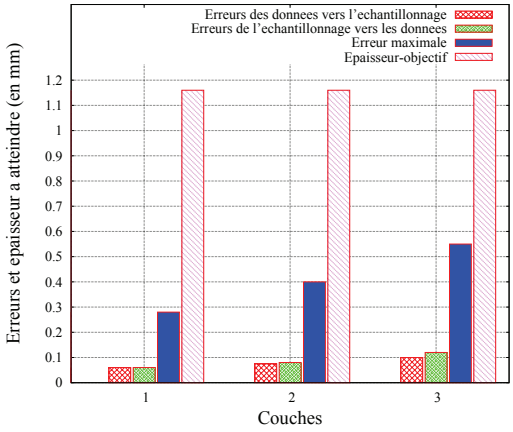


(c) Erreurs maximales et quadratiques moyennes par couche

Figure 2: *Offset d'une super-ellipsoïde de type carambole de* 100 *unités de rayon, de l'extérieur vers l'intérieur.*



(a) Epaisseur de 3,5 mm avec 3 couches



(b) Erreurs maximales et quadratiques moyennes par couche

Figure 3: *Offset d'une paroi vésicale, de l'extérieur vers l'intérieur.*

épaisse obtenue. La construction de l'offset s'effectue sans problème puisque la forme est simple. L'épaisseur n'est pas suffisante pour créer des auto-intersections globales. L'histogramme 3(b) indique que l'erreur maximale à chaque nouvelle création de couche reste inférieure à la distance-offset à atteindre.

Les résultats sont cependant différents pour le rectum. L'épaisseur prise est de 5,5 mm. La figure 4(a) montre les auto-intersections locales à l'extrémité du rectum dues à la forte courbure. L'histogramme de la figure 4(b) illustre effectivement le problème du maximum de l'erreur entre deux couches successives.

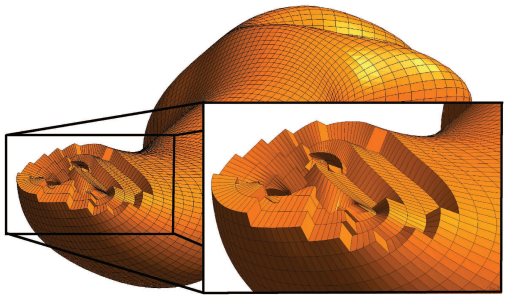
3.3. Avantages et inconvénients

Pour les organes ne présentant pas une forme complexe comme la vessie, l'épaisseur non-négligeable des organes est prise en compte, nous rapprochant par ce biais de la réalité physiologique. Les informations disponibles sur l'épaisseur des organes restent limitées (valeur globale de l'épaisseur choisie pour chaque organe). Les IRM permettent de segmenter les contours, mais pas de repérer l'intérieur.

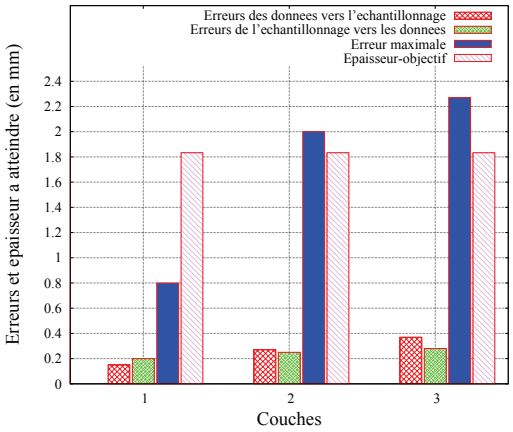
L'utilisation d'une méthode basée sur les moindres carrés permet d'avoir rapidement un algorithme fonctionnel pour construire l'épaisseur de surfaces. Mais en cas de distance-offset trop importante, l'approche moyennante n'est pas suffisante (problème aux extrémités des formes pour les organes, et sur les côtés de la carambole) et des intersections sont générées. Les chevauchements ne sont pas résorbables dans ces zones de forte courbure sans traitement approprié.

Plusieurs solutions seraient possibles pour résoudre ce problème :

1. effectuer un travail a posteriori sur la surface-offset (méthodes citées au paragraphe 2) ;
2. initialiser la surface-offset à l'intérieur strictement du nuage-offset : ainsi, la sous-fonctionnelle guidant l'échantillonnage vers les données ajusterait la surface aux données, en gardant à l'esprit que dans les zones convexes, la surface risque d'être bloquée si la première sous-fonctionnelle des données vers l'échantillonnage n'est pas utilisée. Un arrêt de la paramétrisation du sys-



(a) Auto-intersections avec une épaisseur de 5,5 mm



(b) Erreurs maximales et quadratiques moyennes par couche

Figure 4: *Offset de la paroi d'un rectum, de l'extérieur vers l'intérieur.*

tème est possible puisque chaque échantillon recherche les données les plus proches (cf. [BCR*11]) ;

3. adopter une autre représentation : puisqu'il faut fournir un maillage hexaédrique en sortie, travailler dans le domaine discret avec des liaisons élastiques entre les points constituant les membranes se justifierait.

La définition d'une approche différente répondant à ce troisième critère est finalement présentée dans la partie 4.

4. Une approche discrète à travers un système masses-ressorts

L'approche choisie dans cette partie travaille directement sur le maillage pour créer un offset discret. Puisqu'aucune connaissance précise n'est disponible pour analyser l'épaisseur des organes si ce n'est en quelques points ponctuels sur la surface, nous allons concevoir un système masses-ressorts pour construire la surface interne. Nous différencions notre modèle des approches mécaniques classiques puisque notre objectif n'est pas de faire explicitement de la déformation.

4.1. Différentiation des modèles existants

Dans le cas général, un ensemble de $n+1$ particules $\{P_i\}_{i=0}^{n}$ compose le modèle. Chacune possède une masse m_i, une position dans l'espace 3D p_i, une vitesse $\dot{p}_i$ et une accélération $\ddot{p}_i$ à un temps donné t. Le système est gouverné par la seconde loi de Newton : $F_i = m_i\ddot{p}_i$, avec F_i la somme de l'ensemble des forces exercées au point P_i. Les forces se différencient en deux catégories majeures : les forces internes gérant les contraintes et la stabilité du système (e.g. action sur une particule de la raideur des ressorts voisins), et les forces externes (généralement la force de gravitation, d'amortissement et la réponse à la collision). Les méthodes actuelles se différencient par la nature des forces mises en jeu, ainsi que par la résolution des équations de la dynamique newtonienne.

Dans le domaine de la modélisation et de l'animation, les travaux vont de la modification de la géométrie des réseaux à un changement dans la topologie des systèmes. Terzopoulos et Waters ont conçu un système masses-ressorts pour la modélisation faciale [TW90]. Le réseau était constitué de trois couches de propriétés distinctes : le derme, la couche graisseuse et le muscle. Nous pouvons citer également les travaux de Terzopoulos sur la transition d'état solide à état liquide [TPF89]. Pour se faire, les nœuds sont connectés entre eux formant un treillis d'hexaèdres.

De nombreux travaux ont également été entrepris sur la conservation de forme. Vassilev dans [VS02] définit un nouveau type de ressort préservant le volume pour des ellipsoïdes déformables. Alors que dans les systèmes classiques une masse agit uniquement sur son voisinage immédiat, chaque élément dépend de l'état global du système grâce à un ressort supplémentaire entre chaque masse et le centre de l'ellipsoide.

Le comportement des tissus est également étudié en animation. Les treillis des réseaux sont toutefois souvent semblables. La différenciation se fait au travers de la méthode d'intégration pour résoudre la dynamique du système : schémas explicites, implicites, ordres d'intégration [DSB99, KCCP00, VMT01, MGC07]. La dualité entre précision et stabilité est toujours présente, nécessitant d'étudier l'espace de stabilité du système masses-ressorts [Shi05].

Le paragraphe 4.2 présente en détails la nature du modèle utilisé.

4.2. La création de l'épaisseur

Le choix du treillis nécessite de définir le domaine d'application et les contraintes à satisfaire. Les forces sont déterminées à la suite de cela. Le traitement de la collision n'étant pas encore géré de façon optimale, il est omis pour le moment. Un tour d'horizon de l'existant est effectué au paragraphe 4.2.2.

4.2.1. Choix des particules et des connectivités

Nous partons au départ avec un maillage hexaédrique conforme mais dégénéré. Le maillage n'a qu'une épaisseur d'hexaèdres. La première couche correspond à la discrétisation de la surface paramétrique ajustée sur les données, la seconde à la membrane interne. La dégénérescence vient du fait que les arêtes reliant les deux couches sont de longueur nulle. La connectivité entre les particules est celle issue de la discrétisation. Soit $\{P_{i,j}\}_{i,j=0}^{n,m}$ l'ensemble des $(n+1)\times(m+1)$ particules sur lesquelles nous travaillerons.

4.2.2. Contraintes à préserver et énergies

Système sans masse : bien qu'il s'agisse d'un système masses-ressorts, aucune masse n'est considérée par la suite (elles sont fixées à 1). Nous utilisons un système mécanique, mais il ne s'agit pas d'une simulation physique. La force de gravité n'entre donc pas en jeu pour les mêmes raisons.

Force externe : elle est appliquée à chaque particule $P_{i,j}$, provenant de la discrétisation de la surface paramétrique S. Un couple paramétrique (u_i, v_j) est associé à chaque point. La force externe est basée sur la formulation des approches offsets, en étant orientée dans les mêmes direction et sens que la normale en chaque point de l'échantillonnage :

$$F_{ext}(P_{i,j}) = -\gamma\left(\frac{\partial S(u_i,v_j)}{\partial u}\times\frac{\partial S(u_i,v_j)}{\partial v}\right) \qquad (4)$$

avec $\gamma > 0$ un cœfficient de pondération.

Force interne et préservation des distances : la loi de Hooke est utilisée pour l'élasticité des ressorts et la préservation des distances (cf. équation 5). Elle permet un retour des ressorts à leur position de repos en l'absence de forces externes.

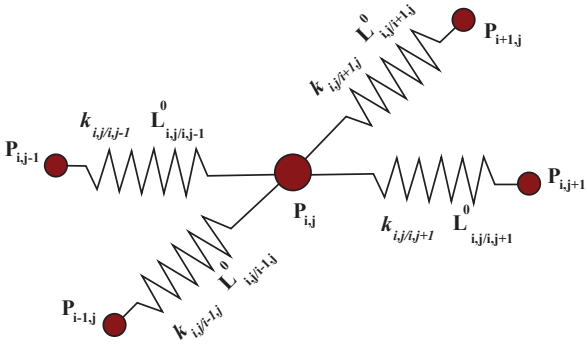
$$F_{stretch}(P_{i,j}) = \sum_{k,l} k_{ij,kl}\left((p_{k,l}-p_{i,j}) - L^0_{ij,kl}\right)\frac{p_{k,l}-p_{i,j}}{\|p_{k,l}-p_{i,j}\|} \qquad (5)$$

où $k_{ij,kl}$ est la constante de raideur reliant les particules $P_{i,j}$ et $P_{k,l}$ si elles sont voisines, et $L^0_{ij,kl}$ est la longueur au repos du ressort.

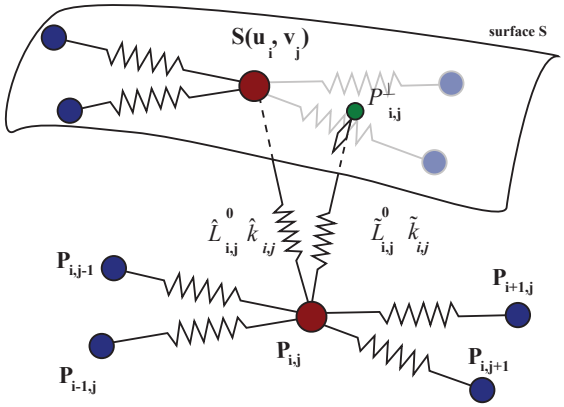
Pour la déformation de vêtements, trois types de ressorts sont souvent rencontrés (modèle de Provot [Pro96]) :

1. les ressorts d'étirement : connectent les masses adjacentes d'après le voisinage de la topologie quadrangulaire ;
2. les ressorts de cisaillement : chaque particule est liée à son voisinage diagonal ; en cas de raideur élevée, l'angle entre deux arêtes adjacentes du réseau quadranglaire reste proche de leur valeur initiale ;
3. les ressorts de flexion : contrôlent la résistance à la courbure, simulent les propriétés différentes des tissus (résistance plus forte pour le cuir que pour la soie) en connectant une masse avec les éléments adjacents à leur voisinage direct.

Le treillis utilisé par la suite répond à la configuration suivante. Quatre ressorts sur le maillage interne correspondent à la configuration des ressorts d'étirement, représentée par la figure 5(a). La notion de distance-offset est reliée au modèle par l'ajout de deux ressorts (cf. figure 5(b)) : le premier lie le point $P_{i,j}$ et le point correspondant à sa localisation paramétrique dans la surface externe $S(u_i, v_j)$; le second est particulier puisqu'il est modifié tout au long de la résolution, par une projection orthogonale $p^{\perp}_{i,j}$ du point situé en $p_{i,j}$ sur la surface S. L'action conjuguée de ces deux ressorts évite une dégénérescence du système interne par l'action des ressorts d'étirement, tout en cherchant à conserver la direction normale à la surface pour créer l'offset. La force interne



(a) Loi de Hooke au point $P_{i,j}$ avec un treillis quadrangulaire



(b) Treillis à six ressorts en chaque particule $P_{i,j}$

Figure 5: *Treillis du système.*

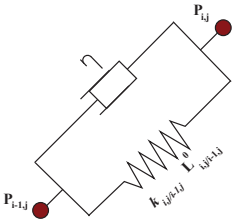


Figure 6: *Modèle de Kelvin-Voigt : amortissement et ressort en parallèle.*

F_{int} est composée des actions de chaque élément du treillis :

$$\begin{aligned} F_{int}(P_{i,j}) &= F_{stretch}(P_{i,j}) \\ &+ \hat{k}_{i,j}\left(\|S(u_i,v_j)-p_{i,j}\|-\hat{L}^0_{i,j}\right)\frac{S(u_i,v_j)-p_{i,j}}{\|S(u_i,v_j)-p_{i,j}\|} \\ &+ \tilde{k}_{i,j}\left(\|p^{\perp}_{i,j}-p_{i,j}\|-\tilde{L}^0_{i,j}\right)\frac{p^{\perp}_{i,j}-p_{i,j}}{\|p^{\perp}_{i,j}-p_{i,j}\|} \end{aligned} \quad (6)$$

D'autres grandeurs sont souvent préservées dans les systèmes masses-ressorts : l'aire définie par trois points, le volume défini par un tétraèdre. Mais nous ne voulons pas que le système revienne au repos, ce qui conduirait au maillage dégénéré initial. Ces deux contraintes ne sont donc pas considérées.

Force d'amortissement : une force d'amortissement basée sur le modèle de Kelvin-Voigt est également ajoutée au système (représenté par la figure 6). Cette force est appliquée dans la direction opposée à la vélocité :

$$F_{damp}(P_{i,j}) = -\eta\dot{p}_{i,j} \quad (7)$$

où η est le cœfficient d'amortissement et $\dot{p}_{i,j}$ la vitesse de la particule $P_{i,j}$.

Force de collision : dans notre modèle, trois natures différentes de collision cohabitent. Les auto-intersections locales (en cas de courbure trop importante), les auto-intersections globales (en cas d'épaisseur localement trop grande ou de manque de contrôle des forces du système), et les intersections entre organes (dues aux erreurs de la discrétisation). Cette dernière catégorie sera traitée par l'application des lois de comportement mécanique. Une pénalité sera appliquée sur les maillages des différents organes pour repousser les zones en contact.

Deux orientations majeures peuvent être mises en évidence pour le traitement des collisions entre objets non-convexes : la subdivision hiérarchique de modèles efficace pour les objets fixes, et la cohérence temporelle pour les objets déformables.

La première utilise les volumes englobants (sphère, OBB, AABB, ...) et les structures en arbre pour tester les contacts [Qui94, GLM96, VDB97, RDL06]. Mais l'application de cette méthode pour les objets déformables nécessite une mise à jour efficace de la représentation hiérarchique.

La seconde prend en considération la cohérence temporelle en utilisant les résultats précédents pour accélérer les calculs. Guy et al. présentent dans [GD04] une approche pour des objets généraux (déformables ou non, convexes ou non). Des régions de collision potentielle sont déterminées et mises à jour avec une méthode aléatoire de Monte-Carlo (cf. [TKZ*04] pour d'autres approches stochastiques). Une liste de paires d'échantillons est créée, chacune contenant un élément appartenant à l'un des maillages testés. Une estimation locale de la distance entre les objets est effectuée. Chaque objet est entouré d'une couche s'adaptant aux distances évaluées pour chaque paire de la liste. La méthode est comparée avec des approches hiérarchiques OBB et AABB et se montre plus efficace en terme de temps de calculs.

4.2.3. Le schéma d'intégration

La force totale du système appliquée sur la particule $P_{i,j}$ à un instant t est finalement :

$$F_{tot}(P_{i,j},t) = F_{int}(P_{i,j},t) + F_{damp}(P_{i,j},t) + F_{ext}(P_{i,j},t) \quad (8)$$

La méthode choisie pour résoudre la seconde loi de Newton est le schéma explicite de Euler. L'étude du modèle ne nécessite pas pour le moment un ordre d'intégration plus élevé comme Runge-Kutta d'ordre 4 ou 5. En outre, la gestion des collisions semble plus appropriée avec ce type de schéma, puisqu'il requiert un pas de temps petit. La mise à jour de la position de la particule $P_{i,j}$ s'obtient de la manière suivante :

$$\begin{cases} \ddot{p}_{i,j}(t+\Delta t) = F_{tot}(P_{i,j},t) \\ \dot{p}_{i,j}(t+\Delta t) = \dot{p}_{i,j}(t) + \Delta t\,\ddot{p}_{i,j}(t+\Delta t) \\ p_{i,j}(t+\Delta t) = p_{i,j}(t) + \Delta t\,\dot{p}_{i,j}(t+\Delta t) \end{cases} \quad (9)$$

avec Δt le pas de temps choisi. Le choix de sa valeur ne doit pas excéder la période naturelle du système [Pro96]. Une augmentation de la constante de raideur doit impliquer une réduction du pas de temps.

L'ensemble des forces internes doit permettre de stabiliser le système en satisfaisant les contraintes du modèle. Le paragraphe 4.2 présente en détails la nature du modèle.

4.3. Résultats

L'entrée du modèle est la discrétisation de la surface paramétrique ajustée. Pour des soucis de visualisation, la discrétisation est grossière par la suite. L'organe étudié dans ce paragraphe est le rectum. Nous rappelons que l'approche paramétrique au paragraphe 3.2 n'a pas réussi à atteindre l'épaisseur de $5,5$ mm sans créer d'auto-intersections.

Nous justifions tout d'abord l'utilisation de chaque partie distincte dans le treillis du système. En premier lieu, seule la force d'étirement $F_{stretch}$ est appliquée en chaque particule,

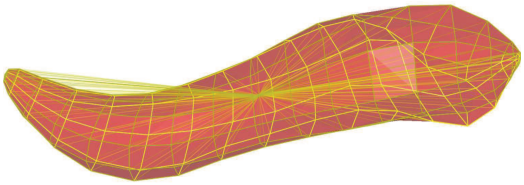
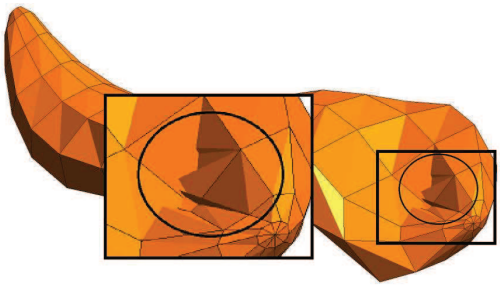


Figure 7: *Dégénérescence du maillage interne avec $F_{stretch}$: les liaisons entre les couches sont représentées*

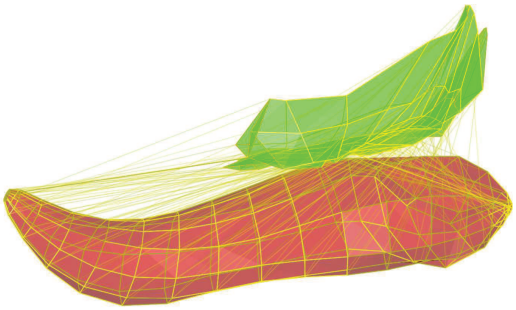
exerçant une influence sur son 1-voisinage. Le résultat est illustré par la figure 7. Les forces liées à la couche externe ne sont pas considérées.

Une dégénérescence du maillage interne est obtenue, puisque seuls les ressorts d'étirement cherchent à retrouver leurs valeurs de repos (fixées faibles pour garantir la tension de la surface). Dans le même esprit, nous appliquons successivement l'action de la force reliant la particule à son point de même localisation paramétrique sur la surface (cf. figure 8(a)), et à sa projection orthogonale (cf. figure 8(b)).

La figure 8(a) présente un problème au niveau des zones



(a) Force de liaison à même localisation paramétrique : les collisions des couches interne et externe sont mises en évidence



(b) Problèmes certains avec seulement la force orthogonale : les liaisons entre les couches sont représentées

Figure 8: *Rôle individuel des deux ressorts supplémentaires ajoutés au treillis avec la force $F_{stretch}$*

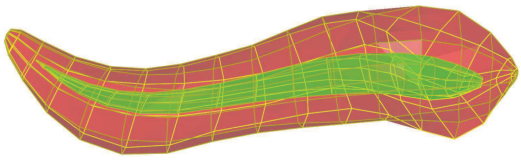


Figure 9: *Application de la force interne totale.*

concaves. L'offset interne ressort de la surface afin de préserver les longueurs des ressorts, mais trouve son équilibre vers l'extérieur du maillage. Il faut ajouter une contrainte pour augmenter la distance entre les deux couches tout en autorisant le déplacement élastique des particules, notamment aux extrémités où les courbures sont les plus importantes. Pour le moment, le problème consiste à créer l'offset d'une surface. Mais dans la réalité physiologique, la membrane interne parcourt évidemment tout l'intérieur. Une épaisseur différente sera calculée pour que l'offset soit présent d'une extrémité à l'autre de la forme.

La figure 8(b) met en évidence l'utilisation du seul ressort orthogonal à la surface. Cependant, à cause de la mise à jour itérative de ces ressorts changeants, la surface présente rapidement d'importants artéfacts. Il faut qu'une connexion fixe persiste de la couche externe à la couche interne.

Finalement, la figure 9 présente l'utilisation simultanée de chaque composante de la force interne.

La dernière étape consiste à calculer les distances exactes entre les couches interne et externe par projection de chaque particule sur la surface paramétrique externe. Pour la particule $P_{i,j}$, ceci revient à résoudre le système :

$$\begin{cases} (p_{i,j} - S(u_i, v_j)).S_u(u_i, v_j) = 0 \\ (p_{i,j} - S(u_i, v_j)).S_v(u_i, v_j) = 0 \end{cases} \quad (10)$$

Les tableaux 2 et 3 ci-dessous présentent un comparatif entre l'approche paramétrique et discrète pour construire la paroi rectale épaisse.

Table 2: *Analyse de l'épaisseur (en mm) de la paroi du rectum : comparaison du modèle B-spline et masses-ressorts.*

	Min.	Max.	Moy.	E.-type
B-spline	0.0004	6.4	3.89	1.96
Ressorts	2.3	11.6	6	1.77

Pour l'approche paramétrique, nous pouvons tout d'abord constater que deux fois moins d'éléments ont atteint l'épaisseur escomptée. En outre, la valeur minimale met en valeur le problème de proximité entre les couches interne et externe dans certaines zones.

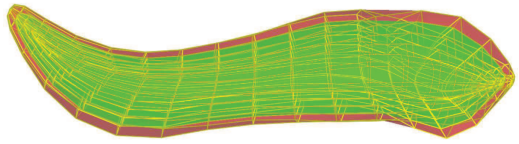
Concernant l'approche discrète, le maximum est bien supérieur à celui de l'approche paramétrique. Ceci est dû au

Table 3: *Proportion des épaisseurs ayant atteint 90% et 100% de la distance-offset avec les modèles B-spline et masses-ressorts.*

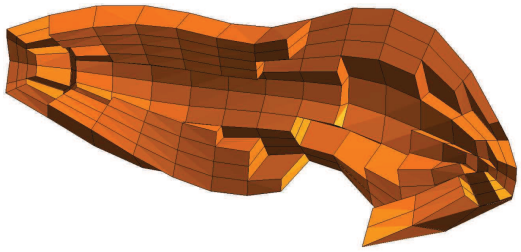
	> 5.5 mm (%)	> 90% de 5.5 mm (%)
B-spline	24.7 %	52 %
Ressorts	55.5 %	73 %

fait que la distance-offset est plus importante dans le système de ressorts pour garantir la stabilité avec la force interne. Une proportion de 25% des éléments reste encore à atteindre. Une heuristique spécifique pour les zones à forte courbure devrait remédier à cela.

Plusieurs couches sont générées à l'issue de la simulation en découpant les liaisons entre la couche externe et interne selon une fonction de répartition définie (cf. figures 10(a) et 10(b)).



(a) Ensemble des mailles dans l'épaisseur



(b) Création des hexaèdres

Figure 10: *Maillage épais d'un rectum.*

4.4. Avantages et inconvénients

Par cette approche, la gestion de l'épaisseur est gérée par l'action conjuguée des forces externes et internes. La gestion des collisions permettra de gérer explicitement les auto-intersections, empêchant les chevauchements dans le maillage.

Grâce à des connaissances supplémentaires acquises sur les images, les experts peuvent déterminer l'épaisseur exacte en quelques points. Ces informations peuvent être facilement intégrées au système en forçant le positionnement de mailles internes fixes à ces localisations : récupération des positions spatiales des points sur la surface externe, calcul de leurs coordonnées (u,v) sur la surface paramétrique et positionnement du point à l'intérieur de l'organe avec l'épaisseur calculée. Ces points n'interviennent plus dans la dynamique du système.

Cependant, le paramétrage empirique des forces est inévitable. Un contrôle du processus est encore nécessaire pour arriver à l'épaisseur exacte recherchée, mais la méthode présente des perspectives intéressantes pour la suite.

5. Conclusions et perspectives

L'approche paramétrique par les moindres carrés présente de nombreux avantages pour des problèmes d'ajustement à des données bruitées. Evidemment, en cas de trop forte courbure de la surface, la création de l'offset ne fonctionne pas. Les méthodes actuelles proposent différents algorithmes pour nettoyer l'offset des repliements générés.

Nous avons choisi cependant une orientation différente. Puisqu'aucune connaissance sur l'épaisseur des organes n'est disponible mis à part des points ponctuels, un offset discret est défini. Un système masses-ressorts est établi pour garantir les contraintes en entrée. Le modèle présente de nombreux avantages, comme un contrôle direct sur les forces et la possibilité de tester directement l'absence de chevauchement dans le maillage. De nombreux travaux présentent en effet des résultats efficaces dans le domaine discret et peuvent être exploités.

Toutefois, le contrôle n'est pas total et le choix des forces peut être étendu. Le test de collision de Monte-Carlo doit être également implémenté. En outre puisque nous travaillons sur un système avec de nombreux paramètres, une analyse de sensibilité serait intéressante à effectuer. Après détermination de plans d'expériences factoriels, une analyse de variance et le calcul d'indices de sensibilité permettraient d'estimer l'impact de chaque paramètre ainsi que leurs interactions.

À l'issue de la boucle de simulation, le maillage hexaédrique peut servir finalement de support à l'application de lois de comportement mécanique des organes.

Remerciements

Ce travail est supporté par l'Agence Nationale de la Recherche, sous la référence « ANR-09-SYSC-008 ».

Références

[BCR*11] BAY T., CHAMBELLAND J.-C., RAFFIN R., DANIEL M., BELLEMARE M.-E. : Geometric modeling of pelvic organs. In *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (Boston, USA, 2011), IEEE, (Ed.), pp. 4329–4332.

[BCR*12] BAY T., CHEN Z.-W., RAFFIN R., DANIEL M., JOLI P., FENG Z.-Q., BELLEMARE M.-E. : Geometric modeling of pelvic organs with thickness. In *3D Image Processing and Applications* (2012).

[CJB03] CHEVALIER L., JAILLET F., BASKURT A. : Segmentation and Superquadric Modeling of 3D Objects. In *WSCG'03* (2003), pp. 232–239.

[DSB99] DESBRUN M., SCHRÖDER P., BARR A. : Interactive animation of structured deformable objects. In *Graphics Interface* (1999), Morgan Kaufmann Publishers Inc., pp. 1–8.

[Far86] FAROUKI R. T. : The approximation of non-degenerate offset surfaces. In *CAGD* (Amsterdam, The Netherlands, The Netherlands, 1986), vol. 3, Elsevier Science Publishers B. V., pp. 15–43.

[GD04] GUY S., DEBUNNE G. : *Monte-Carlo collision detection*. Rapport de recherche RR-5136, INRIA, 2004.

[GLM96] GOTTSCHALK S., LIN M. C., MANOCHA D. : OBB-Tree : a hierarchical structure for rapid interference detection. In *23rd annual conference on CG and interactive techniques* (1996), SIGGRAPH '96, ACM, pp. 171–180.

[Gro98] GROSS M. : Graphics in Medicine : From Visualization to Surgery Simulation. In *Computer Graphics* (1998), vol. 32, pp. 53–56.

[HSW89] HOSCHEK J., SCHNEIDER F.-J., WASSUM P. : Optimal approximate conversion of spline surfaces. In *CAGD* (1989), vol. 6, pp. 293–306.

[KCCP00] KANG Y.-M., CHOI J.-H., CHO H.-G., PARK C.-J. : Fast and stable animation of cloth with an approximated implicit method. In *Computer Graphics International* (2000), IEEE Computer Society, pp. 247–255.

[KN02] KULCZYCKA M. A., NACHMAN L. J. : Qualitative and quantitative comparisons of B-spline offset surface approximation methods. In *CAD* (2002), vol. 34, pp. 19–26.

[KSP02] KUMAR G. V. V. R., SHASTRY K. G., PRAKASH B. G. : Computing non-self-intersecting offsets of NURBS surfaces. In *CAD* (2002), vol. 34, pp. 209 – 228.

[MGC07] MESIT J., GUHA R. K., CHAUDHRY S. : 3D Soft Body Simulation Using Mass-spring System with Internal Pressure Force and Simplified Implicit Integration. In *JCP* (2007), vol. 2, pp. 34–43.

[Mol97] MOLINE J. : Virtual reality for health care : a survey. In *Studies in Health Technology and Informatics* (1997), vol. 44, pp. 3–34.

[PEK08] PEKERMAN D., ELBER G., KIM M.-S. : Self-intersection detection and elimination in freeform curves and surfaces. In *CAD* (Newton, MA, USA, 2008), vol. 40, Butterworth-Heinemann, pp. 150–159.

[Pro96] PROVOT X. : Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface* (1996), pp. 147–154.

[Qui94] QUINLAN S. : Efficient distance computation between non-convex objects. In *International Conference on Robotics and Automation* (1994), pp. 3324–3329.

[RDL06] RAMIREZ F. M., DAY A., LAYCOCK S. : Collision detection for deformable objects using octrees. In *EG UK Theory and Practice of Computer Graphics* (2006), pp. 115–122.

[SEK06] SEONG J.-K., ELBER G., KIM M.-S. : Trimming local and global self-intersections in offset curves/surfaces using distance maps. In *CAD* (2006), vol. 38, Butterworth-Heinemann, pp. 183–193.

[Shi05] SHINYA M. : Theories for mass-spring simulation in computer graphics : Stability, costs and improvements. In *IEICE - Transactions on Information and Systems* (2005), vol. E88-D, Oxford University Press, pp. 767–774.

[SNL04] SUN Y. F., NEE A. Y. C., LEE K. S. : Modifying free-formed NURBS curves and surfaces for offsetting without local self-intersection. In *CAD* (2004), vol. 36, pp. 1161–1169.

[SSS*10] SCHUENKE M., SCHULTE E., SCHUMACHER U., ROSS L., LAMPERTI E., VOLL M., WESKER K. : *Neck and Internal Organs*, vol. 20. Thieme, 2010, ch. 2.

[TH84] TILLER W., HANSON E. : Offsets of two-dimensional profiles. In *IEEE - Computer Graphics and Applications* (Los Alamitos, CA, USA, 1984), vol. 4, IEEE Computer Society, pp. 36–46.

[TKZ*04] TESCHNER M., KIMMERLE S., ZACHMANN G., HEIDELBERGER B., RAGHUPATHI L., FUHRMANN A., CANI M.-P., FAURE F., MAGNETAT-THALMANN N., STRASSER W. : Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)* (2004), pp. 119–139.

[TPF89] TERZOPOULOS D., PLATT J., FLEISCHER K. : From Goop to Glop : Heating and Melting Deformable Models. In *Graphics Interface* (1989), vol. 2, pp. 219–226.

[TW90] TERZOPOULOS D., WATERS K. : Physically-based facial modelling, analysis, and animation. In *The Journal of Visualization and Computer Animation* (1990), vol. 1, John Wiley & Sons, Ltd, pp. 73–80.

[VDB97] VAN DEN BERGEN G. : Efficient collision detection of complex deformable models using AABB trees. In *Journal of Graphics Tools* (1997), vol. 2, A. K. Peters, Ltd., pp. 1–13.

[VMT01] VOLINO P., MAGNENAT-THALMANN N. : Comparing efficiency of integration methods for cloth simulation. In *Computer Graphics International Proceedings* (2001), IEEE Computer Society, pp. 265–274.

[VS02] VASSILEV T., SPANLANG B. : A mass-spring model for real time deformable solids. In *Proceedings of the East-West Vision* (2002), pp. 149–154.

Modélisation à base de règles et de combinaisons de règles topologiques

Gilles GOUATY[1], Houssam HNAIDI[1] et Christian GENTIL[2]

[1]LIRIS
[2]LE2I

Résumé

Nous présentons un formalisme, le modèle BCIFS (Boundary Controlled Iterative Function System), permettant de construire des formes géométriques en décrivant leur processus de subdivision, ainsi que des contraintes de raccord permettant de garantir des propriétés topologiques. Un BCIFS est constitué d'un automate et d'un ensemble de règles d'équivalence sur les chemins de l'automate : un état de l'automate représente une figure, une transition représente un opérateur de subdivision ou d'incidence, et une règle d'équivalence effectue un certain raccord par la mise en commun d'une sous-figure.
Ce modèle est une généralisation des schémas de subdivision classiques. Il permet, par la modulation des opérateurs de subdivision, d'obtenir des formes dont la géométrie peut être lisse ou fractale. Il permet également, par le choix de l'automate et des règles d'équivalence, d'obtenir des formes dont la topologie peut être classique (courbes, surfaces, volumes) ou bien fractale (formes lacunaires, arborescentes,...).
Nous montrons comment il est possible de combiner des BCIFS décrivant une certaine topologie, pour en générer d'autres. Nous présentons l'exemple d'un opérateur qui effectue un certain produit entre deux BCIFS. Il peut générer une surface ou un volume par produits successifs d'une courbe avec elle-même, ou bien générer une topologie fractale par produit de deux topologies fractales.

Mots-clés : Modélisation géométrique, modèles itératifs, topologie fractale, géométrie fractale

1. Introduction

Les modèles géométriques itératifs basés sur le concept de géométrie fractale permettent de générer des objets possédants des topologies complexes. Pour formaliser et contrôler ces topologies TOSAN et GOUATY [TBSG*06, Gou10] ont enrichi le modèle IFS (Iterated Function System) d'une notion de structure B-Rep. Ce modèle exploite les propriétés de treillis associées aux IFS et attracteurs d'IFS [Gen92] pour introduire une notion des cellules topologiques correspondant à des attracteurs de sous-IFS. Ainsi, comme pour les structures B-Rep classiques, les attracteurs sont bordés par des sous-attracteurs à l'aide des relations d'incidence et sont raccordés entre eux à l'aide des relations d'adjacence définies sur des sous attracteurs [GaET11]. Deux volumes fractals peuvent être bordés par des triangles de SIERPINSKI et être reliés entre eux par ces mêmes triangles. Cette structure B-Rep, définies sur les attracteurs, est également exploitée pour décrire la subdivision des attracteurs relative à leur propriété d'auto-similarité. C'est l'une des différences essentielle avec les modèles géométriques classiques pour lesquels, soit la notion de subdivision n'a pas lieu d'être, soit elle est implicite au modèle. A l'aide des BCIFS nous décrivons le système de subdivision topologique et nous explicitons les relations d'incidence et d'adjacence entre les cellules subdivisées. Ces relations d'incidence et d'adjacence garantissent la topologie finale des objets qui peut être complexe (voir figure 1). Ces relations d'incidence et d'adjacence s'écrivent sous forme d'équations à partir des opérateurs de subdivision et des opérateurs d'incidence. La résolution de ces équations induit des contraintes sur les opérateurs de subdivision (structures des matrices représentant les opérateurs de subdivision). Ce sont ces structures partagées par les opérateurs de subdivision qui garantissent la topologie décrite par les relations d'incidence et d'adjacence.

La description de la topologie, structure B-Rep "globale" et de la subdivision topologique, se fait de façon relativement simple et intuitive. Mais celle-ci peut devenir fastidieuse dès que le nombre de subdivisions augmente et surtout que la combinatoire se complexifie du fait de la dimension des cellules. Pour une subdivision classique d'un carreau de surface quadrangulaire, il faut définir 32 relations d'incidence et d'adjacence. Certaines équations sont redondances, mais il est nécessaires d'en définir au moins 12. Pour des volumes la description devient alors conséquente et un minimum de 42 équations sont nécessaires.

Figure 1: *Exemple de structure topologique complexe décrite par BCIFS (*ERIC TOSAN*).*

Dans cet article nous proposons de définir automatiquement une structure topologique à partir de la combinaison de deux autres structures topologiques plus simples. Ainsi nous pouvons définir la subdivision topologique d'une face à partir de la combinaison des subdivisions de deux segments ou combiner la subdivision topologique d'un triangle de SIERPINSKI avec un ensemble de CANTOR (voir figure 2).

Nous proposons deux types d'opérateurs. Pour le premier, l'idée est de formaliser, d'un point de vue topologique, la construction d'un produit tensoriel. Cette opération est définie d'un point de vue géométrique pour les IFS (voir [ZT96]). Le résultat est un IFS dont les opérateurs sont décrits par des matrices obtenues par produits tensoriels des matrices des opérateurs des IFS initiaux. Dans ce cas, la subdivision topologique du résultat est garantie par les structures des matrices. Cependant par cette construction nous ne pouvons accéder qu'à un ensemble limité de solutions. Par exemple en réalisant le produit tensoriel de deux courbes nous ne pouvons définir que des surfaces tensorielles (surfaces de subdivision quadrangulaire). Cependant, il existe d'autres surfaces de subdivision quadrangulaire. Ici, nous cherchons réaliser cette combinaison uniquement d'un point de vue topologiques indépendamment de toute géométrie et du nombre de points de contrôle, ce qui permet d'obtenir des formes géométriques plus générales que des produits tensoriels.

Un autre opérateur de combinaison topologique est défini pour combiner la topologie d'une structure arborescente avec celle d'une autre structure autosimilaire, de telle sorte que la structure arborescente possède des ramifications qui convergent vers la structure autosimilaire.

2. Propriétés générales des BCIFS

La topologie d'un BCIFS est décrite par un automate muni d'une relation d'équivalence sur les chemins de l'automate. On le représente par un quadruplet (Q,Σ,δ,γ) avec :

- Q l'ensemble des états de l'automate,
- Σ un alphabet,
- $\delta : Q \times \Sigma \to Q$ une fonction de transition,
- $\gamma \subset \{(x,(\theta_1,\theta_2)) \mid x \in Q\ ,\ (\theta_1,\theta_2) \in (L^x \times L^x)\ ,\ \delta(x,\theta_1) = \delta(x,\theta_2)\}$ une relation d'équivalence sur les chemins de l'automate, où chaque couple (θ_1,θ_2) représente les membres d'une équation entre ces deux chemins.

Le plongement géométrique d'un BCIFS se fait en associant un espace E^x à chaque état $x \in Q$ et un opérateur géométrique $T_u^x : E^{\delta(x,u)} \to E^x$ à chaque transition, en pratique représenté par une matrice.

2.1. Notations

L'alphabet Σ est partitionné en symboles de subdivision $\Sigma_\div$ et d'incidence Σ_∂ de même que l'ensemble $\{\Sigma^x \mid x \in Q\}$ la restriction des symboles aux transitions sortant de x.

L^x est l'ensemble des chemins de l'automate partant de l'état x et L l'ensemble des chemins partant d'un état quelconque.

Les règles d'équivalence sont partitionnées pour chaque état : $\gamma^x = \{(\theta_1,\theta_2) \mid (x,(\theta_1,\theta_2)) \in \gamma\}$.

γ^x est partitionné en deux types de règles : γ^x_∂ ne contenant que des symboles d'incidence de Σ_∂, et $\gamma^x_\div$ où chaque membre contient un symbole de subivision de $\Sigma_\div$.

On attribue également à chaque état $x \in Q$ un symbole ε^x désignant une transition neutre, $\delta(x,\varepsilon^x) = x$ et on définit $\varepsilon = \bigcup_{x\in Q} \varepsilon^x$.

On généralise δ aux mots de longueur finie quelconque : $\forall (u,\theta) \in \Sigma \times L\ ,\ \delta^*(x,u\theta) = \delta(\delta^*(x,u),\theta)$.

Pour éviter toute ambiguïté sur les symboles des transitions associées aux différents états, on notera systématiquement les symboles en indice de $\div$ pour les symboles de subdivision et ∂ pour les symboles d'incidence, et on notera en exposant l'état de départ de la transition : $\div_{u^x} \in \Sigma^x_\div$ et $\partial_{i^x} \in \Sigma^x_\partial$.

3. Construction d'un produit de 2 BCIFS

Nous montrons la construction d'un BCIFS (Q,Σ,δ,γ) résultation de l'opération notée $\bullet$ entre deux BCIFS $(Q_1,\Sigma_1,\delta_1,\gamma_1) \bullet (Q_2,\Sigma_2,\delta_2,\gamma_2)$.

Nous donnons dans un premier la description générale de

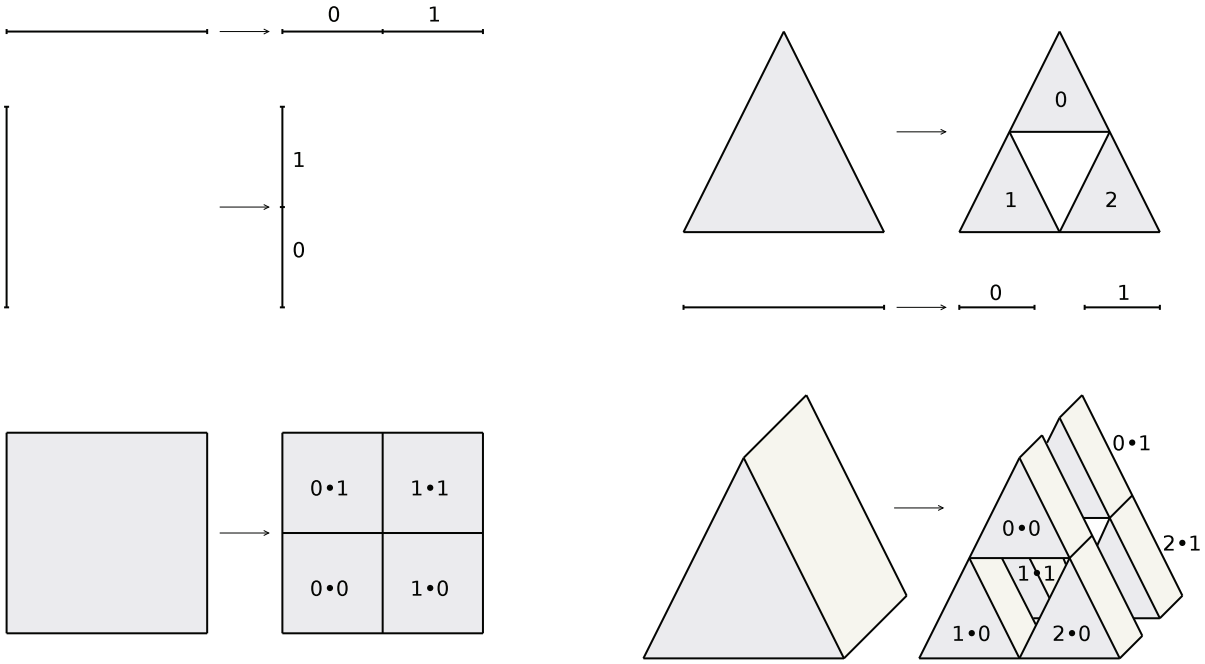


Figure 2: *Construction des opérateurs de subdivision d'un produit de deux BCIFS.*

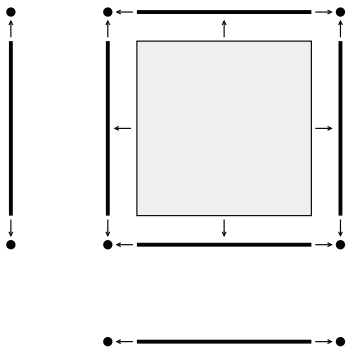


Figure 3: *Construction des opérateurs d'incidence d'un produit de deux BCIFS.*

la méthode. Nous en illustrerons à travers les exemples qui suivent les différents points.

3.1. Description générale de la méthode

3.1.1. États

À chaque couple d'états (x,y) de $Q_1 \times Q_2$ passés en paramètres est associé un état de Q dans l'automate résultant. On a $|Q| = |Q_1| \times |Q_2|$. On notera $\bullet$ l'opérateur effectuant cette association :

$$Q = \{x \bullet y \mid (x,y) \in Q_1 \times Q_2\} \tag{1}$$

3.1.2. Symboles

On notera également $\bullet$ l'opérateur qui associe les symboles de Σ_1 et Σ_2 à ceux de Σ.

À chaque couple de subdivision (u,v) de $\Sigma^x_\div \times \Sigma^y_\div$ est associé une subdivision de $\Sigma^{x\bullet y}_\div$. On a $|\Sigma^{x\bullet y}_\div| = |\Sigma^x_\div| \times |\Sigma^y_\div|$.

$$\begin{array}{l} \forall (x,y) \in Q_1 \times Q_2 \ , \\ \Sigma^{x\bullet y}_\div = \{ \div_{u^x \bullet v^y} \mid (\div_{u^x}, \div_{v^y}) \in \Sigma^x_\div \times \Sigma^y_\div \} \ , \end{array} \tag{2}$$

À chaque symbole d'incidence de $\Sigma^x_\partial \cup \Sigma^y_\partial$ est associé un symbole d'incidence de $\Sigma^{x\bullet y}_\partial$. On a $|\Sigma^{x\bullet y}_\partial| = |\Sigma^x_\partial| + |\Sigma^y_\partial|$. On utilise l'opérateur $\bullet$ pour combiner un symbole d'incidence

avec le symbole neutre ε^x ou ε^y, à gauche ou à droite.

$$\begin{array}{l}\forall(x,y)\in Q_1\times Q_2\ ,\\ \Sigma_\partial^{x\bullet y}=\{\partial_{i^x\bullet\varepsilon^y}\mid\partial_{i^x}\in\Sigma_\partial^x\}\cup\{\partial_{\varepsilon^x\bullet j^y}\mid\partial_{j^y}\in\Sigma_\partial^y\}\ ,\end{array}\tag{3}$$

On définit également le symbole $\varepsilon^{x\bullet y}$ en fonction des deux symboles ε^x et ε^y.

$$\forall(x,y)\in Q_1\times Q_2\ ,\ \varepsilon^{x\bullet y}=\varepsilon^x\bullet\varepsilon^y\tag{4}$$

3.1.3. Transitions

L'état d'arrivée d'une combinaison de subdivisions par l'opérateur $\bullet$ sur les symboles est égal à la combinaison des états d'arrivée par l'opérateur $\bullet$ sur les états. Parcourir une transition $\div_{u^x\bullet v^y}$ dans Q revient à parcourir simultanément la transition $\div_{u^x}$ dans Q_1 et la transition $\div_{v^y}$ dans Q_2.

$$\begin{array}{l}\forall(x,y)\in Q_1\times Q_2\ ,\ \forall(\div_{u^x},\div_{v^y})\in\Sigma_\div^x\times\Sigma_\div^y\ ,\\ \delta(x\bullet y\ ,\ \div_{u^x\bullet v^y})=\delta_1(x\ ,\ \div_{u^x})\bullet\delta_2(y\ ,\ \div_{v^y})\end{array}\tag{5}$$

Pour les symboles d'incidence, on parcourt seulement la transition associée dans Q_1 ou Q_2.

$$\begin{array}{l}\forall(x,y)\in Q_1\times Q_2\ ,\ \forall\partial_i\in\Sigma_\partial^x\ ,\\ \delta(x\bullet y\ ,\ \partial_{i^x\bullet\varepsilon^y})=\delta_1(x\ ,\ \partial_{i^x})\bullet y\end{array}\tag{6}$$

$$\begin{array}{l}\forall(x,y)\in Q_1\times Q_2\ ,\ \forall\partial_j\in\Sigma_\partial^y\ ,\\ \delta(x\bullet y\ ,\ \partial_{\varepsilon^x\bullet j^y})=x\bullet\delta_2(y\ ,\ \partial_{j^y})\end{array}\tag{7}$$

3.1.4. Relation d'équivalence

On utilise ici l'opérateur $\circ$ entre un symbole et un chemin (défini plus loin).

On génère la relation d'équivalence sur les subdivisions par une certaine combinaison entre la relation d'équivalence sur les subdivisions d'un état en paramètre avec les symboles de subdivision de l'autre état. On a $|\gamma_\div^{x\bullet y}|=|\gamma_\div^x|\times|\Sigma_\div^y|+|\gamma_\div^y|\times|\Sigma_\div^x|$.

$$\begin{array}{l}\forall(x,y)\in Q_1\times Q_2\ ,\\ \gamma_\div^{x\bullet y}\ =\ \{(\theta_1\circ\div_{u^y}\ ,\ \theta_2\circ\div_{u^y})\mid\\ \qquad(\theta_1,\theta_2)\in\gamma_\div^x\ ,\ \div_{u^y}\in\Sigma_\div^y\}\cup\\ \qquad\{(\div_{u^x}\circ\theta_1\ ,\ \div_{u^x}\circ\theta_2)\mid\\ \qquad(\theta_1,\theta_2)\in\gamma_\div^y\ ,\ \div_{u^x}\in\Sigma_\div^x\}\end{array}\tag{8}$$

On construit la relation d'équivalence sur les opérateurs d'incidence à partir de celles des deux états en paramètres, de manière indépendante l'une de l'autre, et on combine entre eux leurs opérateurs d'incidence. On a $|\gamma_\partial^{x\bullet y}|=|\gamma_\partial^x|+|\gamma_\partial^y|+|\Sigma_\partial^x|\times|\Sigma_\partial^y|$.

$$\begin{array}{l}\forall(x,y)\in Q_1\times Q_2\ ,\\ \gamma_\partial^{x\bullet y}=\ \{(\theta_1\circ\varepsilon^y\ ,\ \theta_2\circ\varepsilon^y)\mid(\theta_1,\theta_2)\in\gamma_\partial^x\}\cup\\ \qquad\{(\varepsilon^x\circ\theta_1\ ,\ \varepsilon^x\circ\theta_2)\mid(\theta_1,\theta_2)\in\gamma_\partial^y\}\cup\\ \qquad\{(\partial_{i^x\bullet\varepsilon^y}\ \partial_{\varepsilon^{\delta_1(x,i^x)}\bullet j^y}\ ,\ \partial_{\varepsilon^x\bullet j^y}\ \partial_{i^x\bullet\varepsilon^{\delta_2(y,j^y)}})\mid\\ \qquad(\partial_{i^x},\partial_{j^y})\in\Sigma_\partial^x\times\Sigma_\partial^y\}\end{array}\tag{9}$$

L'opérateur $\circ$ entre un symbole et un chemin construit un chemin de même longueur constitué de symboles composés.

$$\begin{array}{l}\forall(\div_{u^x},\div_{v^y},\theta)\in\Sigma_\div^x\times\Sigma_\div^y\times L\ ,\\ \div_{u^x}\circ(\div_{v^y}\theta)\ =\ (\div_{u^x\bullet v^y})\,(\varepsilon^{\delta(x,\div_{u^x})}\circ\theta)\ ,\\ (\div_{u^x}\theta)\circ\div_{v^y}\ =\ (\div_{u^x\bullet v^y})\,(\theta\circ\varepsilon^{\delta(y,\div_{v^y})})\end{array}$$

$$\begin{array}{l}\forall(\div_{u^x},\partial_i,\theta)\in\Sigma_\div^x\times\Sigma_\partial\times L\ ,\\ \div_{u^x}\circ(\partial_i\theta)\ =\ (\partial_{\varepsilon^x\bullet i})\,(\div_{u^x}\circ\theta)\ ,\\ (\partial_i\theta)\circ\div_{u^x}\ =\ (\partial_{i\bullet\varepsilon^x})\,(\theta\circ\div_{u^x})\ ,\\ \partial_i\circ(\div_{u^x}\theta)\ =\ (\partial_{i\bullet\varepsilon^x})\,(\partial_i\circ\theta)\ ,\\ (\div_{u^x}\theta)\circ\partial_i\ =\ (\partial_{\varepsilon^x\bullet i})\,(\theta\circ\partial_i)\end{array}$$

$$\begin{array}{l}\forall(\partial_i,\partial_j,\theta)\in\Sigma_\partial\times\Sigma_\partial\times L\ ,\\ \partial_i\circ(\partial_j\theta)\ =\ (\partial_{i\bullet j})\,(\partial_i\circ\theta)\ ,\\ (\partial_i\theta)\circ\partial_j\ =\ (\partial_{i\bullet j})\,(\theta\circ\partial_i)\end{array}$$

$$\begin{array}{l}\forall(\varepsilon^x,\partial_i,\theta)\in\varepsilon\times\Sigma_\partial\times L\ ,\\ \varepsilon^x\circ(\partial_i\theta)\ =\ (\partial_{\varepsilon^x\bullet i})\,(\varepsilon^x\circ\theta)\ ,\\ (\partial_i\theta)\circ\varepsilon^x\ =\ (\partial_{\varepsilon^x\bullet i})\,(\theta\circ\varepsilon^x)\end{array}$$

3.2. Exemple : génération d'une surface à partir de 2 courbes

Nous montrons sur ce premier exemple le résultat de l'opérateur $\bullet$ entre deux courbes, qui génère ainsi la description d'une surface.

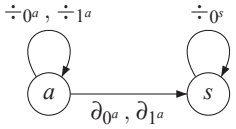
Nous donnons dans un premier temps la description d'une courbe à deux subdivisions dans le formalisme BCIFS.

3.2.1. Description BCIFS d'une courbe

On décrit une courbe a à 2 subdivisions et bordée par 2 sommets s :

$$\begin{array}{lcllcl}\delta(a,\div_{0^a})&=&a&\delta(a,\partial_{0^a})&=&s\\ \delta(a,\div_{1^a})&=&a&\delta(a,\partial_{1^a})&=&s\\ \\ \delta(s,\div_{0^s})&=&s\end{array}$$

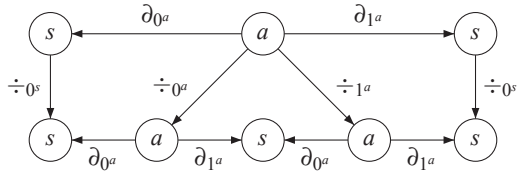
Cette description correspond à l'automate suivant :



On définit la relation d'équivalence suivante :

$$\gamma^a=\gamma^a_\div=\left\{\begin{array}{l}(\ \div_{0^a}\partial_{0^a}\ ,\ \partial_{0^a}\div_{0^s}\ )\\ (\ \div_{1^a}\partial_{1^a}\ ,\ \partial_{1^a}\div_{0^s}\ )\\ (\ \div_{0^a}\partial_{1^a}\ ,\ \div_{1^a}\partial_{0^a}\ )\end{array}\right\}$$

On visualise ces équations sur le graphe quotient :



On fait de même pour b et t.

$$\begin{array}{lll} \delta(b,\div 0^b) = b & & \delta(b,\partial_{0^b}) = t \\ \delta(b,\div 1^b) = b & & \delta(b,\partial_{1^b}) = t \end{array}$$

$$\delta(t,\div 0^t) = t \qquad \gamma^b = \gamma^a$$

3.2.2. États

On effectue le produit $a \bullet b$.

On construit un automate dont les états sont des combinaisons des états des deux automates associés à a et b.

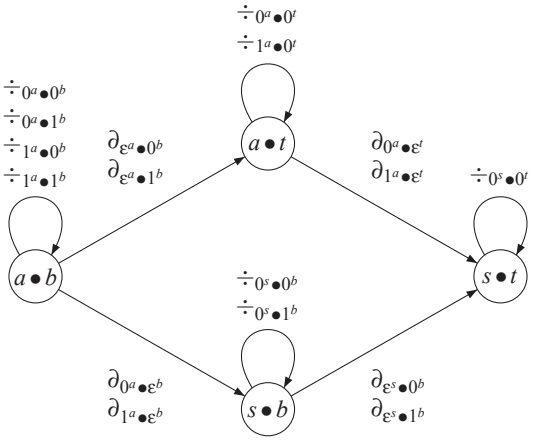
$$Q = \{a\bullet b, a\bullet t, s\bullet b, s\bullet t\}$$

L'état $a \bullet b$ représente une surface, les états $a \bullet t$ et $s \bullet b$ des courbes, et l'état $s \bullet t$ un sommet.

3.2.3. Transitions

On construit les transitions pour chaque état du BCIFS produit d'après les équations 5, 6 et 7.

$$\begin{array}{lll}
\delta(a\bullet b,\div 0^a\bullet 0^b) = a\bullet b & & \delta(a\bullet b,\partial_{\varepsilon^a\bullet 0^b}) = a\bullet t \\
\delta(a\bullet b,\div 0^a\bullet 1^b) = a\bullet b & & \delta(a\bullet b,\partial_{\varepsilon^a\bullet 1^b}) = a\bullet t \\
\delta(a\bullet b,\div 1^a\bullet 0^b) = a\bullet b & & \delta(a\bullet b,\partial_{0^a\bullet\varepsilon^b}) = s\bullet b \\
\delta(a\bullet b,\div 1^a\bullet 1^b) = a\bullet b & & \delta(a\bullet b,\partial_{1^a\bullet\varepsilon^b}) = s\bullet b \\
\\
\delta(a\bullet t,\div 0^a\bullet 0^t) = a\bullet t & & \delta(a\bullet t,\partial_{0^a\bullet\varepsilon^t}) = s\bullet t \\
\delta(a\bullet t,\div 1^a\bullet 0^t) = a\bullet t & & \delta(a\bullet t,\partial_{1^a\bullet\varepsilon^t}) = s\bullet t \\
\\
\delta(s\bullet b,\div 0^s\bullet 0^b) = s\bullet b & & \delta(s\bullet b,\partial_{\varepsilon^s\bullet 0^b}) = s\bullet t \\
\delta(s\bullet b,\div 0^s\bullet 1^b) = s\bullet b & & \delta(s\bullet b,\partial_{\varepsilon^s\bullet 1^b}) = s\bullet t \\
\\
\delta(s\bullet t,\div 0^s\bullet 0^t) = s\bullet t
\end{array}$$



$$\varepsilon^{a\bullet b} = \varepsilon^a\bullet\varepsilon^b \ , \ \varepsilon^{a\bullet t} = \varepsilon^a\bullet\varepsilon^t \ , \ \varepsilon^{s\bullet b} = \varepsilon^s\bullet\varepsilon^b \ , \ \varepsilon^{s\bullet t} = \varepsilon^s\bullet\varepsilon^t$$

3.2.4. Relation d'équivalence sur les opérateurs d'incidence

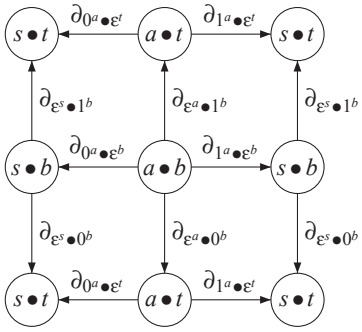
On construit les éléments de $\gamma_\partial^{a\bullet b}$ d'après l'équation 9. Seul le troisième terme est utilisé ici car $\gamma_\partial^a = \gamma_\partial^b = \emptyset$. On combine les éléments de Σ_∂^a et Σ_∂^b.

	∂_{0^b}
∂_{0^a}	$(\ \partial_{0^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 0^b}\ ,\ \partial_{\varepsilon^a\bullet 0^b}\ \partial_{0^a\bullet\varepsilon^t}\ )$
∂_{1^a}	$(\ \partial_{1^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 0^b}\ ,\ \partial_{\varepsilon^a\bullet 0^b}\ \partial_{1^a\bullet\varepsilon^t}\ )$

	∂_{1^b}
∂_{0^a}	$(\ \partial_{0^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 1^b}\ ,\ \partial_{\varepsilon^a\bullet 1^b}\ \partial_{0^a\bullet\varepsilon^t}\ )$
∂_{1^a}	$(\ \partial_{1^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 1^b}\ ,\ \partial_{\varepsilon^a\bullet 1^b}\ \partial_{1^a\bullet\varepsilon^t}\ )$

$$\gamma_\partial^{a\bullet b} = \left\{ \begin{array}{l} (\ \partial_{0^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 0^b}\ ,\ \partial_{\varepsilon^a\bullet 0^b}\ \partial_{0^a\bullet\varepsilon^t}\ ) \\ (\ \partial_{0^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 1^b}\ ,\ \partial_{\varepsilon^a\bullet 1^b}\ \partial_{0^a\bullet\varepsilon^t}\ ) \\ (\ \partial_{1^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 0^b}\ ,\ \partial_{\varepsilon^a\bullet 0^b}\ \partial_{1^a\bullet\varepsilon^t}\ ) \\ (\ \partial_{1^a\bullet\varepsilon^b}\ \partial_{\varepsilon^s\bullet 1^b}\ ,\ \partial_{\varepsilon^a\bullet 1^b}\ \partial_{1^a\bullet\varepsilon^t}\ ) \end{array} \right\}$$

On obtient le graphe quotient ci-dessous, représentant les mises en commun des sommets de la face telles que sur la figure 3.



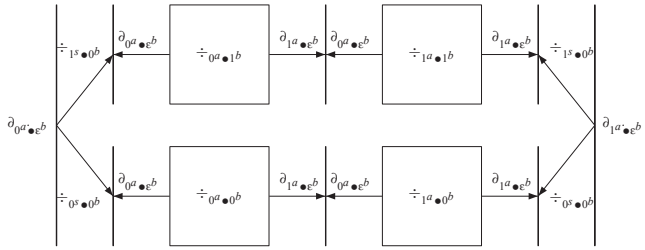
3.2.5. Relation d'équivalence sur les opérateurs de subdivision

Une première partie des éléments de $\gamma_\div^{a\bullet b}$ est obtenue en combinant $\gamma_\div^a$ avec $\Sigma_\div^b$ d'après l'équation 9 :

	$\div 0^b$
$(\ \div 0^a\ \partial_{0^a}\ ,\ \partial_{0^a}\ \div 0^a\ )$	$(\ \div 0^a\bullet 0^b\ \partial_{0^a\bullet\varepsilon^b}\ ,\ \partial_{0^a\bullet\varepsilon^b}\ \div 0^s\bullet 0^b\ )$
$(\ \div 1^a\ \partial_{1^a}\ ,\ \partial_{1^a}\ \div 0^a\ )$	$(\ \div 1^a\bullet 0^b\ \partial_{1^a\bullet\varepsilon^b}\ ,\ \partial_{1^a\bullet\varepsilon^b}\ \div 0^s\bullet 0^b\ )$
$(\ \div 0^a\ \partial_{1^a}\ ,\ \div 1^a\ \partial_{0^a}\ )$	$(\ \div 0^a\bullet 0^b\ \partial_{1^a\bullet\varepsilon^b}\ ,\ \div 1^a\bullet 0^b\ \partial_{0^a\bullet\varepsilon^b}\ )$

	$\div 1^b$
$(\ \div 0^a\ \partial_{0^a}\ ,\ \partial_{0^a}\ \div 0^a\ )$	$(\ \div 0^a\bullet 1^b\ \partial_{0^a\bullet\varepsilon^b}\ ,\ \partial_{0^a\bullet\varepsilon^b}\ \div 0^s\bullet 1^b\ )$
$(\ \div 1^a\ \partial_{1^a}\ ,\ \partial_{1^a}\ \div 0^a\ )$	$(\ \div 1^a\bullet 1^b\ \partial_{1^a\bullet\varepsilon^b}\ ,\ \partial_{1^a\bullet\varepsilon^b}\ \div 0^s\bullet 1^b\ )$
$(\ \div 0^a\ \partial_{1^a}\ ,\ \div 1^a\ \partial_{0^a}\ )$	$(\ \div 0^a\bullet 1^b\ \partial_{1^a\bullet\varepsilon^b}\ ,\ \div 1^a\bullet 1^b\ \partial_{0^a\bullet\varepsilon^b}\ )$

Les six équations précédentes sont représentées sur ce graphe.



On genère les autres éléments de $\gamma_{\div}^{a\bullet b}$ en prenant $\Sigma_{\div}^{a}$ et $\gamma_{\div}^{b}$:

	$\div 0^a$
$(\div 0^b \;\; \partial_{0^b} \;,\; \partial_{0^b} \;\; \div 0^t)$	$(\div 0^a\bullet 0^b \;\; \partial_{\varepsilon^a\bullet 0^b} \;,\; \partial_{\varepsilon^a\bullet 0^b} \;\; \div 0^a\bullet 0^t)$
$(\div 1^b \;\; \partial_{1^b} \;,\; \partial_{1^b} \;\; \div 0^t)$	$(\div 0^a\bullet 1^b \;\; \partial_{\varepsilon^a\bullet 1^b} \;,\; \partial_{\varepsilon^a\bullet 1^a} \;\; \div 0^a\bullet 0^t)$
$(\div 0^b \;\; \partial_{1^b} \;,\; \div 1^b \;\; \partial_{0^b})$	$(\div 0^a\bullet 0^b \;\; \partial_{\varepsilon^a\bullet 1^b} \;,\; \div 0^a\bullet 1^b \;\; \partial_{\varepsilon^a\bullet 0^b})$

	$\div 1^a$
$(\div 0^b \;\; \partial_{0^b} \;,\; \partial_{0^b} \;\; \div 0^t)$	$(\div 1^a\bullet 0^b \;\; \partial_{\varepsilon^a\bullet 0^b} \;,\; \partial_{\varepsilon^a\bullet 0^b} \;\; \div 1^a\bullet 0^t)$
$(\div 1^b \;\; \partial_{1^b} \;,\; \partial_{1^b} \;\; \div 0^t)$	$(\div 1^a\bullet 1^b \;\; \partial_{\varepsilon^a\bullet 1^b} \;,\; \partial_{\varepsilon^a\bullet 1^b} \;\; \div 1^a\bullet 0^t)$
$(\div 0^b \;\; \partial_{1^b} \;,\; \div 1^b \;\; \partial_{0^b})$	$(\div 1^a\bullet 0^b \;\; \partial_{\varepsilon^a\bullet 1^b} \;,\; \div 1^a\bullet 1^b \;\; \partial_{\varepsilon^a\bullet 0^t})$

On génère également $\gamma_{\div}^{a\bullet t}$ en combinant $\gamma_{\div}^{a}$ et $\Sigma_{\div}^{t}$:

	$\div 0^t$
$(\div 0^a \;\; \partial_{0^a} \;,\; \partial_{0^a} \;\; \div 0^s)$	$(\div 0^a\bullet 0^t \;\; \partial_{0^a\bullet\varepsilon^t} \;,\; \partial_{0^a\bullet\varepsilon^t} \;\; \div 0^s\bullet 0^t)$
$(\div 1^a \;\; \partial_{1^a} \;,\; \partial_{1^a} \;\; \div 0^s)$	$(\div 1^a\bullet 0^t \;\; \partial_{1^a\bullet\varepsilon^t} \;,\; \partial_{1^a\bullet\varepsilon^t} \;\; \div 0^s\bullet 0^t)$
$(\div 0^a \;\; \partial_{1^a} \;,\; \div 1^a \;\; \partial_{0^a})$	$(\div 0^a\bullet 0^t \;\; \partial_{1^a\bullet\varepsilon^t} \;,\; \div 1^a\bullet 0^t \;\; \partial_{0^a\bullet\varepsilon^t})$

On fait de même pour $\gamma_{\div}^{s\bullet b}$ avec $\Sigma_{\div}^{s}$ et $\gamma_{\div}^{b}$:

	$\div 0^s$
$(\div 0^b \;\; \partial_{0^b} \;,\; \partial_{0^b} \;\; \div 0^t)$	$(\div 0^s\bullet 0^b \;\; \partial_{\varepsilon^s\bullet 0^b} \;,\; \partial_{\varepsilon^s\bullet 0^b} \;\; \div 0^s\bullet 0^t)$
$(\div 1^a \;\; \partial_{1^b} \;,\; \partial_{1^b} \;\; \div 0^t)$	$(\div 0^s\bullet 1^b \;\; \partial_{\varepsilon^s\bullet 1^b} \;,\; \partial_{\varepsilon^s\bullet 1^b} \;\; \div 0^s\bullet 0^t)$
$(\div 0^b \;\; \partial_{1^b} \;,\; \div 1^b \;\; \partial_{0^b})$	$(\div 0^s\bullet 0^b \;\; \partial_{\varepsilon^s\bullet 1^b} \;,\; \div 0^s\bullet 1^b \;\; \partial_{\varepsilon^s\bullet 0^b})$

3.3. Exemple : génération d'un volume à partir d'une surface et d'une courbe

On effectue le produit $a\bullet b\bullet a$, en supposant que l'opérateur $\bullet$ est associatif. On réutilise les résultats de $a\bullet b$ et on calcule le produit $(a\bullet b)\bullet a$.

On développe ici la génération de l'automate et de la relation d'équivalence sur les opérateurs d'incidence, dont un aspect de la méthode n'était pas apparu sur l'exemple précédent. On ne développera pas la génération de la relation d'équivalence sur les opérateurs de subdivision.

L'automate est représenté sur la figure 4.

La génération de la relation d'équivalence sur les opérateurs d'incidence $\gamma_{\partial}^{a\bullet b\bullet a}$ utilise différents termes de l'équation 9.

Comme dans l'exemple précédent, on effectue des opérations entre $\Sigma_{\partial}^{a\bullet b}$ et Σ_{∂}^{a} :

	∂_{0^a}
$\partial_{\varepsilon^a\bullet 0^b}^{a\bullet b}$	$(\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^s})$
$\partial_{\varepsilon^a\bullet 1^a}^{a\bullet b}$	$(\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^s})$
$\partial_{0^a\bullet\varepsilon^a}^{a\bullet b}$	$(\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^s})$
$\partial_{1^a\bullet\varepsilon^a}^{a\bullet b}$	$(\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^s})$

	∂_{1^a}
$\partial_{\varepsilon^a\bullet 0^b}^{a\bullet b}$	$(\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^s})$
$\partial_{\varepsilon^a\bullet 1^a}^{a\bullet b}$	$(\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^s})$
$\partial_{0^a\bullet\varepsilon^a}^{a\bullet b}$	$(\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^s})$
$\partial_{1^a\bullet\varepsilon^a}^{a\bullet b}$	$(\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^s})$

On établit également des équivalences sur $\gamma_{\partial}^{a\bullet b\bullet a}$ à partir de celles de $\gamma_{\partial}^{a\bullet b}$:

	ε^a
$(\partial_{0^a\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 0^a}, \partial_{\varepsilon^a\bullet 0^a}\partial_{0^a\bullet\varepsilon^t})$	$(\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 0^b\bullet\varepsilon^a}, \partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a}\partial_{0^a\bullet\varepsilon^t\bullet\varepsilon^a})$
$(\partial_{0^a\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 1^a}, \partial_{\varepsilon^a\bullet 1^a}\partial_{0^a\bullet\varepsilon^t})$	$(\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 1^b\bullet\varepsilon^a}, \partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a}\partial_{0^a\bullet\varepsilon^t\bullet\varepsilon^a})$
$(\partial_{1^a\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 0^a}, \partial_{\varepsilon^a\bullet 0^a}\partial_{1^a\bullet\varepsilon^t})$	$(\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 0^b\bullet\varepsilon^a}, \partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a}\partial_{1^a\bullet\varepsilon^t\bullet\varepsilon^a})$
$(\partial_{1^a\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 1^a}, \partial_{\varepsilon^a\bullet 1^a}\partial_{1^a\bullet\varepsilon^t})$	$(\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a}\partial_{\varepsilon^s\bullet 1^b\bullet\varepsilon^a}, \partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a}\partial_{1^a\bullet\varepsilon^t\bullet\varepsilon^a})$

La relation d'équivalence $\gamma_{\partial}^{a\bullet b\bullet a}$ contient les douze relation d'adjacence mettant en commun une arête entre deux faces.

$$\gamma_{\partial}^{a\bullet b\bullet a} = \left\{ \begin{array}{l} (\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^s}) \\ (\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^s}) \\ (\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^s}) \\ (\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a} \;\partial_{\varepsilon^a\bullet\varepsilon^t\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^s}) \\ (\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^s}) \\ (\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^s}) \\ (\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 0^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 0^a} \;\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^s}) \\ (\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet\varepsilon^b\bullet 1^a} \;,\; \partial_{\varepsilon^a\bullet\varepsilon^b\bullet 1^a} \;\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^s}) \\ (\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet 0^b\bullet\varepsilon^a} \;,\; \partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a} \;\partial_{0^a\bullet\varepsilon^t\bullet\varepsilon^a}) \\ (\partial_{0^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet 1^b\bullet\varepsilon^a} \;,\; \partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a} \;\partial_{0^a\bullet\varepsilon^t\bullet\varepsilon^a}) \\ (\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet 0^b\bullet\varepsilon^a} \;,\; \partial_{\varepsilon^a\bullet 0^b\bullet\varepsilon^a} \;\partial_{1^a\bullet\varepsilon^t\bullet\varepsilon^a}) \\ (\partial_{1^a\bullet\varepsilon^b\bullet\varepsilon^a} \;\partial_{\varepsilon^s\bullet 1^b\bullet\varepsilon^a} \;,\; \partial_{\varepsilon^a\bullet 1^b\bullet\varepsilon^a} \;\partial_{1^a\bullet\varepsilon^t\bullet\varepsilon^a}) \end{array} \right\}$$

4. Génération de descriptions d'arbres

On cherche à construire un arbre reliant un sommet s décrit par le BCIFS $(Q_1 = \{s\}, \Sigma_1 = \{0^s\}, \delta_1 = \{((s,0^s),s)\}, \gamma_1 = \{\})$ et une figure b d'un BCIFS $(Q_2, \Sigma_2, \delta_2, \gamma_2)$.

La méthode se base sur quelques principes simples, que l'on exprimera plus loin dans notre formalisme de manière générale afin de générer automatiquement des descriptions BCIFS.

– On définit sur l'arbre un bord pour la racine de type sommet, et un bord pour les feuilles de type b

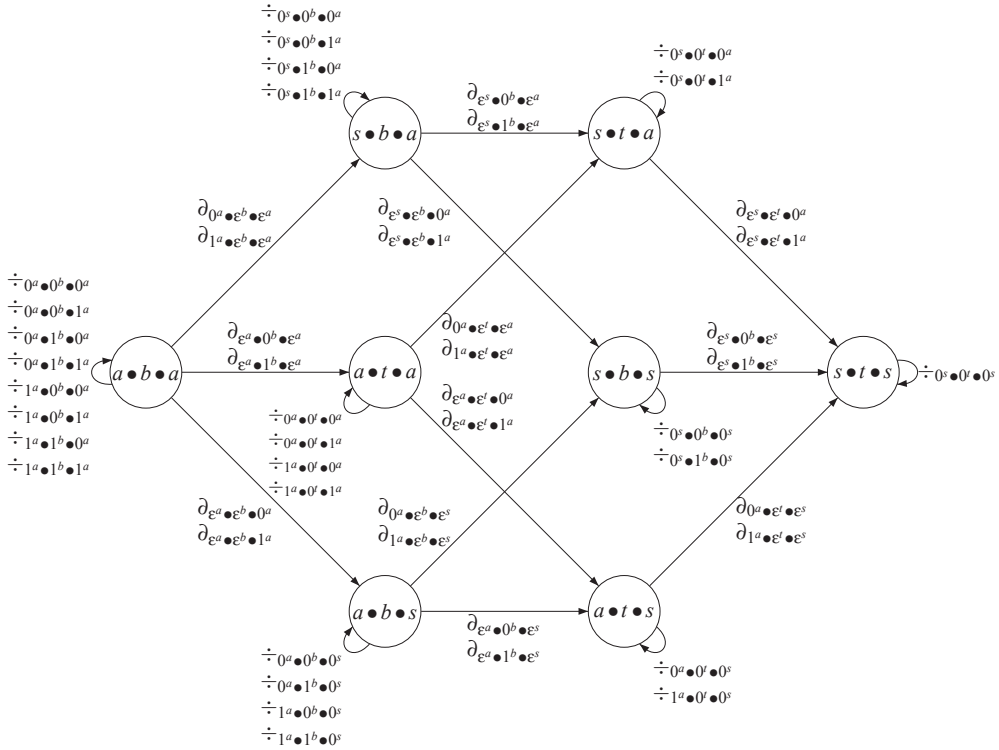


Figure 4: *Automate décrivant le cube.*

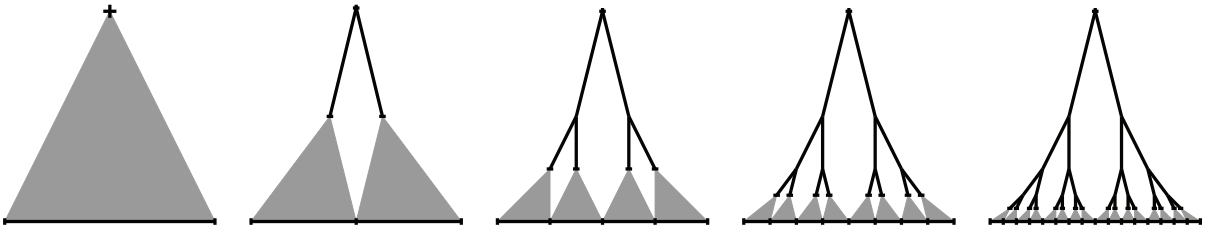


Figure 5: *Premières étapes de subdivision d'un arbre bordé par une courbe.*

- Pour chaque subdivision de b, l'arbre se subdivise en une branche reliant la racine et un sommet intermédiaire, et un arbre reliant le sommet intermédiaire et la subdivision de b.
- Si b se subdivise en un autre type que b, on définit également un autre type d'arbre. On définit autant de types d'arbres que d'états de Q_2 accessibles depuis b par un mot de subdivision $L^b_{\div}$, soit l'ensemble image de $L^b_{\div}$ par la fonction $x \to \delta_2^*(b,x)$, que l'on notera $\delta_2^*(b,L^b_{\div})$.

4.1. Formalisation de la méthode

4.2. États

On notera $\triangleleft$ l'opérateur sur les états qui associe chaque couple d'états de $Q_1 = \{s\}$ et $L^b_{\div}$ à un type d'arbre. On crée également un état a décrivant une branche de l'arbre :

$$Q = \{s \triangleleft x \mid x \in \delta_2^*(b,L^b_{\div})\} \cup Q_1 \cup Q_2 \cup \{a\} \qquad (10)$$

4.3. Symboles

On notera $\partial_{0^{s\triangleleft x}}$ la racine de l'arbre et $\partial_{1^{s\triangleleft x}}$ les feuilles.

$$\forall x \in \delta_2^*(b,L^b_{\div}) \ , \ \Sigma_{\partial}^{s\triangleleft x} = \{\partial_{0^{s\triangleleft x}}, \partial_{1^{s\triangleleft x}}\} \qquad (11)$$

On notera également $\div_{u_{-}^{s\triangleleft x}}$ et $\div_{u_{\triangleleft}^{s\triangleleft x}}$ les symboles de sub-

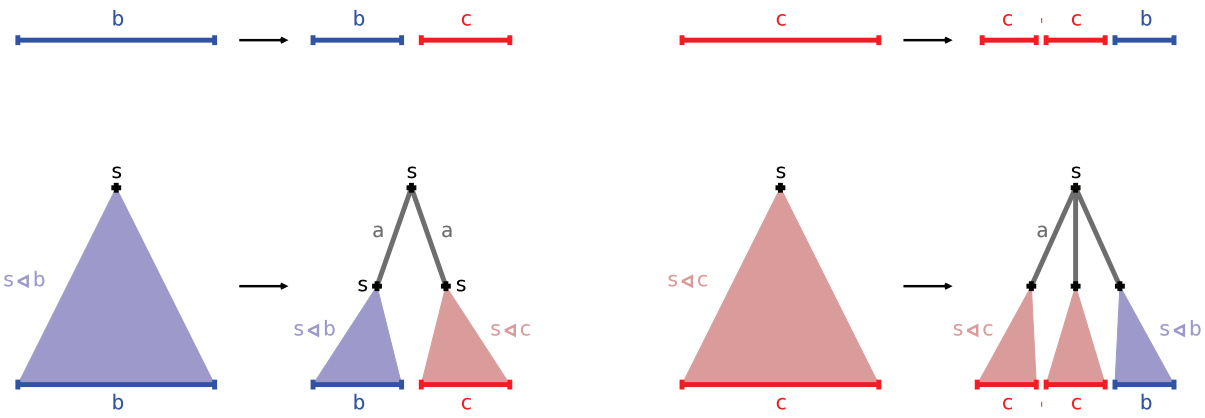


Figure 6: *Types de subdivision des arbres en fonction des types de subdivision de leur bord.*

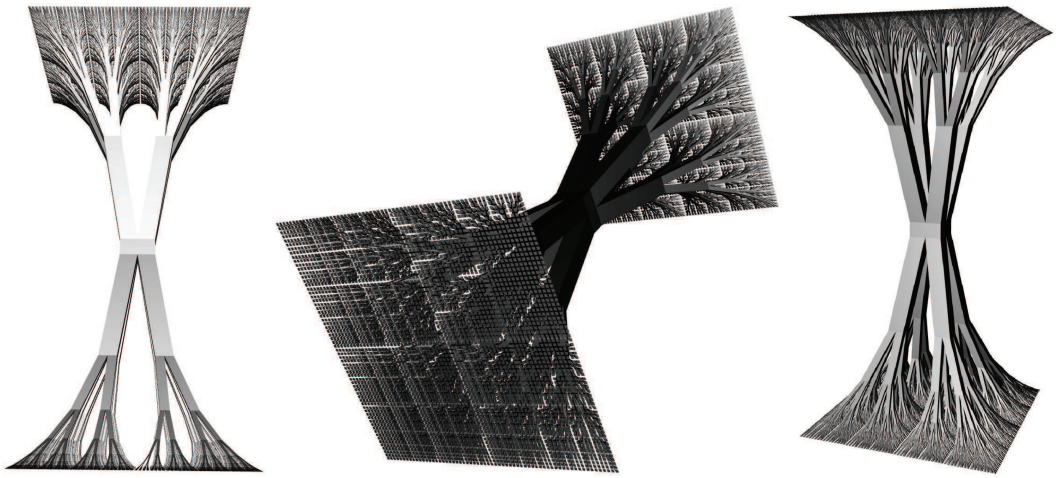


Figure 7: *Arbres bordés par une surface.*

division de l'arbre en une branche et un arbre associés à la subdivision $\div_{u^x}$:

$$\begin{array}{l}\forall x \in \delta_2^*(b, L^b_{\div}) \,, \\ \Sigma^{s\lhd x}_{\div} = \{\div_{u^{s\lhd x}_{-}} \mid \div_{u^x} \in \Sigma^x_{\div}\} \cup \{\div_{u^{s\lhd x}_{\lhd}} \mid \div_{u^x \in \Sigma^x_{\div}}\}\end{array} \tag{12}$$

Pour la branche a on définit :

$$\Sigma^a_{\div} = \{\div_{0^a}\} \;,\; \Sigma^a_{\partial} = \{\partial_{0^a}, \partial_{1^a}\} \tag{13}$$

4.4. Transitions

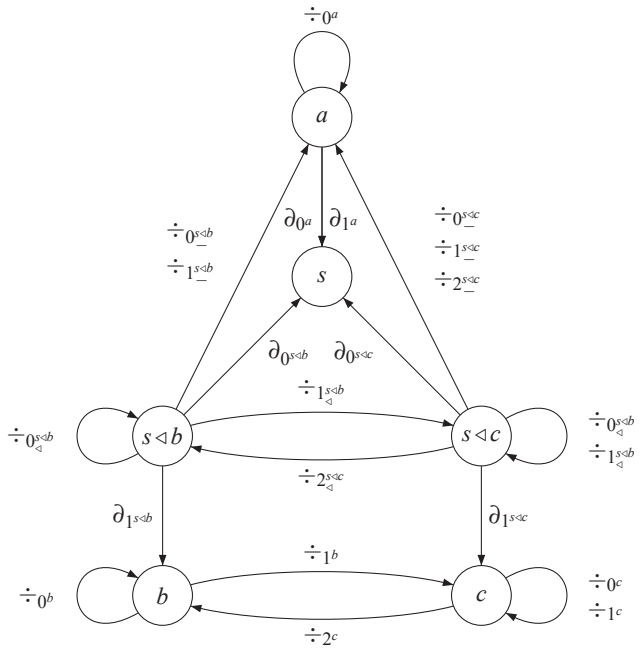
La branche a est bordée par deux sommets s. On lui attribue un opérateur de subdivision, qui permet de conservér cette branche aux étapes suivantes de subdivision après sa création.

$$\begin{array}{lll}\delta(a, \div_{0^a}) = a & \delta(a, \partial_{0^a}) = s \\ & \delta(a, \partial_{1^a}) = s\end{array} \tag{14}$$

À chaque subdivision du bord d'un arbre est associé deux subdivisions pour l'arbre, une qui génère une branche, et l'autre qui génère un sous-arbre, dont le type dépend du type de subdivision du bord, comme le montre la figure 6.

$$\begin{array}{l}\forall x \in \delta_2^*(b, L^b_{\div}) \,, \forall \div_{u^x} \in \Sigma^x_{\div} \,, \\ \delta(s\lhd x, \div_{u^{s\lhd x}_{-}}) = a \\ \delta(s\lhd x, \div_{u^{s\lhd x}_{\lhd}}) = s\lhd \delta_2(x, \div_{u^x})\end{array} \tag{15}$$

Pour l'exemple de la figure 6, l'automate généré est le suivant :



4.5. Relation d'équivalence

Les branches générées par la subdivision de l'arbre sont raccordées à la racine de l'arbre. L'autre extrémité de la branche est un sommet intermédiaire auquel est raccordé la racine d'un sous-arbre de subdivision. Les feuilles de ce sous-arbre correspondent à une subdivision des feuilles de l'arbre initial.

$$\begin{aligned}
&\forall x \in \delta_2^*(b, L_\div^b)\ ,\\
&\gamma_\div^{s\lhd x} \quad = \quad \{(\partial 0^{s\lhd x}\ \div 0^s\ ,\ \div u_-^{s\lhd x}\ \partial 0^a \mid \div u^x \in \Sigma_\div^x\} \cup\\
&\qquad\qquad \{(\div u_-^{s\lhd x}\ \partial 1^a\ ,\ \div u_\lhd^{s\lhd\delta(x,\div u^x)}\ \partial 0^{s\lhd x}) \mid \div u^x \in \Sigma_\div^x\} \cup\\
&\qquad\qquad \{(\div u_\lhd^{s\lhd x}\ \partial 1^{s\lhd\delta(x,\div u^x)}\ ,\ \partial 1^{s\lhd x}\ \div u^x) \mid \div u^x \in \Sigma_\div^x\}
\end{aligned} \tag{16}$$

Enfin, la subdivision de la branche a conserve ses extrémités.

$$\gamma_\div^a = \{(0\ \partial_0\ ,\ \partial_0\ 0)\ ,\ (0\ \partial_1\ ,\ \partial_1\ 0)\} \tag{17}$$

5. Conclusion

Le modèle BCIFS permet d'expliciter le codage de la subdivision topologique des objets. Ainsi il est possible de décrire et de générer n'importe quel schéma de subdivision classique ou même fractal et ceci quelle que soit la dimension des cellules topologiques. La contre partie est que la définition des relations d'incidence et d'adjacence peut devenir fastidieuse dès que les dimensions des cellules sont de 2 et de 3. Dans cet article nous montrons qu'il est possible de définir des opérateurs de combinaison ou de construction de topologie, à partir de la donnée de structures topologiques initiales. La première méthode proposée génère un "produit topologique". La deuxième méthode génère la description topologique d'une structure arborescente à partir d'une structure autosimilaire qui sera la limite de la structure arborescente. Ces opérateurs génère automatiquement toutes les contraintes d'incidence et d'adjacence et déterminent la topologie des objets. Par la suite, à ces structures topologiques peuvent être associés des ensembles de points de contrôle et à l'aide des points de subdivision nous contrôlons le plongement géométrique.

Références

[GaET11] GOUATY G., ANS ERIC TOSAN H. H. : Vers un modeleur basé sur un formalisme itératif et sur la géométrie fractale. In *GTMG 2011, Journées du Groupe de Travail en Modélisation Géométrique, Dijon* (30 - 31 Mars 2011), pp. –.

[Gen92] GENTIL C. : *Les fractales en synthèse d'image : le modèle IFS*. PhD thesis, Université LYON 1, 1992.

[Gou10] GOUATY G. : *Modélisation géométrique itérative sous contraintes*. PhD thesis, École Polytechnique Fédération de Lausanne, 2010.

[TBSG*06] TOSAN E., BAILLY-SALLINS I., GOUATY G., STOTZ I., BUSER P., WEINAND Y. : Une modélisation géométrique itérative basée sur les automates. In *GTMG 2006, Journées du Groupe de Travail en Modélisation Géométrique, Cachan* (22-23 Mars 2006), pp. 155–169.

[ZT96] ZAIR C. E., TOSAN E. : Fractal modeling using free form techniques. *Comput. Graph. Forum. Vol. 15*, Num. 3 (1996), 269–278.

Étude des opérations de CAO pour la géométrie fractale *

Anton Mishkinis[1], Christian Gentil[1] , Sandrine Lanquetin[1] et Dmitry Sokolov[2]
[1]LE2I - Université de Bourgogne
[2] LORIA - Université Nancy I

2012

Résumé

Dans le but de fabriquer des formes fractales pour l'architecture, le design, la plasturgie, on étudie la problématique liée à leur conception. Les formes géométriques fractales sont décrites à partir du modèle BCIFS. On cherche à contrôler les différentes propriétés de ces objets pour proposer un ensemble d'outils d'aide à la conception.

Les systèmes de CAO actuels ne sont pas adaptés à la manipulation des structures fractales car ils reposent sur des concepts qui sont souvent mis en défaut par les propriétés spécifiques des fractales. Il est alors nécessaire de développer des concepts nouveaux, d'étendre les concepts existants ou d'adapter les algorithmes. Par exemple, les notions de différentielle et de topologie sont plus complexes et plus subtiles en géométrie fractale.

Le premier objectif de cet article est de répertorier les différentes opérations d'aide à la conception et de construction présentes dans un système de CAO. Le deuxième est d'identifier les propriétés élémentaires nécessaires à leur réalisation. Le résultat est donné sous forme d'un graphe de dépendance. Enfin le troisième objectif est de classifier ces opérations en fonction de leur adaptation au modèle BCIFS. On propose un algorithme général permettant de calculer une approximation du résultat d'une opération de CAO dans le cas où on ne peut pas en donner une expression formelle.

Mots clés : BCIFS, Boundary Controlled Iterated Function System, opérations de CAO, classification, graphe de dépendance, approximation

1 Introduction

La définition de forme à partir d'un procédé de construction itératif permet d'obtenir des structures aux propriétés particulières : rugosité, lacunarité, etc. Il est généralement difficile, voire impossible, de représenter ces formes à l'aide des modèles géométriques classiques bien que ce principe de construction itératif soit exploité pour évaluer certains objets : algorithme de De Casteljau [1, 2], algorithme de Chaikin [3], surfaces de subdivi-

*REMERCIEMENT : Ces travaux ont été en partie financés par l'Agence Nationale de la Recherche (ANR) au travers du programme COSINUS (projet MODITERE n° ANR-09-COSI-014)

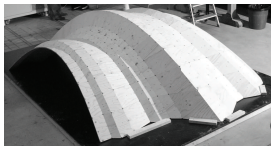


FIGURE 1 – Exemple d'application en architecture, EPFL 2010.

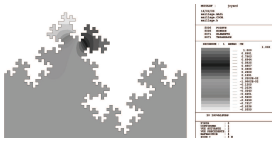


FIGURE 2 – L'étude des propriétés acoustiques des structures fractales.

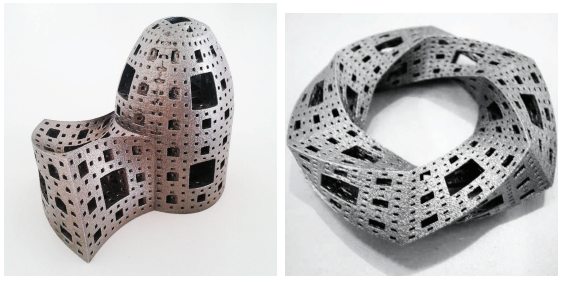


FIGURE 3 – Exemples de fabrication des formes fractales.

sion [4, 5]. Des modèles spécifiques ont été développés pour répondre de façon générale à cette problématique : L-system [6], IFS [7, 8], LRIFS/CIFS [9], BCIFS [10]. Ces modèles sont notamment utilisés pour la génération de structures fractales en synthèse d'images mais également pour la compression [11].

On s'intéresse à l'exploitation de ces structures décrites à partir du modèle BCIFS dans un contexte de CAO. On cherche à contrôler les différentes propriétés de ces objets et proposer un ensemble d'outils d'aide à la conception. Les applications sont l'architecture (voir figure 1), le design, la conception de formes pour l'étude de propriétés acoustiques (voir figure 2), la conception de surfaces complexes, etc. Cette démarche est entreprise dans l'objectif de fabriquer ces formes fractales (voir figure 3). Dans la suite de l'article le terme fractale désigne un objet décrit par un BCIFS.

Les systèmes de CAO actuels ne sont pas adaptés à la manipulation des structures fractales. Ils reposent sur des concepts qui sont mis en défaut du fait des propriétés spécifiques des fractales. Il est alors nécessaire de développer des concepts nouveaux ou d'étendre les concepts existants pour les objets fractals. Par exemple, la notion de différentielle est plus complexe et plus subtile en géométrie fractale [12, 13] : courbe non-différentiable sur un ensemble de points dense dans la courbe, courbe de longueur infinie. Il en est de même pour la topologie [14, 15, 16].

D'un autre coté, Zaïr et Tosan [17] ont montré que le résultat de certaines opérations peuvent être obtenues formellement. Par exemple le produit tensoriel de deux courbes fractales peut être décrit par un BCIFS composé des produits tensoriels des matrices représentant les opérateurs. Mais généralement, il est en moins possible de proposer un algorithme calculant une approximation du résultat d'une opération donnée : calcul de l'enveloppe convexe des attracteurs [18], le calcul d'intersection entre deux attracteurs, visualisation par lancer de rayon [19, 20, 21]. Le premier objectif de cet article est de répertorier les différentes opérations de construction présentes dans un système de CAO. Le deuxième est d'identifier les propriétés élémentaires nécessaires à leur réalisation. Le résultat est donné sous forme d'un graphe de dépendance. Et enfin le troisième objectif est de classifier ces opérations en fonction de leur adaptation à notre modèle.

L'article est organisé de la manière suivante. On commence par rappeler la définition des modèles IFS et BCIFS dans la section 2. Dans la section 3, on répertorie les opérations de construction disponibles dans les systèmes de CAO et on les classifie en fonction des propriétés nécessaires à leur application. On définit 4 propriétés principales exploitées par toutes les opérations de construction. On propose de classer ces opérations en 2 classes selon le type de résultat obtenu et on identi-

fie 3 situations possibles dans l'adaptation des opérations et algorithmes au modèle BCIFS. Dans la section 4, on propose un algorithme général permettant d'évaluer une approximation, quelle que soit l'opération, à condition que celle-ci vérifie certaines propriétés. On commence par décrire le principe de l'algorithme et identifier les propriétés nécessaires à l'application de l'algorithme. Ensuite, on montre qu'il est possible d'optimiser cet algorithme en vérifiant certaines propriétés supplémentaires. Enfin, pour conclure on répertorie les opérations de construction auxquelles l'algorithme d'approximation peut être appliqué dans le tableau 1.

2 Rappels et notations

Dans cette section, on rappelle les définitions principales et les propriétés des modèles itératifs (IFS, CIFS, BCIFS) et on établit les notations utilisées dans l'article.

2.1 IFS

Étant donné un espace métrique complet $(\mathbb{X}, d)$. Un *IFS* (Iterated Function System) est défini par un ensemble fini d'opérateurs contractants $T = \{T_i\}_{i=0}^{N-1}$ opérant sur les points de $\mathbb{X}$.

On munit l'ensemble des compacts non vides de $\mathbb{X}$, noté $\mathcal{H}(\mathbb{X})$, de la distance de Hausdorff d_H induite par la métrique d :

$$d_H(A,B) = \max\{\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\}.$$

L'espace métrique $(\mathbb{X}, d)$ étant complet, alors $(\mathcal{H}(\mathbb{X}), d_H)$ l'ensemble des compacts non vides de $\mathbb{X}$ muni de d_H est lui-même complet.

Il est possible de définir un opérateur $\mathbb{T} : \mathcal{H}(\mathbb{X}) \to \mathcal{H}(\mathbb{X})$, appelé opérateur de Hutchinson [7], comme l'union des opérateurs T_i :

$$\mathbb{T}(K) = \bigcup_{i=0}^{N-1} T_i(K).$$

Les opérateurs d'un IFS définissent un attracteur $\mathcal{A} \in \mathcal{H}(\mathbb{X})$ de manière récursive :

$$\mathcal{A} = \mathbb{T}(\mathcal{A}) \quad (1)$$

L'attracteur peut être défini comme limite d'une suite de figures construites à partir d'une figure initiale K :

$$\lim_{n\to\infty} K = \mathbb{T}^n(K) \quad (2)$$

Pour un nombre d'itérations n fixé, $\mathbb{T}^n(K)$ donne une approximation de $\mathcal{A}$.

2.2 CIFS

Pour les IFS, toutes les transformations sont appliquées à chaque itération. Il est possible d'enrichir ce modèle en ajoutant des règles contrôlant le processus itératif. C'est le principe des *CIFS* (Controlled IFS). La description se fait à l'aide d'un automate [22]. Les transitions représentent les transformations qui auront le droit d'être appliquées à partir de chaque état. Aux états sont associés les espaces de construction des attracteurs. Un état initial est défini. Il est alors possible de contrôler plus finement le processus de génération de l'attracteur.

Un CIFS définit une famille d'attracteurs associés aux états. Les attracteurs d'un CIFS sont définis de manière mutuellement récursive. L'attracteur du CIFS correspond à l'attracteur associé à l'état initial.

Les modèles IFS et CIFS permettent de modéliser un ensemble de formes variées. Cependant, il est difficile de contrôler leurs propriétés topologiques. Les formes sont déterminées par la donnée d'un ensemble d'opérateurs géométriques. La modification de ces opérateurs entraîne des modifications de la figure aussi bien globales que locales, ce qui affecte non seulement la géométrie, mais également la topologie. Afin de dissocier le contrôle global du contrôle local, les attracteurs sont construits dans des espaces barycentriques puis projetés suivant un ensemble de points de contrôle dans l'espace de modélisation. La forme globale est modifiable à l'aide de points de contrôle alors que la géométrie locale (texture géométrique) est définie par les opérateurs de subdivision de l'attracteur, i.e. les transformations du CIFS.

2.3 BCIFS

Afin de contrôler la structure topologique des formes modélisées, on enrichie le modèle CIFS en intégrant la notion de B-Rep pour obtenir le modèle *BCIFS* (Boundary Controlled Iterated Function System) [10].

Il est alors possible de spécifier des contraintes d'incidence et d'adjacence [23, 24, 25] en relation avec le procédé de subdivision et ainsi de contrôler la topologie résultante : topologie classique (courbe, surface,..) ou topologie fractale. Les notions B-Rep utilisées sont plus générales que les notions B-Rep classiques. Une cellule topologique peut être, le cas échéant, un objet fractal. Par exemple, une face peut être un triangle de Sierpiński, ou une arête un ensemble de Cantor, mais la structure B-Rep reste cohérente.

3 Graphe des opérations

Pour permettre l'exploitation du modèle BCIFS dans un système de CAO, il faut être capable de réaliser la plupart des opérations "standards" de construction. Dans cette section on répertorie les différentes opérations présentes dans un système de CAO (ici Rhino, choisi en fonction de l'utilisation faite par nos partenaires).

Certaines de ces opérations sont composites, cela signifie qu'elles peuvent être obtenues à partir d'opérations élémentaires. Par exemple, pour réaliser un raccord G_1 entre deux courbes il faut être capable de calculer les tangentes à chacune des extrémités des courbes. Mais il faut également être capable de construire une courbe dont les tangentes aux extrémités sont imposées.

Dans notre analyse on propose une première structuration de ces opérations en déterminant les dépendances entre elles. Après avoir répertorié l'ensemble des opérations, on propose un diagramme représentant les relations entre ces opérations sous forme d'un graphe de dépendance (voir figure 9).

Le graphe illustre les relations entre ces opérations et les propriétés des objets. Dans le diagramme, les propriétés et les méthodes de base sont représentées sous forme de losange. Ce graphe ne prétend pas être exhaustif que ce soit par rapport aux opérations ou aux dépendances. Mais il est construit en considérant au moins une solution pour chaque opération.

Ces opérations sont classées dans deux catégories suivant le types de solutions proposées pour le système de CAO :

1. celles pour lesquelles une solution exacte est fournie (repérées par une rectangle)
2. celles pour lesquelles une solution approchée est donnée (repérées par une ellipse)

Les opérations de construction exploitent ou sont basées sur quatre propriétés principales : invariance affine, structure B-Rep, paramétrisation et propriétés différentielles. Pour les formes décrites par BCIFS, la propriété d'invariance affine est immédiatement vérifiée car les objets sont définis à partir de points de contrôle. La structure B-Rep est formalisée par le modèle BCIFS et permet d'étendre les notions de topologie classique à celle de topologie fractale [14, 15, 16]. La paramétrisation est obtenue à partir de la notion d'adresse associée au BCIFS. Enfin les propriétés différentielles on été abordées et sont encore à développer.

Les propriétés de base étant en partie définies, il est possible d'aborder l'adaptation des opérations ou algorithmes au modèle BCIFS. On distingue alors trois cas de figure.

1. On peut donner une expression formelle du résultat de l'opération. Un premier exemple est l'application d'une transformation affine sur un objet. Les formes étant définies à partir d'un ensemble de points de contrôle, il suffit d'appliquer la transformation à ces points de contrôle. Un autre exemple est le produit tensoriel de deux courbes fractales qui est obtenu en réalisant le produit tensoriel des matrices représentant les opérateurs des BCIFS [17].
2. Les propriétés fractales étant spécifiques,

certaines constructions nécessitent une généralisation. Par exemple, une opération "Pull curve onto surface" ne peut pas être utilisée dans sa forme originale. Cette opération projette la courbe sur la surface le long de la normale à la surface. Mais, la normale de la surface fractale est discontinue, ainsi l'image de cette projection est également discontinue. Pour avoir un résultat plus intuitif il est nécessaire de redéfinir ou de généraliser cette opération.

3. On peut construire une approximation du résultat sans en donner une représentation formelle. Dans la section suivante, on détaille ce cas et on propose un algorithme général.

4 Calculs approximatifs

4.1 Principe de l'algorithme

Dans le cas où il n'est pas possible de donner une description formelle du résultat, on propose un algorithme général de calcul d'une approximation. Plus précisément, on montre qu'il est possible de définir un algorithme permettant d'évaluer une approximation, quelle que soit l'opération, à condition que celle-ci vérifie certaines propriétés.

On illustre nos propos à partir du modèle IFS, sachant que la généralisation au modèle BCIFS est immédiate.

Soit un IFS $\{T_i\}_{i=0}^{N-1}$ défini sur l'espace métrique complet $(\mathbb{X}, d_{\mathbb{X}})$ avec un attracteur $\mathcal{A}$. On considère une opération arbitraire $Op : \mathcal{H}(\mathbb{X}) \to \mathbb{Y}$. L'objectif de l'algorithme général est de fournir le résultat par une approximation à ε près donnée. On distingue alors deux cas.

L'espace $\mathbb{Y}$ est un espace métrique. Notre objectif se formalise par la construction de l'ensemble $C \in \mathbb{Y}$ tel que :

$$d_{\mathbb{Y}}(Op(\mathcal{A}), C) \leqslant \varepsilon.$$

A partir d'un compact $B \in \mathcal{H}(\mathbb{X})$, on construit une suite d'objets convergents vers l'attracteur $\mathcal{A}$ de la manière suivante. On note $r = d_H(\mathcal{A}, B)$. L'opérateur d'Hutchinson de l'IFS étant contractant, on a :

$$d_H(\mathcal{A}, \mathbb{T}^k(B)) \leqslant s(\mathbb{T})^k \cdot r \xrightarrow[k\to\infty]{} 0,$$

où $s(\mathbb{T})$ est un coefficient de contraction de l'opérateur d'Hutchinson $\mathbb{T}$ et k représente le nombre d'itérations.

Par conséquent, $\forall \varepsilon > 0 \;\; \exists K \in \mathbb{N}$:

$$\forall k \geqslant K \;\; d_H(\mathcal{A}, \mathbb{T}^k(B)) < \varepsilon.$$

Pour résoudre le problème assigné, il suffit que l'opérateur Op soit continu. On obtient alors :

$$d_{\mathbb{Y}}(Op(\mathcal{A}), Op(\mathbb{T}^k(B))) \xrightarrow[k\to\infty]{} 0,$$

mais dans le cas général, il est impossible de déterminer le nombre d'itérations à effectuer pour obtenir la précision ε donnée.

Cependant, si l'opérateur Op est lipschitzien avec une constante de Lipschitz L :

$$d_{\mathbb{Y}}(Op(\mathcal{A}), Op(\mathbb{T}^k(B))) \leqslant L \cdot d_H(\mathcal{A}, \mathbb{T}^k(B)) \leqslant$$
$$\leqslant L \cdot s(\mathbb{T})^k \cdot r \xrightarrow[k\to\infty]{} 0.$$

Ainsi, $\forall \varepsilon > 0 \;\; \exists K \in \mathbb{N}$:

$$\forall k \geqslant K \;\; d_{\mathbb{Y}}(Op(\mathcal{A}), Op(\mathbb{T}(B))) < \varepsilon.$$

On construit l'ensemble $Op(\mathbb{T}^k(B))$ qui approche $Op(\mathcal{A})$ avec une précision ε donnée. Le nombre d'itérations nécessaire est donnée par :

$$k = \left\lceil \frac{log(\frac{\varepsilon}{L \cdot r})}{log(s(\mathbb{T}))} \right\rceil$$

Des exemples de telles opérations sont :

- l'intersection avec un ensemble $E \subset \mathbb{X}$ $Op_E : \mathcal{A} \mapsto \mathcal{A} \cap E$ (voir figure 4)
- la différence avec un ensemble $E \subset \mathbb{X}$ $Op_E : \mathcal{A} \mapsto \mathcal{A} \setminus E$
- l'enveloppe convexe de l'attracteur $Op : \mathcal{A} \mapsto conv(\mathcal{A})$

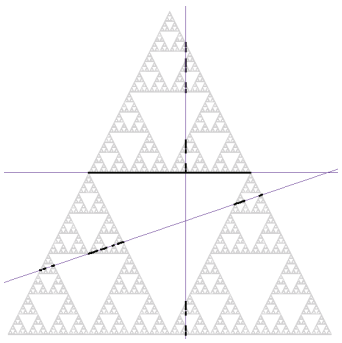


FIGURE 4 – Résultat d'intersection entre le triangle de SIERPINSKI et les droites.

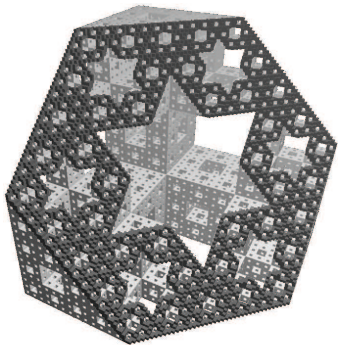


FIGURE 5 – La figure illustre une approximation de l'intersection du cube de MENGER avec un demi-espace.

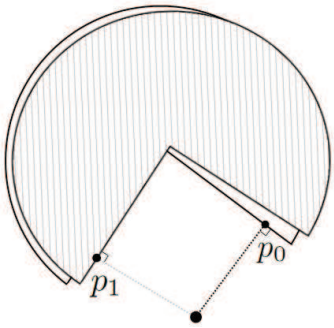


FIGURE 6 – Exemple d'un opérateur discontinu. Le point le plus proche d'une approximation p_1 diffère du point le plus proche de l'attracteur p_0.

Remarque. Si l'opérateur Op n'est pas continu, on ne peut rien dire sur la convergence de $Op(\mathbb{T}^k(B))$ vers $Op(\mathcal{A})$. De tels opérateurs peuvent donner un résultat faux pour toutes les approximations de l'attracteur. Par exemple, l'opérateur déterminant le point de l'attracteur le plus proche à un point donné :

$$Op \ : \mathcal{A} \mapsto p \in \mathcal{A} \text{ tq } d(p,\mathcal{A}) \leqslant d(q,\mathcal{A}) \ \ \forall q \in \mathcal{A},$$

n'est pas continu. La convergence de l'approximation vers l'attracteur ne garantit pas celle du résultat de l'application de l'opérateur vers la solution (voir figure 6).

Si l'espace $\mathbb{Y}$ n'est pas métrique, il est difficile d'affirmer quoi que ce soit sur le résultat de l'opération Op. On peut simplement "espérer" que l'application de l'opérateur Op à une approximation à ε près de l'attracteur donne le résultat correct. Le choix de ε peut être fait en fonction de l'opérateur et/ou du contexte.

Un exemple d'un tel opérateur est de vérifier si un point donné appartient à l'attracteur :

$$Op_p \ : \mathcal{A} \mapsto p \in \mathcal{A}.$$

L'espace $\mathbb{Y}$ est alors un espace booléen $\{oui, non\}$. Pour cette opération, on ne peut dire que si un point appartient à l'ε-approximation de l'attracteur, mais pas à l'attracteur. Ici le choix de ε peut être fait en fonction d'un seuil de tolérance à l'erreur.

4.2 Modification de l'algorithme

La construction de l'approximation de l'attracteur est coûteuse, car on doit calculer un nombre d'objets ($\mathbb{T}^n(B)$) croissant exponentiellement avec le nombre d'itérations. On applique chaque transformation de l'IFS à l'objet initial B. A l'itération suivante, on procède de même sur chaque résultat précédent. Le calcul de $\mathbb{T}^n(B)$ peut être symbolisé par un arbre dont chaque feuille est le résultat de l'application d'une suite de transformations de l'IFS à B (voir figure 7). Si B et Op vérifient certaines propriétés, qu'on précisera, l'algorithme décrit dans la section 4.1 peut être optimisé.

Dans l'algorithme précédent, on calcule une approximation $\mathbb{T}^n(B)$ de l'attracteur, puis l'opération est évaluée sur cette approximation, i.e. on calcule $Op(\mathbb{T}^n(B))$.

L'étape préliminaire à l'optimisation est de pouvoir évaluer $Op(\mathbb{T}^n(B))$ à partir de l'application de l'opérateur à chaque feuille de l'arbre

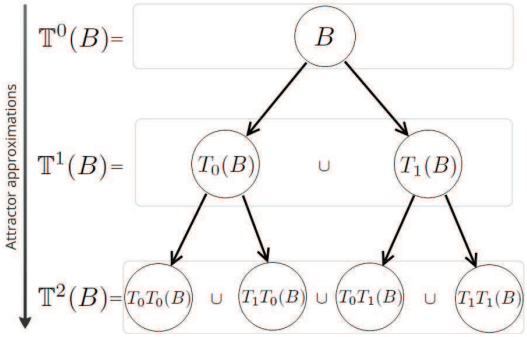


FIGURE 7 – Le calcul des approximations de l'attracteur est illustré par un arbre dont chaque feuille est le résultat de l'application d'une suite de transformations de l'IFS à un compact initial B.

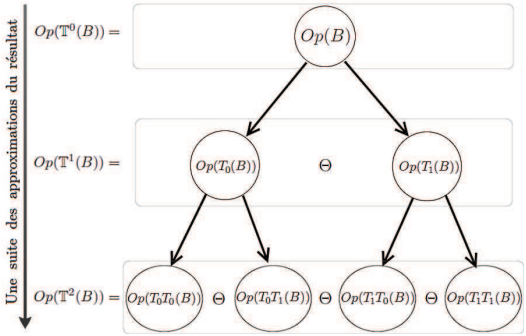


FIGURE 8 – Le calcul de l'approximation du résultat à partir de l'application de l'opérateur à chaque feuille de l'arbre. On compose les résultats par une loi de composition interne associative Θ.

(voir figure 8)

S'il existe une loi de composition interne associative Θ définie sur $\mathbb{Y}$ telle que :

$$Op(A \cup B) = Op(A) \ \Theta \ Op(B),$$

alors l'application de Op sur $\mathbb{T}^n(B)$ peut être obtenue par composition des évaluations Op sur chaque feuille, i.e. :

$$Op(\bigcup_i T_i(B)) = \underset{i}{\Theta} \ Op(T_i(B)).$$

Ainsi, si les propriétés suivantes sont vérifiées :

- la loi de composition interne Θ possède un élément neutre, noté e, dans l'espace $\mathbb{Y}$, (i.e. $\mathbb{Y}$ est un *monoïde*),
- $\forall A \subset B \quad Op(B) = e \Rightarrow Op(A) = e$,

alors, en choisissant l'objet initial B tel que $B \supset T_i(B) \quad \forall i \in \{0, \cdots, N-1\}$, il n'est pas nécessaire d'explorer la totalité de l'arbre. En effet, s'il existe i tel que $Op(T_i(B)) = e$ alors $Op(T_iT_j(B)) = e, \forall j$.

L'élément neutre e représente le résultat de l'opérateur Op qui n'influence pas un autre résultat lors de leur composition par Θ : si $Op(A) = e \Rightarrow Op(A \cup B) = Op(A) \ \Theta \ Op(B) = Op(B)$

A titre d'exemple on considère l'opérateur déterminant si l'attracteur intersecte ou pas une droite donnée. Soit l une droite, l'opération Op est alors définie de la manière suivante :

$$Op_l(\mathcal{A}) = \begin{cases} \text{vrai}, & \text{si } \mathcal{A} \cap l \neq \emptyset; \\ \text{faux}, & \text{autrement.} \end{cases}$$

Dans ce cas, l'espace $\mathbb{Y}$ est une ensemble non métrique $\{vrai, faux\}$.

Pour construire un approximation de l'attracteur $\mathcal{A}$, on choisit comme compact initial une boule englobante $B \supseteq \mathcal{A}$, telle que $\mathbb{T}(B) \subset B$. Pour cet exemple, l'espace $\mathbb{Y}$ est une monoïde sur la fonction logique OU avec un élément neutre "faux".

L'opérateur Op remplit alors toutes les conditions et on a :

$$\left(\bigcup_i T_i(B) \cap l \neq \emptyset \right) \Leftrightarrow \underset{i}{\text{OU}} \left(T_i(B) \cap l \neq \emptyset \right).$$

Pour un niveau d'approximation n donné, il suffit de déterminer si $Op(B) = e$. Si c'est le cas, on arête et on peut en déduire que $Op(\mathbb{T}^n(B)) = e$ c'est-à-dire que le résultat est "faux". Sinon, on détermine $Op(T_i(B), \forall i$. Si tous les résultats sont égaux à e on en déduit $Op(\mathbb{T}(B)) = e$ et

que la droite n'intersecte pas $Op(\mathbb{T}^n(B))$. S'il existe i_0 tel que $Op(T_{i_0}(B) \neq e$ alors on poursuit l'exploration en déterminant $Op(T_{i_0}T_j(B), \forall j \in \{0, \cdots, N-1\}$. Pour les résultats donnant e on arête l'exploration pour les autres on continue jusqu'au niveau n.

On remarque alors qu'il est uniquement nécessaire d'être capable d'évaluer l'opération sur un objet (généralement une boule) de la géométrie classique ou sur la transformée de cet objet par une transformation affine.

Un autre exemple est l'opération qui détermine l'intersection de $\mathcal{A}$ avec une droite donnée. Le résultat est un ensemble de points de la droite. En se donnant un repère sur la droite, $\mathbb{Y}$ peut être assimilé à $\mathbb{R}$ que l'on munit de la distance de Hausdorff. Et on a : $\Theta = \cup$ et $e = \emptyset$. En appliquant notre algorithme général on peut déterminer une approximation de l'intersection de la droite avec l'attracteur (voir figure 4).

La figure 5 montre une approximation de l'intersection du cube de Menger avec un demi-espace.

Voici quelques exemples d'opérations qui satisfont aux conditions :

- la différence avec une ensemble $C \subset \mathbb{X}$
  $Op_C : \mathcal{A} \mapsto \mathcal{A} \setminus C$, $\Theta = \cup$
- l'intersection avec une ensemble $C \subset \mathbb{X}$
  $Op_C : \mathcal{A} \mapsto \mathcal{A} \cap C$, $\Theta = \cup$
- la fonction caractéristique de $s \in \mathbb{X}$
  $Op_s : \mathcal{A} \mapsto s \in \mathcal{A}$, $\Theta = \mathrm{OU}$
- la distance de sommet $s \in \mathbb{X}$
  $Op_s : \mathcal{A} \mapsto \min_{a \in \mathcal{A}} d(s, a)$, $\Theta = \min$

5 Conclusion

La conception de forme dans un système de CAO est une démarche complexe. Pour assister l'utilisateur, les modeleurs géométriques sont enrichis d'opérations de construction. Ces opérations sont généralement basées sur certaines propriétés géométriques des objets.

Dans le but de développer un modeleur itératif pour concevoir des objets fractals décrits à l'aide du modèle BCIFS, on étudie dans quelle mesure ces opérations peuvent être exploitées dans ce contexte.

Après avoir répertorié l'ensemble des opérations d'un système de CAO, on propose un diagramme représentant les relations de dépendances entre ces opérations sous forme d'un graphe.

Une première analyse montre que ces opérations reposent sur quatre types de propriétés : l'invariance affine, la structure B-Rep, la paramétrisation et les propriétés différentielles.

Ces propriétés sont alors essentielles et doivent trouver leur équivalent en géométrie fractale.

On classe ces opérations dans deux catégories suivant le type de solutions proposées pour les systèmes de CAO : s'il existe une représentation formelle exacte du résultat, le résultat est donnée sous forme d'un modèle approché.

Pour le modèle BCIFS cette classification se décline en trois catégories. Les deux premières sont identiques aux deux catégories précédemment décrites. La troisième correspond aux cas pour lesquels les propriétés des objets fractals sont plus fines que celles des objets géométriques classiques (notion de rugosité ou de lacunarité par exemple) et l'application des opérateurs n'est pas immédiate. Il faut alors étudier les concepts sur lesquels reposent ces opérateurs pour en proposer une généralisation.

Enfin, on propose un algorithme général permettant d'appliquer un opérateur à une fractale à condition que cet opérateur vérifie un certain nombre de contraintes. Ainsi s'il est possible de calculer le résultat de l'application de l'opérateur sur une boule alors il est possible d'en calculer un résultat approché sur une fractale. En ajoutant quelques conditions supplémentaires sur l'opérateur on montre que cet algorithme peut être optimisé.

On formalise la notion de calcul approché du résultat en identifiant les situations pour lesquelles il est possible de contrôler cette approximation et les cas où cela n'est pas possible.

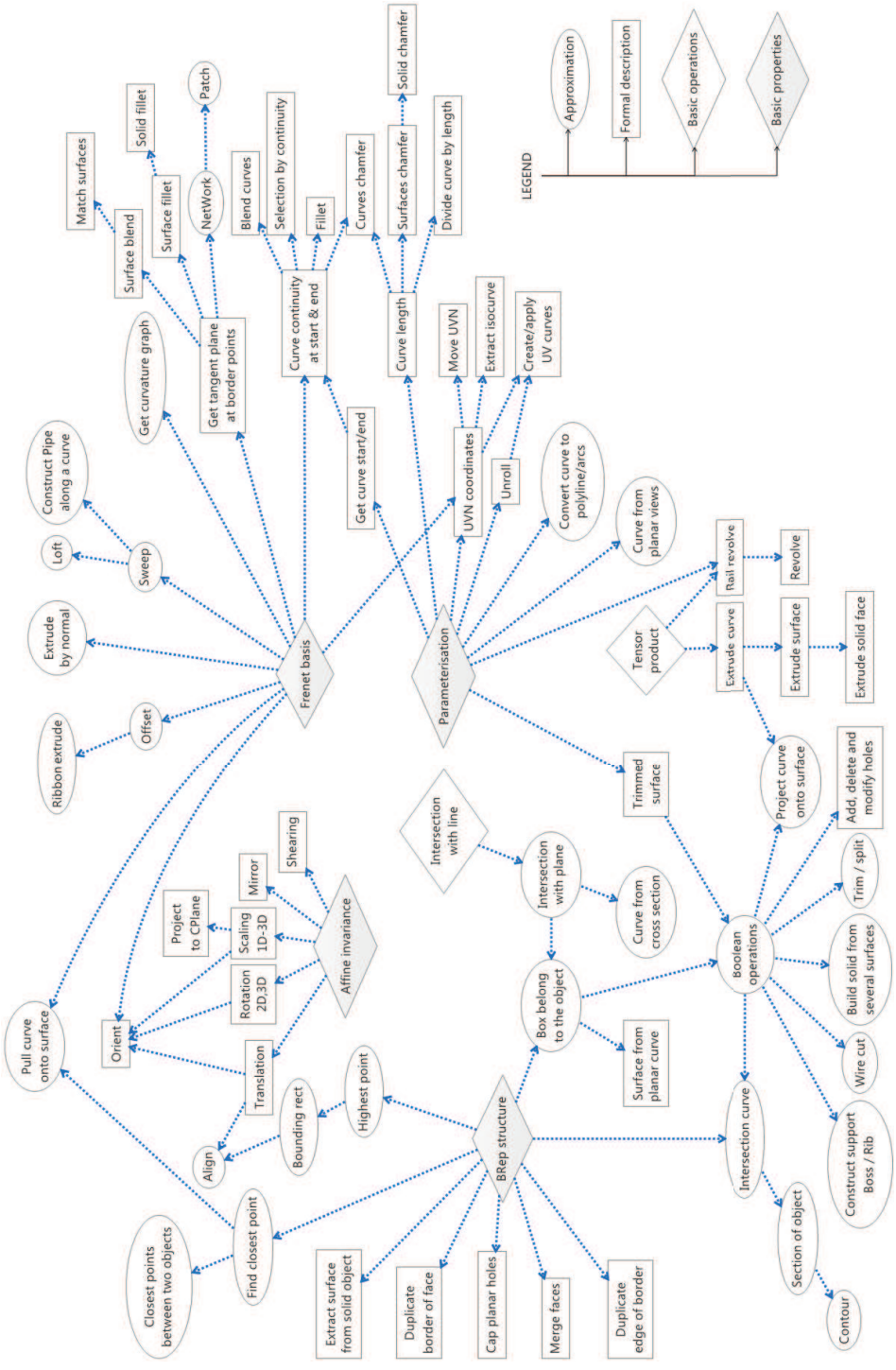


FIGURE 9 – Graphe des relations entre les opérations de CAO et les propriétés des objets.

Références

[1] P. D. Casteljau, *Formes à Pôles*. Paris, Hermes, (1985).

[2] G. Farin, D. Hansford, *The Essentials of CAGD*. Natic, MA : A. K. Peters, Ltd, (2000).

[3] G. Chaikin, *An algorithm for High Speed Curve Generation*. CGIP 3 (1974), pp. 346-349.

[4] J. Warren, *Subdivision methods for geometric design*. Dept. of Computer Science, Rice University, (Nov 1995)

[5] S. Lanquetin, *Étude des surfaces de subdivision : intersection, precision et profondeur de subdivision*. Thèse de doctorat. Université de Bourgogne, 4 oct 2004. Jury : S. Akkouche, M. Daniel, S. Hahmann, F. Merienne, G. Morin, M. Neveu.

[6] P. Prusinkiewicz, A. Lindenmayer *The Algorithmic Beauty of Plants*. Springer-Verlag (1990), pp. 101–107.

[7] M. Barnsley, *Fractals everywhere*. Academic Press Professional, Inc., San Diego, CA, USA, (1988).

[8] C. Gentil, *Les fractales en synthèse d'images : le modèle IFS*. Thèse de doctorat. Université LYON I, 24 mar 1992. Jury : D. Vandorpe, P. Chenin, J. Mazoyer, J. P. Reveilles, J. Levy Vehel, M. Terrenoire, E. Tosan.

[9] P. Prusinkiewicz, M. Hammel *Language-Restricted Iterated Function Systems, Koch Constructions, and L-systems*. SIGGRAPH'94 Course Notes. ACM Press, (1994).

[10] E. Tosan, I. Bailly-Salins, G. Gouaty, I. Stotz, P. Buser, Y. Weinand, *Une modélisation géométrique itérative basée sur les automates*. GTMG 2006, Journées du Groupe de Travail en Modélisation Géométrique (Mar. 2006), pp. 155-169. Organisation LURPA EA 138, AFIG, GDR ALP. 2

[11] Y. Fisher, *Fractal Image Compression, Theory and Application*. Springer-Verlag (1995).

[12] H. Bensoudane, *Étude différentielle des formes fractales*. Thèse de doctorat. Université de Bourgogne, 20 oct 2009. Jury : M. Barnsley, S. Hahmann, E. Tosan, J. Levy Vehel, D. Michelucci, M. Sabin, M. Neveu, C. Gentil

[13] H. Bensoudane, C. Gentil, M. Neveu, *The Local Fractional Derivative of Fractal Curves*. Signal Image Technology and Internet Based Systems, SITIS '08, IEEE 2008, pp. 422-429.

[14] G. Gouaty, *Étude différentielle des formes fractales*. Thèse de doctorat. Université de Bourgogne, 20 oct 2009. Jury : M. Barnsley, S. Hahmann, E. Tosan, J. Levy Vehel, D. Michelucci, M. Sabin, M. Neveu, C. Gentil

[15] G. Gouaty, H. Tosan, Y. Weinand, I. Stotz *Modélisation géométrique itérative et approche B-Rep*. GTMG 2010, Journées du Groupe de Travail en Modélisation Géométrique, Dijon, (2010).

[16] G. Gouaty, H. Hnaidi, E. Tosan, *Vers un modeleur basé sur un formalisme itératif et sur la géométrie fractale*. GTMG 2011, Journées du Groupe de Travail en Modélisation Géométrique, Grenoble, (2011).

[17] C. E. Zair, E. Tosan, *Fractal modeling using free form techniques*. EUROGRAPHICS'96, Blackwell Publishers, Eurographics Association, 15(3) (1996)

[18] A. Mishkinis, C. Gentil, S. Lanquetin, D. Sokolov *Approximation de l'enveloppe convexe de l'attracteur d'un IFS affine*. GTMG 2011, Journées du Groupe de Travail en Modélisation Géométrique, Grenoble, (2011).

[19] J. C. Hart, T. A. DeFanti *Efficient anti-aliased rendering of 3D linear fractals.*. Computer & Graphics 25(4) (1991), pp. 91–100.

[20] T. Martyn, *Realistic rendering 3D IFS fractals in real-time with graphics accelerators*. Computers & Graphics 34(2) (2010), pp. 167-175.

[21] T. Martyn, *Efficient ray tracing affine IFS attractors*. Computers & Graphics 25 (2001), pp. 665-670.

[22] R. D. Mauldin, S. C. Williams, *Hausdorff dimension in graph directed constructions*. Transactions of the American Mathematical Society 309(2) (1988), pp. 811–829.

[23] E. Brisson, *Representing geometric structures in d dimensions : Topology and order*. Tech. Rep. 88-11-07, Departement of Computer Science, University of Washington, (1988).

[24] Y. E. Kalay, *Modelling polyhedral solids bounded by multicurved parametric surfaces*. CAD 15(3) (1983), pp. 141–146.

[25] P. Lienhardt, *Topological models for boundary representation : A comparison with n-dimensional generalized maps*, CAD 23(1) (1991), pp. 59–82, 2.

[26] T. Martyn, *Efficient ray tracing affine IFS attractors*. Computers & Graphics 25 (2001), pp. 665-670.

[27] T. Martyn, *Realistic rendering 3D IFS fractals in real-time with graphics accelerators*. Computers & Graphics 34 (2010), pp. 167-175.

Construction d'un raccord entre deux surfaces autosimilaires de subdivision topologique quadrangulaire †

Sergey Podkorytov[1], Christian Gentil[1] , Dmitry Sokolov[2] et Sandrine Lanquetin[1]

[1]LE2I - Université de Bourgogne
[2] LORIA - Université Nancy I

Résumé

Le but de notre travail est d'étudier les propriétés des courbes et surfaces construites à partir de procédés itératifs afin d'en contrôler le comportement différentielle. Ces courbes et surfaces sont décrites à l'aide de BCIFS (Boundary Controled Iterated Function System).

Dans cet article, on montre comment le modèle BCIFS peut être utilisé pour construire une surface intermédiaire joignant deux autres surfaces de natures différentes, c'est-à-dire construites à partir de deux procédés itératifs différents (par exemple : une surface fractale avec une surface de Bézier ou deux surfaces B-splines de degrés différents). On ne traite ici que le cas de la subdivision topologique quadrangulaire.

Le formalisme BCIFS est utilisé pour décrire la subdivision topologique de la surface intermédiaire. Les contraintes de raccords topologiques sont déduites du graphe associé au BCIFS pour garantir la continuité de la surface globale. Notre construction est basée uniquement sur la description de cette subdivision topologique et ceci indépendamment de la configuration des réseaux de points de contrôle de chacune des surfaces initiales. En particulier, cela signifie qu'il est possible d'appliquer cette construction entre un schéma de subdivision primal et un schéma de subdivision dual (par exemple une surface spline bi-quadratique avec une surface spline bi-cubique). Enfin, on donne quelques résultats sur les propriétés différentielles de la construction.

Mots-clés : Système de fonctions itérées, attracteur d'un IFS, raccord, surface, dérivabilité

1. Introduction

Notre objectif global est de développer un modeleur géométrique itératif basé sur le concept de géométrie fractale. Cette approche donne accès à un nouvel univers de formes difficilement représentables et manipulables par les modeleurs classiques. Pour que les utilisateurs puissent disposer des outils de conception classiques, on doit adapter les opérations et les algorithmes de construction aux spécificités des formes fractales. Par exemple, on a montré que les courbes et surfaces fractales possèdent des propriétés différentielles particulières [Ben09]. Ces propriétés se traduisent d'un point de vue géométrique par des notions de textures géométriques ou de "rugosités", qui correspondent à des états intermédiaires entre une continuité G_0 et une continuité G_1. Ainsi, définir un raccord entre deux surfaces fractales devient encore plus complexe qu'en géométrie classique.

Notre modeleur géométrique est développé à partir du modèle BCIFS (Boundary Controled Iterated Function System) proposé par TOSAN [TBSG*06]. Le formalisme BCIFS est basé sur celui d'IFS (Iterated Function System) initialement proposé par HUTCHINSON [Hut81] et développé par BARNSLEY [Bar88]. A la notion d'attracteur autosimilaire est ajoutée celle de structure B-Rep, où les volumes, faces et arêtes peuvent être des structures fractales (éponge de Menger, triangle de Sierpinski, ensemble de Cantor). Dans notre modèle, les opérateurs de subdivision utilisés sont des opérateurs linéaires.

Dans [PGSL11], on propose une méthode de construction de raccords entre deux courbes fractales définies par BCIFS. Dans cet article on montre comment cette méthode peut être

†. REMERCIEMENT : Ces travaux ont été en partie financés par l'Agence Nationale de la Recherche (ANR) au travers du programme COSINUS (projet MODITERE n° ANR-09-COSI-014)

utilisée pour réaliser un raccord entre deux surfaces fractales.

Ce problème a été étudié dans le contexte des surfaces de subdivision [LL03] dans le cas où les surfaces sont définies par des schémas de subdivision différents (quadrangulaire/triangulaire). Les méthodes sont essentiellement basées sur le raffinement des maillages de contrôle. Ces approches engendrent des problèmes pour la construction de raccord entre un schéma primal et un schéma dual. C'est notamment le cas pour des subdivisions quadrangulaires avec les schémas de Catmull-Clark et Doo-sabin, le nombre de points de contrôle de part et d'autres n'étant pas compatibles.

Plus précisément, on aborde le cas des surfaces de subdivision quadrangulaires. Notre méthode utilise le modèle BCIFS qui permet d'explicité et de coder la subdivision topologique. Ainsi, la construction de la surface raccord est réalisée à partir d'une subdivision choisie de façon cohérentes par rapport aux deux surfaces initiales. L'écriture des contraintes d'incidence et d'adjacence garantissent la continuité C_0. Comme notre méthode exploitant uniquement les propriétés de subdivision topologique, elle peut être appliquée à partir de toutes les surfaces (de subdivision quadrangulaire) indépendamment du nombre et de la configuration des points de contrôle. En particulier on peut traiter le cas de deux surfaces de subdivision, l'une étant définie à partir du schéma de Catmull-Clark et l'autre à partir du schéma de Doo-sabin.

La surface résultante est évaluée par une procédure récursive basée sur la subdivision topologique. A chaque niveau d'itération on peut donner une approximation de la surface sous forme d'un maillage cohérent. Enfin l'analyse spectrale des opérateurs de subdivision permet de caractériser le comportement différentiel du raccord.

L'article est organisé de la manière suivante. Dans la section 2, on rappelle la notion d'IFS, de BCIFS et de contraintes d'adjacence et d'incidence. Puis dans la section 3, on décrit le processus itératif permettant de construire la surface intermédiaire entre deux surfaces données et on définit les contraintes d'adjacence et d'incidence permettant de garantir la continuité de la surface intermédiaire et de ces jonctions avec les surfaces initiales. Ensuite on étudie les propriétés différentielles des surfaces obtenues dans la section précédente (section 4). Finalement, dans la section 5 on donne des exemples de surfaces construites avec notre méthode.

2. Rappel IFS-BCIFS

2.1. IFS et IFS projeté

Étant donné un espace métrique complet (E,d), un *IFS* (Iterated Function System) est un ensemble fini d'opérateurs contractants $\mathbb{T} = \{T_i\}_{i=0}^{N-1}$ opérant sur les points de E.

Il est possible de définir un opérateur $\mathbb{T} : \mathcal{H}(E) \to \mathcal{H}(E)$, appelé opérateur de Hutchinson, comme l'union des opérateurs T_i. À chaque compact non-vide K, cet opérateur associe $\bigcup_{i=0}^{N-1} T_i(K)$. Les opérateurs T_i étant contractants dans l'espace (E,d), l'opérateur $\mathbb{T}$ est contractant dans $(\mathcal{H}(E), d_{\mathcal{H}})$, l'espace des compacts non-vides de E muni de la distance de Hausdorff [Bar88].

D'après [Hut81], il existe un unique compact A tel que $\mathbb{T}(A) = A$: le point fixe de $\mathbb{T}$. De plus, A peut être calculé comme la limite de $\mathbb{T}^n : A = \lim_{n\to\infty} \mathbb{T}^n(K)$. Cette limite est indépendante du choix du compact non-vide K. Cette propriété est illustrée par la suite d'images de la figure 1.

La notion d'IFS projeté a été introduite par Zaïr et Tosan [ZT96]. En séparant l'espace d'itération de l'espace de modélisation, il est possible de construire des formes fractales à pôles. De la même manière que les courbes splines sont déterminées par les fonctions de base définies dans un espace barycentrique, les attracteurs sont définis dans des espaces barycentriques dont les dimensions correspondent aux nombres de points de contrôle : $A \subset BI^n = \{\lambda \in \mathbb{R}^n | \sum_{i=0}^{P-1} \lambda_i = 1\}$, où n représente le nombre de points de contrôle. Ensuite l'attracteur est projeté dans l'espace de modélisation suivant la matrice des points de contrôle : $PA = \{\sum_{i=0}^{n-1} P_i\lambda_i | \lambda_i \in A\}$, où $P = [P_0\ P_1 \cdots P_{n-1}]$ est le vecteur composé des points de contrôle.

Cette construction impose aux transformations d'être des opérateurs internes aux espaces barycentriques. Pour des opérateurs linéaires représentés sous forme matricielle cela revient à avoir des colonnes dont la somme des termes est égale à 1.

Figure 1: La fougère de BARNSLEY.

2.2. BCIFS

Le modèle BCIFS permet de munir les IFS d'une structure B-Rep [TBSG*06]. Ceci présente l'avantage d'expliciter les notions de face-arête-sommet implicites aux structures des attracteurs [Gen92]. Il est alors possible d'écrire des contraintes d'incidence et d'adjacence en relation avec le procédé de subdivision et ainsi de contrôler la topologie résultante : topologie classique (courbe, surface,..) ou topologie fractale. Les notions B-rep utilisées sont plus générales que les notions B-Rep classiques. Une cellule topologique peut être, le cas échéant, un objet fractal. Par exemple, une face peut être un triangle de SIERPINSKI, ou une arête un ensemble de CANTOR, mais la structure B-Rep reste cohérente.

Le système de subdivision est décrit par un automate dont

les états représentent les cellules topologiques. On identifie deux types de transition :

- Les transitions qui bouclent sur un état. Elles représentent les opérateurs de subdivision qui décrivent comment chaque cellule se subdivise en elle-même.
- Les transitions reliant deux états différents. Elles représentent des *Opérateurs de bord*. Ces opérateurs décrivent comment un bord (une cellule) est plongé dans la cellule de niveau supérieur.

Les cellules topologiques sont définies dans des espaces barycentriques et ensuite projetées dans l'espace de modélisation suivant une matrice de points de contrôle. Ainsi, à chaque état associé à une cellule topologique on fait correspondre un espace barycentrique dont la dimension correspond au nombre de points de contrôle de la cellule. Une primitive d'affichage est également affectée à chaque état et est utilisée comme figure initiale pour la construction de l'approximation de l'attracteur.

On distingue un état initial noté $\natural$, auquel est associé l'espace de modélisation. Une première transition relie cet état à la cellule de plus au niveau. Cette transition correspond à l'opération de projection de l'espace barycentrique dans l'espace de modélisation suivant l'ensemble des points de contrôle. Cette opération est représentée par la matrice composée des points de contrôle. Le formalisme détaillé peut être trouvé dans [TBSG*06, Gou10]

La figure 2 montre un automate associé à la description d'une courbe. Dans cet exemple on distingue le sommet droit du sommet gauche.

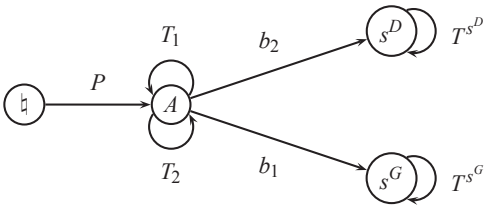


Figure 2: Un automate représentant la subdivion d'une courbe.

2.2.1. Contraintes d'incidence et d'adjacence

La description d'une courbe fournie par l'automate de la figure 2 est insuffisante pour garantir que l'attracteur final soit effectivement une courbe. A cet automate, il faut associer un ensemble de contraintes d'incidence et d'adjacence garantissant les raccords des différentes cellules entre elles au cours du processus de subdivision. Le déroulement de l'automate au premier niveau d'itération engendre le processus de subdivision illustré par le graphe de la figure 3. Pour

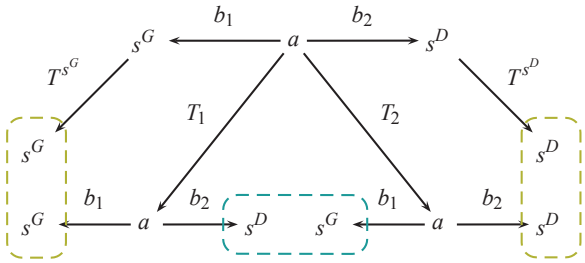


Figure 3: Automate représentant la subdivision d'une courbe.

des opérateurs de subdivision quelconques, rien ne garantit que lors du processus de subdivision les arêtes subdivisées vont bien se raccorder selon les sommets et que les sommets seront bien incidents aux arêtes. Il est alors nécessaire de contraindre les chemins dans le graphe pour garantir la topologie finale. On doit alors définir 2 contraintes d'incidences $b_2 T^{s^D} = T_2 b_2$ et $b_1 T^{s^G} = T_1 b_1$ et une contrainte d'adjacence $T_1 b_2 = T_2 b_1$. La résolution de ces contraintes induit des structures dans les matrices des opérateurs de subdivision. La figure 4 montre la structure générale que doivent posséder les opérateurs de subdivision d'une courbe définie par n points de contrôle et dont les sommets dépendent de p points de contrôle.

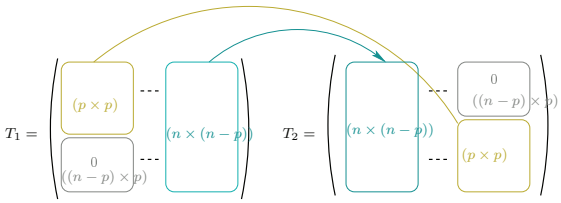


Figure 4: Structure générale des matrices de subdivision pour une courbe construite à partir de n points de contrôle et dont les sommets dépendent de p points de contrôle.

2.3. Définition d'une surface quadrangulaire

Le même principe peut être utilisé pour décrire le processus de subdivision d'une face quadrangulaire. Un automate pour surface quadrangulaire a trois états : F pour la face, A pour les arêtes bordant la face (ici toutes les arêtes sont de même nature, i.e. définies par le même procédé itératif) et S pour les sommets (voir figure 5).

Dans le cas des surfaces, le nombre de contraintes d'incidence et d'adjacence est plus important mais le principe de base est le même. Pour une subdivision quadrangulaire l'ensemble des contraintes d'incidence et d'adjacence est illustré dans la figure 6.

On obtient deux contraintes d'adjacence pour chaque

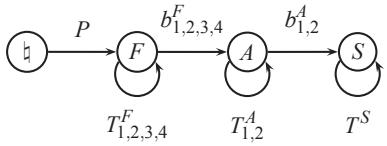


Figure 5: BCIFS pour les surfaces quadrangulaires.

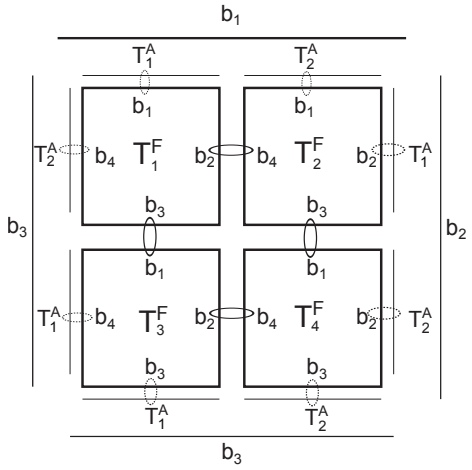


Figure 6: Illustration des contraintes d'incidence et d'adjacence pour une subdivision topologique quadrangulaire. Chaque subdivision de la face doit se raccorder aux subdivisions des autres faces suivant une arête (Ces raccords sont symbolisés par des ellipses en trait plein). Les subdivisions des bords de la face doivent correspondre aux bords des subdivisions (Ces correspondances sont représentées par des ellipses en pointillés)

arête de carreaux. Pour l'arête supérieure :

$$T_1^F b_1^F = b_1^F T_1^A,$$

$$T_2^F b_1^F = b_1^F T_2^A.$$

Pour l'arête droite :

$$T_2^F b_2^F = b_2^F T_1^A,$$

$$T_3^F b_2^F = b_2^F T_2^A.$$

Pour l'arête inférieure :

$$T_4^F b_3^F = b_3^F T_1^A,$$

$$T_3^F b_3^F = b_3^F T_2^A.$$

Et finalement pour l'arête gauche :

$$T_1^F b_4^F = b_4^F T_1^A,$$

$$T_4^F b_4^F = b_4^F T_2^A.$$

Ensuite, il y a quatre contraintes d'incidence pour chaque arête entre deux carreaux :

$$T_1^F b_2^F = T_2^F b_4^F,$$

$$T_2^F b_3^F = T_3^F b_1^F,$$

$$T_3^F b_4^F = T_4^F b_2^F,$$

$$T_4^F b_1^F = T_1^F b_3^F.$$

3. Construction de la surface intermédiaire

Dans cette section, on décrit le processus itératif construisant la surface raccordant deux surfaces données. On part d'un BCIFS décrivant deux surfaces initiales (voir figure 7). L'état $\natural$ désigne l'espace de modélisation, les états G et D sont les racines des deux sous-automates définissant les deux surfaces initiales. A chacun de ces sous-automates est associé l'ensemble des contraintes d'incidence et d'adjacence décrites au paragraphe 2.3 garantissant la topologie de ces surfaces.

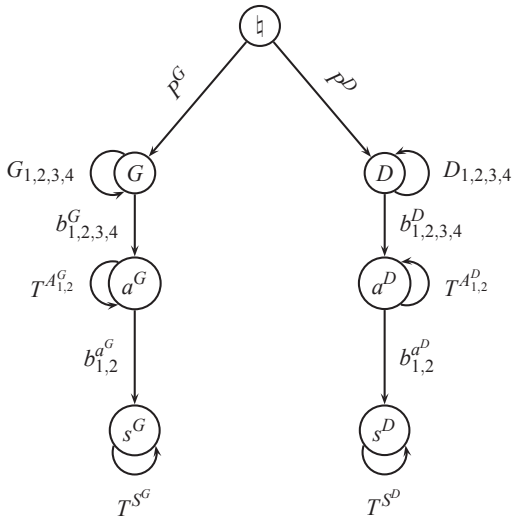


Figure 7: BCIFS initial. Les deux branches correspondent aux surfaces initiales.

On ajoute un état I pour représenter et définir la construction de la surface intermédiaire. Cette surface se subdivise en six carreaux : deux carreaux de même type que la première surface initiale (état G), deux carreaux de même type que la deuxième surface initiale (état D), et les deux derniers carreaux de subdivision similaire à la surface intermédiaire elle-même (voir figure 8).

Ainsi la surface intermédiaire est construite itérativement à l'aide de copies des surfaces initiales. Il ne reste plus qu'à

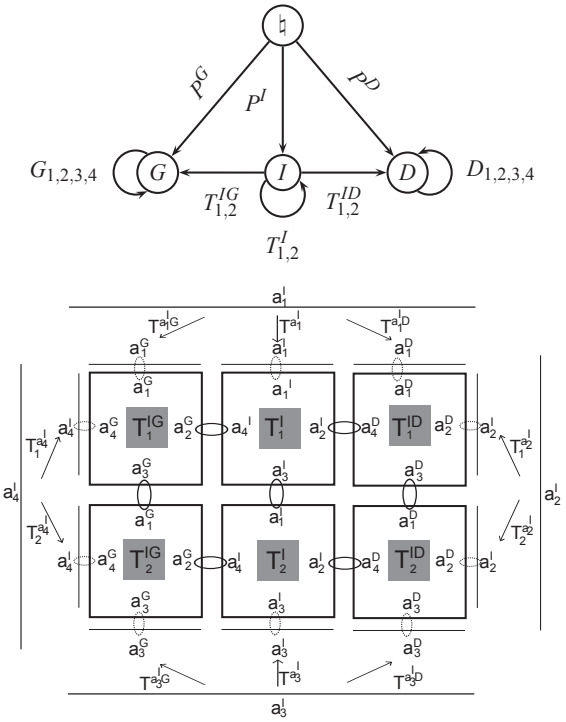


Figure 8: En haut : un automate décrivant la subdivision topologique de la surface intermédiaire. En bas : représentation symbolique de cette subdivision à partir de laquelle on déduit les contraintes d'incidence (ellipse en pointillés) et les contraintes d'adjacence (ellipse en trait plein).

définir les contraintes d'incidence et d'adjacence garantissant la continuité de l'ensemble. Pour cela on commence par définir les contraintes garantissant la continuité de la surface intermédiaire (contraintes d'incidence paragraphe 3.1, contraintes d'adjacence 3.2). Puis on définit les conditions de raccord avec les deux surfaces initiales - contraintes d'adjacence entre les surface initiales et surface intermédiaire (paragraphe 3.3).

3.1. Contraintes d'incidence

La surface intermédiaire est bordée par quatre courbes. Pour chacune d'entre elles il faut contraindre le système de subdivision pour que la subdivision du bord corresponde aux bords des subdivisions.

Pour le bord inférieur, on obtient les équations suivantes :

$$\begin{aligned} T_1^{IG} b_1^G &= b_1^I T_1^{a^I G} \\ T_1^I b_1^I &= b_1^I T_1^{a^I} \\ T_1^{ID} b_1^D &= b_1^I T_1^{a^I D} \end{aligned}$$

Pour le bord supérieur, on obtient les équations suivantes :

$$\begin{aligned} T_2^{IG} b_3^G &= b_2^I T_2^{a^I G} \\ T_2^I b_3^I &= b_2^I T_2^{a^I} \\ T_2^{ID} b_3^D &= b_2^I T_2^{a^I D} \end{aligned}$$

Pour le bord gauche, on obtient les équations suivantes :

$$\begin{aligned} T_1^{IG} b_4^G &= b_4^I T_1^{a^G} \\ T_2^{IG} b_4^I &= b_4^I T_2^{a^G} \end{aligned}$$

Pour le bord droit, on obtient les équations suivantes :

$$\begin{aligned} T_1^{ID} b_2^D &= b_2^I T_1^{a^D} \\ T_2^{ID} b_2^D &= b_2^I T_2^{a^D} \end{aligned}$$

3.2. Contraintes d'adjacence

Les contraintes d'adjacence pour le carreau intermédiaire vont établir que les subdivisions sont bien raccordées entre elles. Il y a 7 arêtes partagées par les carreaux obtenus après la première itération (voir figure 8). Pour chacune de ces arêtes, nous devons écrire les contraintes sur les opérateurs de subdivision.

Pour les trois raccords "horizontaux", on obtient les équations suivantes :

$$T_1^{IG} b_3^G = T_2^{IG} b_1^G \quad (1)$$
$$T_1^I b_3^I = T_2^I b_1^I \quad (2)$$
$$T_1^{ID} b_3^D = T_2^{ID} b_1^D \quad (3)$$

Pour les quatre raccords "verticaux", on obtient les équations suivantes :

$$\begin{aligned} T_1^{IG} b_2^G &= T_1^I b_4^I \\ T_2^{IG} b_2^G &= T_2^I b_4^I \\ T_1^{ID} b_4^D &= T_1^I b_2^I \\ T_2^{ID} b_4^D &= T_2^I b_2^I \end{aligned}$$

3.3. Connexion aux surfaces initiales

Finalement, on doit s'assurer que le carreau intermédiaire relie les deux surfaces initiales.

Considérez le bord gauche et le bord droit de surface intermédiaire (voir figure 8). Le bord gauche est composé de deux parties de courbe (subdivisions) de type a^G (car chacune bord d'une copie d'un carreau de type G. Ces deux carreaux G sont raccordés entre eux par une contrainte d'adjacence (équation 1). Cette dernière induit que les deux parties du bord gauche est bien une courbe dont les subdivisions sont raccordées entre elles et donc est de type a^G. Il en est de même pour le bord droit qui est bien de type a^D.

Puisque les bords gauche et droit sont de même nature que

les bords des surfaces initiales respectives, il suffit de garantir qu'elle soient construites à partir des mêmes ensembles de points de contrôle à l'aide des équations suivantes :

$$P^I b_2^I = P^D b_4^D$$
$$P^I b_4^I = P_G b_2^G$$

A partir de ces conditions, on peut également déduire la dimension minimale de l'espace barycentrique associé à l'état I. C'est la somme des dimensions des espaces correspondants aux arêtes des surfaces initiales. Si nécessaire, cette dimension peut être augmentée. Ceci revient à ajouter des points de contrôle supplémentaires pour le carreau intermédiaire sans intervenir sur la continuité C_0.

3.4. Visualisation / Évaluation de la surface

Les contraintes d'incidence et d'adjacence garantissent la continuité C_0 de l'attracteur final du BCIFS. La visualisation se fait par la construction d'une suite de figures convergeant vers l'attracteur. On choisit alors un nombre d'itérations pour obtenir une approximation de l'attracteur (ici la surface finale). La construction de cette suite de figures se fait uniquement à partir des opérateurs de subdivision des faces et suivant les règles de composition décrites par l'automate [TBSG*06]. A chaque état E est associée une primitive K_E (un compact non-vide) qui est choisie en fonction de la nature de la subdivision topologique de l'attracteur associé à l'état. Dans l'absolu, il est possible de prendre n'importe quel compact non-vide, les propriétés des BCIFS garantissent la convergence vers l'attracteur (sous certaines conditions sur les opérateurs de subdivision [TBSG*06]). Mais la continuité ne sera obtenue que pour un nombre infini d'itérations.

Cependant, on peut choisir les primitives pour obtenir un maillage pour tous les niveaux d'itérations. Dans notre cas, chaque carreau possède une structure quadrangulaire, les primitives seront des facettes quadrangulaires décrites par quatre sommets. Ces facettes sont définies dans des espaces barycentriques et les sommets sont décrits par leurs coordonnées barycentriques en relation avec les points de contrôle. Deux carreaux adjacents possèdent des points de contrôle en commun. Donc, pour les primitives de partager les sommets, il faut que leurs sommets avoir les mêmes coordonnées barycentriques. Une solution particulière consiste à choisir les points fixes des opérateurs de subdivision comme sommets des primitives. Cette solution présente l'avantage de construire un maillage de la surface dont les sommets appartiennent à la surface quel que soit le niveau d'itération. La figure 9 illustre les cinq premiers niveaux d'itération.

Dans cette construction, la structure des points de contrôle n'intervient pas. Tout est basé sur la notion de subdivision topologique. Ainsi cela ne pose aucun problème pour raccorder un surface de subdivision primal avec un surface de subdivision dual comme les surfaces de Doo-Sabin et de Catmull-Clark.

La figure 10 (en haut) symbolise les trois surfaces (Catmull-Clark à gauche, intermédiaire au milieu et Doo-Sabin à droite) avec les réseaux des points de contrôle de chaque surface initiale. La figure 10 (en bas) représente les réseaux des points de contrôle raffinés par le processus de subdivision. Des nouveaux points de contrôle (repérés en pointillé) sont automatiquement ajoutés par le procédé. Ces nouveaux points de contrôle dépendent des deux réseaux de points de contrôle initiaux. Ce raffinement n'est qu'une conséquence de la description de la subdivision topologique et des contraintes d'incidence et d'adjacence. Cette illustration est réalisée en appliquant le procédé de subdivision à l'ensemble des points de contrôle comme étant un compact non-vide quelconque. Ces réseaux n'étant pas connexe, à chaque itération, il y aura une discontinuité. Mais pour un nombre infini d'itérations le procédé converge vers la même surface limite continue.

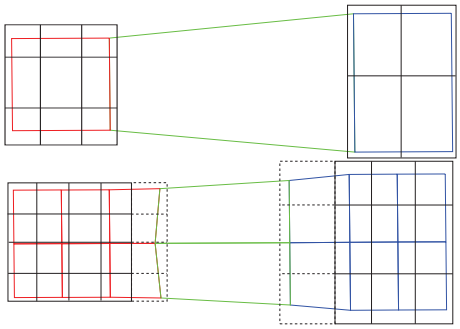


Figure 10: Représentation du raffinement des points de contrôle par le processus de subdivision.

4. Différentiabilité

Les propriétés différentielles sont liées aux valeurs et vecteurs propres des matrices des opérateurs de subdivision [Pra98, Ben09]. Dans cette section on présente une analyse des propriétés différentielles du raccord. Notre surface globale est composée de trois parties : les deux carreaux initiaux et le carreau intermédiaire. Le carreau intermédiaire est construit à partir de copies des carreaux initiaux, copies obtenues par des transformations linéaires. Ces copies possèdent donc les mêmes propriétés différentielles que les carreaux d'origine. Les propriétés différentielles des raccords entre les copies et les carreaux d'origine dépendent de l'appartenance des valeurs propres sous dominantes aux sous espaces associés aux bords et ceci dépend des valeurs des coefficients des matrices de subdivision. Dans le cas de schémas de subdivision classiques (Doo-sabin ou Catmull-Clark), on obtient le même type de continuité que pour un raccord de deux surfaces de même nature, i.e. on perd un degré de différentiabilité.

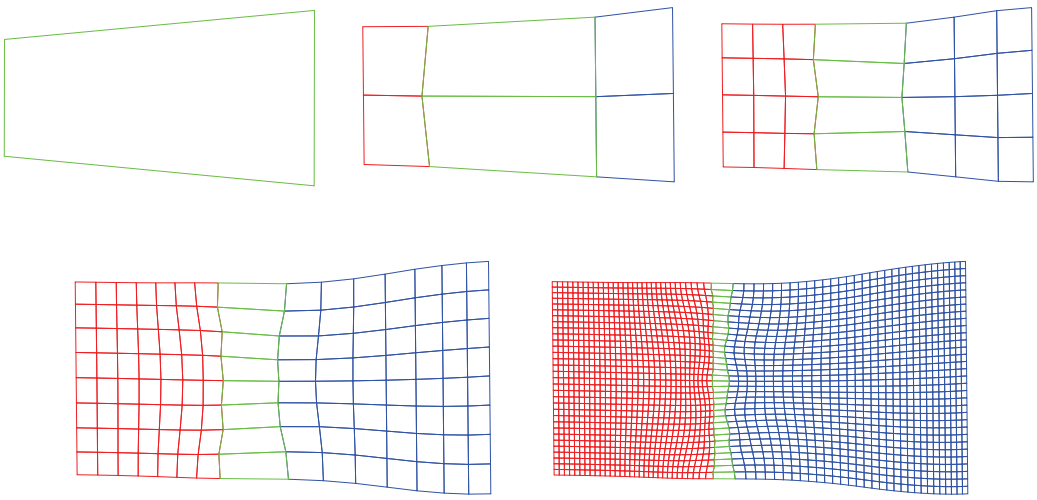


Figure 9: Visualisation de subdivision de surface intermédiaire.

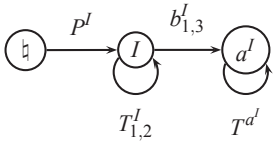


Figure 11: BCIFS sans les états G, D, e^G, e^D et les transitions correspondantes.

La construction du carreau intermédiaire est récursive et le procédé de copie des carreaux initiaux se poursuit jusqu'à combler l'espace entre les deux carreaux initiaux. Ainsi les copies gauche et droite vont se rejoindre suivant une courbe. Les propriétés différentielles au voisinage de cette courbe vont maintenant être abordées.

Cette courbe peut être obtenue comme sous attracteur du BCIFS global. Si on élimine les états G, D, e^G, e^D et les transitions associées, on obtient l'automate de la figure 11. L'état e^I a perdu deux de ces trois transitions. Cela signifie que cet état correspond à un sommet. De la même façon, l'état I est devenu une arête, les autres états n'étant pas considérés les contraintes d'incidence et d'adjacence sont celles d'une courbe. On a donc une arête bornée par deux sommets. On appellera cette courbe la *courbe de raccord* qui représente le bord limite des deux surfaces construites de part et d'autre par copies des carreaux d'origine.

Dans un premier temps, on étudie le comportement différentiel aux sommets de cette courbe. Ces sommets sont respectivement les points fixes des opérateurs T^I_1 et T^I_2. On considère le point fixe de T^I_1. On note S la surface intermédiaire. Sans perte de généralité, on peut supposer que le point fixe de T^I_1 est 0, i.e. $T^I_1(0) = 0$. Soit $\{\lambda_j\}^n_{j=0}$ l'ensemble des valeurs propres de T^I_1 tel que $1 > |\lambda_1| = |\lambda_2| > |\lambda_3| \geq \ldots \geq |\lambda_n|$, $\lambda_0 = 1$ et λ_1, λ_2 sont positives. On veut prouver que $\vec{v_1}$ et $\vec{v_2}$ constituent la base de l'espace tangent.

Pour cela il suffit de montrer que, pour toute suite $\{y_n\}^\infty_{n=0}$, telle que $\forall n, y_n \in S$ et $lim_{n\to\infty} y_n = 0$, on a : $f_n = f(y_n) \to 0$ où : $f(x) = \frac{dist(x,p)}{|Pr_p(x)|} \to 0$; $p = \{y | y = a\lambda_1 + b\lambda_2, \forall a, b \in \mathbb{R}\}$; Pr_p est la projection orthogonale sur le plan p.

Soit A_n une suite d'ensembles compacts où $A_0 = S$ et $A_n = \underline{T^I_1(A_{n-1})}, \forall n > 0$. On définit B_n une autre suite par : $B_n = A_{n+1} \setminus A_n$.

On note que $\forall n \geq 0 \quad B_n$ est un ensemble compact tel que $0 \notin B_n$. On a alors :

$$\cup^\infty_{n=k} B_n = A_k$$

Sans perdre de généralité, on peut considérer que le plan p est le plan Oxy. Ainsi :

$$f(x) = \frac{|x - Pr_p(x)|}{|Pr_p(x)|} = \frac{|(0,0,0,x_3,\ldots,x_n)|}{|(0,x_1,x_2,0,\ldots,0)|}.$$

Soit $x_1 \in B_1$, $\exists x_0 \in B_0$ telle que $T_0(x_0) = x_1$.

$$\begin{aligned} f(x_1) &= f(T_1^I(x_0)) = \\ &= \frac{|(0,0,0,\lambda_3 x_3,\ldots,\lambda_n x_n)|}{|(0,\lambda_1 x_1,\lambda_2 x_2,0,\ldots,0)|} \leq \\ &\leq \frac{|\lambda_3 \cdot (0,0,0,x_3,\ldots,x_n)|}{|\lambda_1 \cdot (0,x_1,x_2,0,\ldots,0)|} \leq \\ &\leq \frac{|\lambda_3|}{|\lambda_1|} f(x_0) \end{aligned}$$

Donc, pour tout n on peut écrire :

$$\max_{x\in B_n} f(x) = \frac{|\lambda_3|}{|\lambda_1|} \max_{x\in B_{n-1}} f(x) = \left(\frac{|\lambda_3|}{|\lambda_1|}\right)^n \max_{x\in B_0} f(x).$$

Depuis $\frac{|\lambda_3|}{|\lambda_1|} < 1$ donc $\lim_{n\to\infty} \max_{x\in B_n} f(x) = 0$. Cela implique également que $\max_{x\in A_n} f(x) \to 0$. Donc $\forall\{y_n\}\ \lim_{n\to\infty} f(y_n) = 0$.

Ainsi, des conditions suffisantes pour l'existence d'un plan tangent au point fixe de T_1^I sont les suivantes :

- T_1^I doit avoir deux valeurs propres égales positives sous-dominantes.
- Les espaces propres correspondants ne sont pas confondus.

La même démarche peut être appliquées pour le point fixe de T_2^I.

Si on suppose que les plans tangents existent aux points fixes de T_1^I and T_2^I. Puisque la courbe de raccord est continue, l'application de T_1^I au point fixe de T_2^I donne le même résultat que l'application de T_2^I au point fixe de T_1^I. Ainsi, si les résultats de l'application de ces opérateurs aux normales des plans tangents respectifs sont colinéaires, alors le plan tangent existe également aux points de raccord. Par autosimilarité cette propriété se propage sur un ensemble de points dense dans la courbe. L'étude du comportement différentiel des autres points reste pour l'instant un sujet de recherche.

5. Exemples

Dans cette section, on donne quelques exemples de surfaces de raccord construites par la méthode proposée.

Le premier exemple est réalisé à partir de deux surfaces fractales de subdivision topologique quadrangulaire (voir figure 12).

Le deuxième exemple (voir figure 13) est le cas particulier de construction d'une surface intermédiaire entre une surface de Doo-Sabin et une surface de Catmull-Clark.

Enfin, la figure 14 montre un troisième exemple de construction de raccord entre une surface lisse et une surface fractale.

6. Conclusion

A partir du modèle BCIFS, on a montré comment il est possible de construire une surface raccordant deux autres

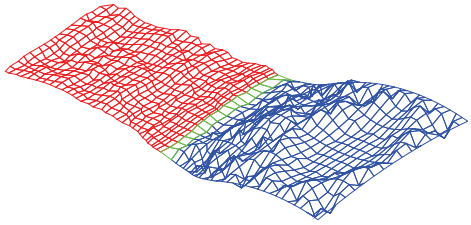


Figure 12: Une surface intermédiaire entre deux surfaces fractales.

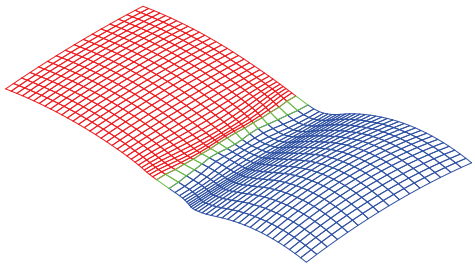


Figure 13: Une surface intermédiaire entre une surface de Doo-Sabin et une surface de Catmull-Clark.

surfaces de subdivision topologique quadrangulaire. La surface intermédiaire est décrite à l'aide d'un BCIFS, c'est-à-dire un automate décrivant le processus de subdivision et des équations traduisant les contraintes d'incidence et d'adjacence. Les contraintes garantissant la continuité C_0 d'une part de la surface de raccord et d'autre part des raccords de celle-ci avec les surfaces initiales. Le principe de fonctionnement de l'automate consiste à prolonger les deux surfaces initiales à l'aide de copies d'elles-mêmes jusqu'à se qu'elles raccordent le long d'une courbe de jonction. Cette construction est fondée uniquement sur les propriétés de subdivision topologique des surfaces initiales. Ainsi elle peut être utilisée quelles que soient les structures des points de contrôle des surfaces initiales. Un exemple d'application est donné pour la construction d'un raccord entre une surface construite par un schéma de subdivision primal (Catmull-Clark) et une autre construite par un schéma de subdivision dual (Doo-Sabin).

On propose une première analyse des propriétés différentielles de la construction. On montre que cette analyse doit être faite pour trois types de zone : pour chaque surface initiale, pour le raccord entre les surfaces initiales et leurs copies et enfin pour les limites des copies le long de la courbe de jonction. Les deux premiers cas s'étudient de façon classique. Pour le dernier cas on donne des conditions suffisantes pour que le raccord soit G_1 pour un ensemble de points dense dans la courbe de jonction.

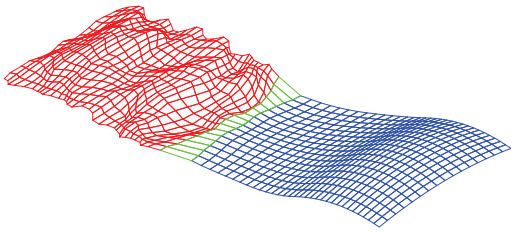


Figure 14: Une surface intermédiaire entre une surface lisse et une surface fractale.

Dans des travaux futurs on cherchera à caractériser plus finement le comportement différentiel le long de la courbe de jonction et surtout on cherchera, par l'ajout de points de contrôle à la surface intermédiaire, à contrôler ces propriétés différentielles.

Références

[Bar88] BARNSLEY M. : *Fractals everywhere*. Academic Press Professional, Inc., San Diego, CA, USA, 1988.

[Ben09] BENSOUDANE H. : *Etude differentielle des formes fractales*. PhD thesis, Université de Bourgogne, 2009.

[Gen92] GENTIL C. : *Les fractales en synthèse d'image : le modèle IFS.* PhD thesis, Université LYON 1, 1992.

[Gou10] GOUATY G. : *Modélisation géométrique itérative sous contraintes*. PhD thesis, École Polytechnique Fédération de Lausanne, 2010.

[Hut81] HUTCHINSON J. : Fractals and self-similarity. *Indiana University Journal of Mathematics. Vol. 30*, Num. 5 (1981), 713–747.

[LL03] LEVIN A., LEVIN D. : Analysis of quasi-uniform subdivision. *Applied and Computational Harmonic Analysis. Vol. 15*, Num. 1 (2003), 18 – 32.

[PGSL11] PODKORYTOV S., GENTIL C., SOKOLOV D., LANQUETIN S. : Propriétés différentielles du raccord entre deux courbes fractales. In *GTMG 2011, Journées du Groupe de Travail en Modélisation Géométrique, Dijon* (30 - 31 Mars 2011), pp. –.

[Pra98] PRAUTZSCH H. : Smoothness of subdivision surfaces at extraordinary points. *Advances in Computational Mathematics. Vol. 9* (1998), 377–389. 10.1023/A :1018945708536.

[TBSG*06] TOSAN E., BAILLY-SALLINS I., GOUATY G., STOTZ I., BUSER P., WEINAND Y. : Une modélisation géométrique itérative basée sur les automates. In *GTMG 2006, Journées du Groupe de Travail en Modélisation Géométrique, Cachan* (22-23 Mars 2006), pp. 155–169.

[ZT96] ZAIR C. E., TOSAN E. : Fractal modeling using free form techniques. *Comput. Graph. Forum. Vol. 15*, Num. 3 (1996), 269–278.

Intersection entre courbes et surfaces rationnelles au moyen des représentations implicites matricielles

Laurent Busé

INRIA Sophia Antipolis - Méditerranée
2004 routes des Lucioles, B.P. 93,
06902 Sophia Antipolis, France.

Résumé

Dans cet article, on introduit une nouvelle représentation implicite des courbes et des surfaces paramétrées rationelles, représentation qui consiste pour l'essentiel à les caractériser par la chute de rang d'une matrice plutôt que par l'annulation simultanée d'une ou plusieurs équations polynomiales. On montre comment ces représentations implicites, que l'on qualifiera de matricielles, établissent un pont entre la géométrie et l'algèbre linéaire, pont qui permet de livrer des problèmes géométriques à des algorithmes classiques et éprouvés d'algèbre linéaire, ouvrant ainsi la possibilité d'un traitement numérique plus robuste. La contribution de cette approche est discutée et illustrée sur des problèmes importants de la modélisation géométrique tels que la localisation (appartenance d'un point à un objet), le calcul d'intersection de deux objets, ou bien encore la détection d'un lieu singulier.

Mots-clés : Modélisation géométrique, courbes et surfaces paramétrées, problèmes d'intersection, réduction de pinceaux de matrices.

1. Introduction

En modélisation géométrique, les courbes et les surfaces paramétrées sont très utilisées. Pour les manipuler il est très avantageux d'en posséder une représentation implicite, en plus de leur représentation paramétrique. En effet, si les représentations paramétriques sont par exemple très utiles pour la visualisation de carreaux, les représentations implicites sont pour leur part d'un intérêt notoire dans les calculs d'intersection. Le but de cet article est de présenter une méthode simple qui permet, à partir d'une représentation paramétrique d'une courbe ou d'une surface, d'en produire une représentation implicite sous la forme d'une matrice, ce que nous nommerons une *représentation implicite matricielle.*

La représentation implicite des courbes et des surfaces paramétrées sous la forme d'une matrice a déjà été abordée dans littérature existante à de nombreuses reprises (voir par exemple [ACGS07, CGZ00, MC91, SC95]). Cependant, elle l'a toujours été dans le but d'écrire une équation implicite comme le déterminant d'une matrice (carrée). Le cas des courbes planes est bien connu, notamment car l'on sait toujours trouver de manière très simple une telle matrice carrée ; on pourra ici consulter l'article T. Sederberg et F. Chen [SC95] qui introduit cette technique pour les problèmes d'intersection entre courbes planes pour la modélisation géométrique. Le cas des surfaces paramétrées est beaucoup plus délicat, notamment dû au fait que la géométrie de leurs paramétrisations devient plus riche avec l'apparition inévitable des points de base (ce sont les points où une paramétrisation n'est pas bien définie). Afin de pouvoir trouver une matrice carrée dont le déterminant est une équation implicite, il faut se restreindre à des classes particulières de paramétrisations [CGZ00, BCD03, KD06], ce qui s'avère être très contraignant en pratique. Dans cet article, nous montrons comment, en se libérant de la contrainte d'une matrice carrée, on peut former très simplement une représentation implicite sous la forme d'une matrice pour une surface paramétrée très générale. La matrice en question n'est alors plus carrée, mais permet toujours de caractériser la surface : l'annulation d'un déterminant est ici remplacée par une propriété de chute de rang. De plus, le traitement des problèmes d'intersection peut être ramené à des calculs d'algèbre linéaire numérique, permettant ainsi l'exploitation d'outils robustes et performants pour le calcul approché, tels que la décompo-

sition en valeurs singulières et le calcul de valeurs et vecteurs propres généralisés.

Le présent article couvre une succession de travaux [BJ03, BC05, BCJ09] qui ont abouti à la notion de représentations implicites matricielles d'une courbe ou d'une surface paramétrée, ainsi qu'au développement de ses applications pour les problèmes d'intersection en modélisation géométrique [LBBM09, BLB10, BLB]. Nous commencerons par illustrer l'idée générale de l'approche développée dans cet article en s'intéressant au lancer de rayons sur une surface paramétrée, problème pour lequel il faut pouvoir résoudre l'intersection d'une droite et d'une surface paramétrée. Le reste de l'article expose la notion de représentation implicite matricielle, ainsi que ses applications aux problèmes d'intersection, pour le cas des surfaces paramétrées puis pour celui des courbes paramétrées.

Quelques précisions sur les notations et les usages dans ce qui suit. Une paramétrisation (d'une courbe ou d'une surface) est souvent donnée dans une notation affine, c'est-à-dire que les coordonnées de l'image sont des fractions rationnelles des paramètres. Il est cependant souvent plus simple de compactifier l'image de cette paramétrisation, tout comme son espace de paramètres. Plusieurs choix sont alors possibles ; dans ce qui suit, nous avons choisi de nous restreindre au cas des espaces projectifs classiques afin de ne pas gêner l'exposition par des considérations techniques qui sont secondaires dans les méthodes que nous présentons (nous traitons donc le cas des surfaces dites "triangulaires" ; voir [BD07] pour le cas des surfaces dites "tensor product", ce qui revient à paramétrer par un produit de deux droites projectives, ou bien plus généralement [BDD09a, BDD09b] pour des paramétrisations par des variétés toriques). Toujours par souci de simplicité, nous décrirons les polynômes (qui interviennent dans les paramétrisations) dans la base usuelle des monômes, plutôt que dans la base de Bernstein. Enfin, nous choisissons également de considérer des paramétrisations à coefficients des nombres réels $\mathbb{R}$, mais de nombreuses propriétés peuvent s'énoncer dans un cadre plus général (en particulier pas nécessairement sur un corps). Chaque fois qu'une étape de résolution nécessite le passage à la clotûre algébrique, nous pourrons ainsi le souligner en travaillant dans le corps des nombres complexes $\mathbb{C}$.

2. Une motivation : le lancer de rayons

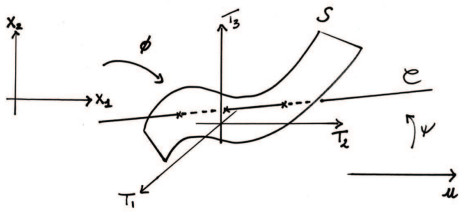
La motivation principale de ce travail est d'améliorer le traitement des problèmes d'intersection entre courbes et surfaces paramétrées. La lancer de rayons est une application possible pour laquelle on peut bien illustrer l'objectif poursuivi. En effet, cette technique nécessite d'intersecter de manière intensive un rayon, c'est-à-dire une droite paramétrée, avec un objet dont le bord est représenté par une surface paramétrée. Nous noterons

$$\begin{array}{rcl} \mathbb{R}^2 & \xrightarrow{\phi} & \mathbb{R}^3 \\ (X_1,X_2) & \mapsto & \left(\frac{f_1}{f_4},\frac{f_2}{f_4},\frac{f_3}{f_4}\right)(X_1,X_2) \end{array}$$

où f_1,f_2,f_3,f_4 sont des polynômes en X_1,X_2, une paramétrisation de notre surface $\mathcal{S}$ et

$$\begin{array}{rcl} \mathbb{R}^1 & \xrightarrow{\psi} & \mathbb{R}^3 \\ u & \mapsto & O+u.\vec{d} \end{array}$$

où O désigne un point de $\mathbb{R}^3$ et $\vec{d}$ un vecteur direction, une paramétrisation de notre rayon (i.e. une droite) $\mathcal{C}$. Afin de



déterminer l'intersection entre le rayon $\mathcal{C}$ et la surface $\mathcal{S}$, deux stratégies classiques peuvent être utilisées. La première consiste à projeter les points qui nous intéressent sur $\mathbb{R}^2$, l'espace des paramètres de $\mathcal{S}$. Pour cela, on calcule une représentation implicite de la droite $\mathcal{C}$, c'est-à-dire un système de deux équations de plans dans $\mathbb{R}^3$ dont l'intersection est notre rayon :

$$a_1T_1+a_2T_2+a_3T_3+a_4 = b_1T_1+b_2T_2+b_3T_3+b_4 = 0.$$

Dès lors, en substituant dans ce système les variables T_1,T_2 et T_3 par la paramétrisation ϕ, c'est-à-dire par $f_1/f_4, f_2/f_4$ et f_3/f_4 respectivement, on obtient, après mise au même dénominateur, un système de deux équations algébriques en X_1,X_2 dont les solutions sont les paramètres des points de $\mathcal{C}\cap\mathcal{S}$ au travers de la paramétrisation ϕ. Il faut alors utiliser des algorithmes de résolution pour un tel système, par exemple un algorithme de type "Newton" est dans ce cas très classique.

La deuxième stratégie possible est de projeter non plus sur $\mathbb{R}^2$, mais sur $\mathbb{R}^1$, c'est-à-dire sur l'espace des paramètres de la droite $\mathcal{C}$. Pour cela, ce n'est plus la droite dont il nous faut donner une représentation implicite, mais la surface $\mathcal{S}$. En effet, supposons que nous connaissions une équation implicite $S(T_1,T_2,T_3)=0$ de $\mathcal{S}$, alors la substitution de la paramétrisation de la droite $\mathcal{C}$ dans cette équation fournirait un polynôme en la variable u dont les racines seraient les paramètres des points de $\mathcal{C}\cap\mathcal{S}$ au travers de la paramétrisation ψ.

En résumé, on voit bien que dans la première stratégie l'étape de changement de représentation est simple alors que la résolution du système obtenu peut s'avérer délicate et difficile à contrôler, notamment en termes de robustesse et de précision. Dans la deuxième stratégie, c'est l'étape de

changement de représentation qui est difficile, aussi bien en termes de mise en oeuvre d'un algorithme qu'en coût de calculs nécessaires, alors que l'étape de résolution est plutôt simple et il existe de nombreux algorithmes permettant de la traiter de manière robuste, précise et efficace.

Dans ce qui suit, nous introduisons un nouveau concept de représentation implicite pour la surface $\mathcal{S}$ afin de palier à la difficulté du calcul de l'équation implicite $S(T_1,T_2,T_3)=0$. Cette nouvelle représentation de $\mathcal{S}$ prend une forme matricielle et permet de manipuler implicitement $\mathcal{S}$ sans jamais avoir à calculer son équation implicite. Sa nature matricielle permet de ramener toutes les questions de type "intersection" à des calculs d'algèbre linéaire numérique dont la robustesse et la précision ont été très largement étudiées et développées. Aussi, il est important de souligner qu'aucune arithmétique sur les polynômes n'est nécessaire dans notre approche.

3. Représentations implicites matricielles pour les surfaces paramétrées

Étant donnée une surface représentée par une de ses paramétrisations rationnelles, nous décrivons ici une nouvelle représentation implicite pour cette surface ainsi que ses principales propriétés. Afin de simplifier cette présentation, il est préférable d'adopter des coordonnées projectives. Ainsi, nous supposerons dans la suite que $\mathcal{S}$ est une surface de $\mathbb{P}^3_{\mathbb{R}}$ (l'espace projective de dimension 3 sur $\mathbb{R}$) dont une paramétrisation rationnelle est

$$\begin{array}{ccc} \mathbb{P}^2_{\mathbb{R}} & \xrightarrow{\phi} & \mathbb{P}^3_{\mathbb{R}} \\ (X_1:X_2:X_3) & \mapsto & (f_1:f_2:f_3:f_4)(X_1,X_2,X_3) \end{array} \tag{1}$$

où f_1,f_2,f_3 et f_4 sont maintenant des polynômes homogènes en les variables X_1,X_2 et X_3 de même degré $d\geq 1$. En outre, nous supposerons que ces polynômes n'ont pas de facteur commun, situation à laquelle il est toujours possible de se ramener facilement en opérant une simplification après calcul d'un plus grand diviseur commun.

3.1. Construction d'une famille de matrices

Introduisant les variables T_1,T_2,T_3,T_4 pour désigner les coordonnées homogènes de l'espace projectif $\mathbb{P}^3$, on s'intéresse aux relations de degré ν, pour ν un entier naturel, des polynômes f_1,f_2,f_3,f_4, c'est-à-dire aux polynômes

$$\sum_{i=1}^{4} g_i(X_1,X_2,X_3)T_i \in \mathbb{R}[X_1,X_2,X_3][T_1,T_2,T_3,T_4]$$

tels que g_1,g_2,g_3 et g_4 sont des polynômes homogènes de degré ν qui satisfont à la propriété

$$\sum_{i=1}^{4} g_i(X_1,X_2,X_3)f_i(X_1,X_2,X_3)\equiv 0. \tag{2}$$

On note $\mathcal{R}_\nu$ l'ensemble de toutes ces relations. Il est important d'observer que $\mathcal{R}_\nu$ est un $\mathbb{R}$-espace vectoriel de dimension finie. De plus une base de $\mathcal{R}_\nu$ s'obtient par la résolution du système linéaire induit par (2) que l'on écrit dans la base des monômes de degré $\nu+d$ en les variables X_1,X_2,X_3.

Partant de là, pour tout entier $\nu\geq 0$, on définit la matrice $\mathbb{M}(\phi)_\nu$ par l'égalité matricielle

$$\begin{pmatrix} X_1^\nu & X_1^{\nu-1}X_2 & X_1^{\nu-1}X_3 & \cdots & X_3^\nu \end{pmatrix}\mathbb{M}(\phi)_\nu = \begin{pmatrix} L^{(1)} & L^{(2)} & \cdots & L^{(n_\nu)} \end{pmatrix}$$

où $L^{(1)},\ldots,L^{(n_\nu)}$ est une base du $\mathbb{R}$-espace vectoriel $\mathcal{R}_\nu$. En d'autres termes, $\mathbb{M}(\phi)_\nu$ est la matrice dont les colonnes sont formées par les coefficients de chaque relation $L^{(j)}$, $j=1,\ldots,n_\nu$, dans la base des monômes en les variables X_1,X_2,X_3 de degré ν.

Example 1 Considérons la paramétrisation de la sphère de la forme (1) où

$$f_1=X_1^2-X_2^2-X_3^2,\ f_2=2X_1X_3,$$

$$f_3=2X_1X_2,\ f_4=X_1^2+X_2^2+X_3^2.$$

Il est évident de constater que la matrice $\mathbb{M}(\phi)_0$ est la matrice nulle. Un simple calcul fournit

$$\mathbb{M}(\phi)_1=\begin{pmatrix} -T_2 & -T_3 & -T_1+T_4 & 0 \\ 0 & T_1+T_4 & -T_3 & -T_2 \\ T_1+T_4 & 0 & -T_2 & T_3 \end{pmatrix}.$$

On pourrait de même former $\mathbb{M}(\phi)_2$, $\mathbb{M}(\phi)_3$, etc.

3.2. Représentation implicite matricielle

La famille de matrices que nous venons de construire est particulièrement intéressante car à partir d'un certain degré ν_0, toutes ces matrices peuvent être qualifiées de représentations implicites de la surface $\mathcal{S}$. En effet, pour tout entier $\nu\geq\nu_0$ la matrice $\mathbb{M}(\phi)_\nu$ vérifie les propriétés suivantes :

- elle est formée de $C^2_{\nu+2}=\frac{(\nu+2)(\nu+1)}{2}$ lignes et d'au moins autant de colonnes (en général strictement plus),
- ses entrées sont des formes linéaires en T_1,T_2,T_3,T_4 à coefficients dans $\mathbb{R}$,
- évaluée en un point $P\in\mathbb{P}^3\setminus\mathcal{S}$, elle est de rang maximum égal à $C^2_{\nu+2}$,
- évaluée en un point $P\in\mathcal{S}\subset\mathbb{P}^3$, elle est de rang strictement plus petit que $C^2_{\nu+2}$.

Ces propriétés justifient la qualification de "représentation implicite" pour ces matrices puisqu'il apparaît qu'elles permettent de caractériser la surface $\mathcal{S}$ par une chute de rang, de manière similaire au fait qu'une équation implicite $S(T_1,T_2,T_3,T_4)=0$ de $\mathcal{S}$ la caractérise par le fait de s'annuler ou non.

D'un point de vue pratique, c'est la matrice $\mathbb{M}(\phi)_{\nu_0}$ qu'il faut privilégier car c'est celle de plus petite taille. L'entier ν_0 est très simple à contrôler : on peut toujours choisir $\nu_0=2(d-1)$ et lorsque les polynômes f_1,f_2,f_3 et f_4

possèdent au moins une racine commune dans $\mathbb{P}^2_{\mathbb{C}}$ (on parle de point de base) alors on peut prendre $\nu_0 = 2(d-1)-1$. Ainsi, la matrice donnée dans l'exemple 1 est une matrice de représentation de la sphère.

Les résultats que nous venons d'énoncer sont démontrés dans [BJ03, BC05]. Ils sont valables sous une hypothèse très faible, mais technique, qui demande que le système d'équations $f_1 = f_2 = f_3 = f_4 = 0$ de $\mathbb{P}^2$ soit localement défini par deux équations au voisinage de chacune de ses solutions, s'il en existe. Le lecteur peut consulter *loc. cit.* pour plus de détails.

Ces représentations implicites matricielles s'avèrent être relativement souples. En effet, une fois une matrice de représentation formée, notons la $\mathbb{M}$, tester si un point P donné appartient à la surface $\mathcal{S}$ (c'est-à-dire détecter une intersection point/surface) se fait en calculant le rang de $\mathbb{M}(P)$ qui est une matrice à coefficients dans $\mathbb{R}$ (la notation $\mathbb{M}(P)$ signifie que l'on a évalué la matrice $\mathbb{M}$ au point P, coefficient par coefficient). On peut donc ici utiliser l'outil très puissant de l'algèbre linéaire numérique qu'est la décomposition en valeurs singulières (DVS). En effet, une DVS de la matrice $\mathbb{M}$ permet de quantifier (par la détermination d'un saut dans les rapports successifs des valeurs singulières) si le point P est sur la surface $\mathcal{S}$ à une précision donnée ; on parle de rang numérique. Pour faire le parallèle avec une équation implicite $S = 0$ de la surface, cela correspondrait à voir si $\|S(P)\| < \varepsilon$ pour un epsilon petit donné.

En termes de représentation machine, il est important de noter que ces représentations matricielles correspondent simplement à la donnée de 4 matrices $\mathbb{M}_1, \mathbb{M}_2, \mathbb{M}_3, \mathbb{M}_4$ à coefficients dans $\mathbb{R}$ puisque

$$\mathbb{M} = \mathbb{M}_1 T_1 + \mathbb{M}_2 T_2 + \mathbb{M}_3 T_3 + \mathbb{M}_4 T_4.$$

Ainsi, la manipulation de cette représentation implicite se fait sans aucun usage d'une arithmétique de polynômes ; il suffit de posséder des opérations élémentaires sur les matrices à coefficients des flottants. Par exemple, une évaluation d'une matrice de représentation en un point $P \in \mathbb{P}^3$ correspond à 4 multiplications scalaire/matrice et 3 additions matrice/matrice.

3.3. Inversion d'un point

Supposons donnée une surface $\mathcal{S}$ paramétrée par (1) et un point $P = (P_1 : P_2 : P_3 : P_4) \in \mathbb{P}^3_{\mathbb{R}}$. Lorsque ce point appartient à $\mathcal{S}$ et possède un unique antécédent $(x_1 : x_2 : x_3)$ par la paramétrisation ϕ, alors il est important de savoir déterminer $(x_1 : x_2 : x_3)$ à partir de $P = (P_1 : P_2 : P_3 : P_4)$ et ϕ. L'utilisation d'une matrice de représentation $\mathbb{M}(\phi)_\nu$, pour tout $\nu \geq \nu_0$, rend cette opération très simple. En effet, par construction même de cette matrice, on a l'égalité matricielle

$$\begin{pmatrix} x_1^\nu & x_1^{\nu-1}x_2 & x_1^{\nu-1}x_3 & \cdots & x_3^\nu \end{pmatrix} \times \mathbb{M}(\phi)_\nu(P) = 0.$$

De plus, on montre que le noyau de la transposée de la matrice $\mathbb{M}(\phi)_\nu(P)$ est de dimension 1, sous l'hypothèse naturelle que P possède une unique antécédent (noter qu'en dehors de cette hypothèse, la question même d'inversion est mal définie). Il s'en suit que tout vecteur non nul dans le noyau de la transposée de $\mathbb{M}(\phi)_\nu(P)$ est de la forme

$$c.\begin{pmatrix} x_1^\nu & x_1^{\nu-1}x_2 & x_1^{\nu-1}x_3 & \cdots & x_3^\nu \end{pmatrix}$$

où c est une constante non nulle. On peut dès lors en extraire le point $(x_1 : x_2 : x_3)$ comme suit : une des coordonnées homogènes x_1, x_2, x_3 est non nulle, disons x_1 par exemple. Alors, notant $V = (v_1, v_2, v_3, \ldots)$ un vecteur non nul du noyau de la transposée de $\mathbb{M}(\phi)_\nu(P)$, il est immédiat de constater que $(x_1 : x_2 : x_3) = (v_1 : v_2 : v_3)$.

D'un point de vue du calcul, il est important d'observer que cette technique est particulièrement bien adaptée aux outils de l'algèbre linéaire numérique. En effet, calculons une DVS de la matrice $\mathbb{M}(\phi)_\nu(P)$:

$$\mathbb{M}(\phi)_\nu(P) = U\Sigma V.$$

La matrice Σ est diagonale et contient les valeurs singulières qui nous permettent de déterminer si P appartient à la surface $\mathcal{S}$, comme nous l'avons signalé dans le paragraphe précédent. Si la réponse est positive, c'est-à-dire si le rang numérique de cette matrice chute, alors la dernière ligne de la matrice U est un élément du noyau (approché) qui permet de calculer l'inversion du point P comme décrit ci-dessus. Noter que, par propriété de la DVS, cet élément est la meilleure approximation du noyau de $\mathbb{M}(\phi)_\nu(P)$ à une précision donnée et qu'aucun calcul supplémentaire n'est nécessaire par rapport au test d'appartenance de P à la surface $\mathcal{S}$. La robustesse et la précision sont ici contrôlable par héritage de la DVS [GVL96].

Revenant à notre illustration du lancer de rayons, on peut ici souligner l'importance de l'inversion dans notre approche consistant à projeter l'intersection d'un rayon avec la surface sur le paramètre du rayon. En effet, lorsqu'un point d'intersection est déterminé, il est nécessaire de calculer la normale à la surface en ce point afin de pouvoir réfléchir ou réfracter la lumière. Ne disposant pas d'une équation implicite de $\mathcal{S}$, c'est la paramétrisation ϕ qui permet de trouver cette normale. Le paramètre correspondant est alors obtenu directement pendant le calcul d'intersection, par inversion, et l'on peut en déduire la normale.

Enfin, remarquons que ce processus d'inversion est impossible avec une équation implicite classique $S(T_1, T_2, T_3, T_4) = 0$ car, contrairement à notre représentation implicite matricielle, celle-ci est complètement décorrélée de la paramétrisation ϕ.

4. Intersection entre une courbe et une surface paramétrées

En plus de notre surface $\mathcal{S}$ paramétrée par (1), on suppose à présent donnée une courbe $\mathcal{C}$ non contenue dans $\mathcal{S}$ et

paramétrée en coordonnées homogènes par

$$\begin{array}{rcl}\Psi : \mathbb{P}^1_{\mathbb{R}} & \to & \mathbb{P}^3_{\mathbb{R}} \\ (u:v) & \mapsto & (x(u,v):y(u,v):z(u,v):w(u,v)).\end{array}$$

Les polynômes $x(u,v), y(u,v), z(u,v), w(u,v)$ sont des polynômes homogènes dans $\mathbb{R}[u,v]$ sans facteur commun et de même degré. L'ensemble des points d'intersection $\mathcal{C} \cap \mathcal{S}$ est fini et notre objectif est de le décrire explicitement.

Si l'on dispose d'une équation implicite $S(T_1,T_2,T_3,T_4) = 0$ de $\mathcal{S}$, la détermination de l'ensemble d'intersection $\mathcal{C} \cap \mathcal{S}$ peut se faire en calculant, de manière approchée, les racines du polynôme homogène

$$S(x(u,v),y(u,v),z(u,v),w(u,v)) \in \mathbb{R}[u,v]$$

puisque ces dernières sont en correspondance avec l'ensemble $\mathcal{C} \cap \mathcal{S}$ par Ψ. Dans ce qui suit, nous présentons une méthode alternative basée sur l'utilisation d'une matrice de représentation de ϕ qui permet d'obtenir les points de $\mathcal{C} \cap \mathcal{S}$ non seulement dans l'espace des paramètres de $\mathcal{C}$, mais également dans l'espace des paramètres de $\mathcal{S}$ par les résultats du paragraphe précédent.

4.1. Formulation matricielle

Soit $\mathbb{M}(\phi)_{\nu_0}$ une matrice de représentation de ϕ (nous choisissons ici la plus petite matrice). En substituant les polynômes $x(u,v)$, $y(u,v)$, $z(u,v)$, $w(u,v)$ aux variables T_1,T_2,T_3,T_4 dans cette matrice, nous obtenons une nouvelle matrice, que nous noterons

$$\mathbb{M}(u,v) := \mathbb{M}(\phi)_{\nu_0}(x(u,v),y(u,v),z(u,v),w(u,v))$$

et dont les entrées sont des polynômes homogènes dans $\mathbb{R}[u,v]$. Par propriété d'une matrice de représentation, nous avons que pour tout point $(u_0:v_0) \in \mathbb{P}^1_{\mathbb{C}}$, le rang de la matrice $\mathbb{M}(u_0,v_0)$ chute si et seulement si le point $\Psi(u_0,v_0)$ appartient à l'ensemble $\mathcal{C} \cap \mathcal{S}$. Par conséquent, l'ensemble $\mathcal{C} \cap \mathcal{S}$ est en correspondance avec les points de $\mathbb{P}^1_{\mathbb{C}}$ où le rang de la matrice $\mathbb{M}(u,v)$ chute, c'est-à-dire où le rang de $\mathbb{M}(u,v)$ est strictement inférieur à son nombre de lignes $C^2_{\nu_0+2}$. Nous présentons maintenant des techniques d'algèbre linéaire qui vont nous permettre de déterminer ces points par un calcul de valeurs (et vecteurs) propres généralisées [LBBM09].

4.2. Linéarisation d'une matrice polynomiale

Étant donnée une matrice $\mathbb{N}(t) = (a_{i,j}(t))$ de taille $m \times n$ à coefficients dans $\mathbb{R}[t]$, on peut l'écrire sous la forme

$$\mathbb{N}(t) = \mathbb{N}_d t^d + \mathbb{N}_{d-1} t^{d-1} + \ldots + \mathbb{N}_0$$

où $d = \max_{i,j}\{\deg(a_{i,j}(t))\}$ et $\mathbb{N}_i$, $i = 1,\ldots,d$, est une matrice de taille $m \times n$ à coefficients dans $\mathbb{R}$. On définit alors les *matrices compagnes généralisées*, que nous noterons A et B, par

$$A = \begin{pmatrix} 0 & \mathbb{I}_m & 0 & \cdots & 0 \\ 0 & 0 & \mathbb{I}_m & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \mathbb{I}_m \\ {}^t\mathbb{N}_0 & {}^t\mathbb{N}_1 & \cdots & {}^t\mathbb{N}_{d-2} & {}^t\mathbb{N}_{d-1} \end{pmatrix},$$

$$B = \begin{pmatrix} \mathbb{I}_m & 0 & 0 & \cdots & 0 \\ 0 & \mathbb{I}_m & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbb{I}_m & 0 \\ 0 & \cdots & 0 & 0 & -{}^t\mathbb{N}_d \end{pmatrix}$$

où $\mathbb{I}_r$ désigne la matrice identité de taille r et où la notation ${}^t-$ désigne l'opération de transposition d'une matrice. Ce sont deux matrices de taille $((d-1)m+n) \times dm$ à coefficients dans $\mathbb{R}$. Elles permettent de *linéariser* la matrice polynomiale $M(t)$ au sens où il existe deux matrices unimodulaires $E(t)$ et $F(t)$ à coefficients dans $\mathbb{R}[t]$ et de taille respective dm et $(d-1)m+n$ telles que

$$E(t)\,(A - tB)\,F(t) = \left(\begin{array}{c|c} {}^t\mathbb{N}(t) & 0 \\ \hline 0 & \mathbb{I}_{d(m-1)} \end{array}\right). \qquad (3)$$

Ainsi, les valeurs de $t \in \mathbb{C}$ pour lesquelles le rang de $\mathbb{N}(t)$ chute sont en correspondance avec les valeurs de $t \in \mathbb{C}$ pour lesquelles le rang de $A - tB$ chute. On les appelle les *valeurs propres généralisées du pinceau de matrices* $A - tB$. Si les matrices A et B sont des matrices carrées et que B est une matrice inversible alors les valeurs propres généralisées du pinceau $A - tB$ se calculent directement à l'aide de l'algorithme dit "QZ" [GVL96]. Dans le cas contraire, il faut réduire le pinceau ; c'est l'objectif du paragraphe suivant.

4.3. Extraction de la partie régulière d'un pinceau de matrices

Pour tout pinceau de matrices $A - tB$, il existe des matrices constantes et inversibles P et Q telles que le pinceau de matrices $P(A - tB)Q = PAQ - tPBQ$ est diagonal par blocs de la forme

$$\mathrm{diag}\{L_{i_1},\ldots,L_{i_s},L^t_{j_1},\ldots,L^t_{j_u},\Omega_{k_1},\ldots,\Omega_{k_v},A' - tB'\} \qquad (4)$$

où les matrices A' et B' sont carrées, la matrice B' est inversible et où pour tout entier $k \geq 1$ les matrices $L_k(t)$, de taille $k \times (k+1)$, et $\Omega_k(t)$, de taille $k \times k$, sont définies par

$$L_k(t) = \begin{pmatrix} 1 & t & 0 & \ldots & 0 \\ 0 & 1 & t & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 1 & t & 0 \\ 0 & 0 & \ldots & 1 & t \end{pmatrix},$$

$$\Omega_k(t) = \begin{pmatrix} 1 & t & 0 & \dots & 0 \\ 0 & 1 & t & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 1 & t \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

On parle de la *forme de Kronecker* du pinceau $A - tB$ (voir par exemple [Gan66, p. 31-34]).

Le bloc $A' - tB'$ est la partie régulière du pinceau $A - tB$. Dans notre contexte du problème d'intersection, il est très intéressant car il contient toute l'information sur les valeurs propres généralisées (à distance finie) du pinceau $A - tB$ et car l'on sait calculer ces valeurs propres généralisées par un algorithme QZ puisque c'est un bloc carré et que B' est inversible. Nous présentons ici un algorithme permettant d'extraire la partie régulière $A' - tB'$ d'un pinceau de matrice $A - tB$ et renvoyons le lecteur à [LBBM09] pour plus de détails.

L'extraction de la partie régulière $A' - tB'$ du pinceau de matrice $A - tB$ est basée sur des réductions que l'on obtient aisément par des calculs de DVS. C'est un extraction itérative. Partant du pinceau $A - tB$, on procède comme suit :

– On calcule une forme échelon par colonne de B (par exemple en calculant une DVS de B)

$$B.{}^tV = (*|0)$$

– On applique cette transformation à la matrice A :

$$A.{}^tV = (*|A_1)$$

– On calcule une forme échelon par lignes de la matrice A_1 (par exemple en calculant une DVS de A_1) :

$${}^tU.A_1 = \begin{pmatrix} * \\ \hline 0 \end{pmatrix}$$

À la fin de cette première itération, on obtient

$${}^tU(A - tB)^tV = \left(\begin{array}{c|c} * & * \\ \hline A' - tB' & 0 \end{array}\right)$$

et on peut donc réduire le pinceau $A - tB$ en le pinceau $A' - tB'$ puisque le rang de la partie supérieure de cette matrice est indépendant de t. On recommence alors la procédure jusqu'au moment où cette partie supérieure n'apparaît plus. Il convient alors d'opérer la même réduction sur la transposée du pinceau $A' - tB'$. On aboutit ainsi à un pinceau carré qui est la partie régulière du pinceau de départ $A - tB$.

4.4. Algorithme pour l'intersection courbe/surface

Nous avons maintenant tous les éléments pour énoncer un algorithme permettant de calculer, de manière approchée et robuste, l'intersection entre une surface rationnelle et une courbe rationnelle (lorsque cette dernière n'est pas contenue dans la première).

Les données en entrée sont notre surface $\mathcal{S}$ paramétrée par ϕ, ainsi qu'une matrice de représentation $\mathbb{M}(T_1, \dots, T_4)$, et notre courbe $\mathcal{C}$ paramétrée par Ψ. On procède alors de la façon suivante :

1. On forme la matrice $\mathbb{M}(\Psi(u,1))$.
2. On forme les matrices compagnes A, B de la matrice $\mathbb{M}(\Psi(u,1))$.
3. On extrait la partie régulière $A' - uB'$ du pinceau $A - uB$.
4. On calcule l'ensemble des valeurs propres (généralisées) $\{u_1, \dots, u_r\}$ du pinceau $A' - uB'$.
5. On renvoie l'ensemble de points $\{\Psi(u_1), \dots, \Psi(u_r)\}$ qui correspond aux points d'intersection de $\mathcal{S} \cap \mathcal{C}$ (à l'exception du point $\Psi(1:0)$ qui pourrait éventuellement appartenir à $\mathcal{S}$).

Seules les étapes 3 et 4 nécessitent des calculs, calculs qui sont basés sur des algorithmes éprouvés de l'algèbre linéaire numérique et qui permettent d'assurer une robustesse importante aux points d'intersection calculés.

Afin d'illustrer cet algorithme sur un exemple élémentaire, revenons à notre première motivation : le lancer de rayons. Pour cela, reprenons la sphère et sa matrice de représentation $\mathbb{M}(\phi)_1$ donnée dans l'exemple 1 dont nous précisons ici que ces lignes sont indexées par les monômes X_1, X_2, X_3 (dans cet ordre). On choisit de l'intersecter avec la droite de $\mathbb{R}^3$ passant par le point $(1/2, 1/2, 1/2)$ et de vecteur directeur $(1,1,1)$. On substitue donc dans la matrice $\mathbb{M}(\phi)_1$, les variables T_1, T_2 et T_3 par $1/2 + u$ et T_4 par 1. On obtient la matrice

$$\mathbb{M}(u) := \begin{pmatrix} -\frac{1}{2} - t & -\frac{1}{2} - t & \frac{1}{2} - t & 0 \\ 0 & \frac{3}{2} + t & -\frac{1}{2} - t & -\frac{1}{2} - t \\ \frac{3}{2} + t & 0 & -\frac{1}{2} - t & \frac{1}{2} + t \end{pmatrix}.$$

Puisque nous intersectons avec une droite, le procédé de linéarisation est trivial et on obtient

$$A := \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{3}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{3}{2} & 0 & -\frac{1}{2} & \frac{1}{2} \end{pmatrix},$$

$$B := \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & -1 \end{pmatrix},$$

de telle sorte que $\mathbb{M}(u) = A - uB$. Le calcul de la partie régulière du pinceau (A, B) fournit le pinceau (A', B') donné par

$$A' := \begin{pmatrix} -\frac{1}{6} & \frac{1}{3} \\ \frac{2}{3} & -\frac{5}{6} \end{pmatrix}, \quad B' := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Le calcul des valeurs propres renvoit le vecteur (exceptionnellement exact dans notre exemple très simple)

$$(u_1, u_2) := (-1/2 + 1/3\sqrt{3}, -1/2 - 1/3\sqrt{3}).$$

De là on obtient les points d'intersection entre la droite et la sphère :

$$P_1 := (\frac{1}{2} + u_1, \frac{1}{2} + u_1, \frac{1}{2} + u_1), P_2 := (\frac{1}{2} + u_2, \frac{1}{2} + u_2, \frac{1}{2} + u_2).$$

Nous souhaitons à présent obtenir la normale à la sphère au point d'intersection P_1. Pour ce faire, nous inversons le point P_1 par ϕ : on forme tout d'abord la matrice de représentation de la sphère au point P_1

$$\mathbb{M}(\phi)_1(P_1) = \mathbb{M}(u_1)$$

puis on calcule le noyau de sa transposée. Le calcul donne le vecteur

$$(1+\sqrt{3}, 1, 1)$$

dont on déduit que la pré-image de P_1 par ϕ est le point à distance finie $(x_1, x_2) = (1+\sqrt{3}, 1)$. Partant de là, on déduit facilement la normale à la sphère au point P_1 à l'aide de sa paramétrisation ϕ.

4.5. Extension à l'intersection surface/surface

Le problème de l'intersection entre deux surfaces paramétrées est un problème classique de la modélisation géométrique. Cependant, cette intersection est en général une courbe et pas un ensemble fini de points comme ce que nous avons vu jusqu'à présent (et verrons à nouveau par la suite). L'étude de ce lieu d'intersection requiert donc d'anticiper sur la façon dont on va le représenter, d'autant plus que ce choix de représentation, contrairement au cas des ensembles finis de points, est multiple et pas toujours naturel. En fait, c'est plutôt l'information que l'on souhaite extraire ou utiliser de ce lieu d'auto-intersection qui doit guider ce choix.

Dans ce court paragraphe, nous choisissons l'approche introduite par J. Canny et D. Manocha qui consiste à représenter la courbe d'intersection entre deux surfaces comme une courbe paramétrée par une autre courbe plane qui est définie dans l'espace des paramètres d'une des deux surfaces. On peut la résumer comme suit :

- On projette la courbe intersection sur l'espace des paramètres d'une des deux surfaces. On obtient ainsi une courbe plane qui est représentée par une équation implicite.
- On détermine alors la topologie exacte de cette courbe (présence d'ovales, singularités, etc.).
- On en déduit alors une bonne représentation de l'intersection au travers de la paramétrisation de la surface.

La figure 1 ci-contre illustre ce procédé ; on a ici dessiné les deux courbes possibles, suivant que l'on choisisse de projeter sur l'espace des paramètres d'une surface ou bien de l'autre.

Comme décrite dans [MC91], cette approche est tout fait pertinente car elle permet de ramener des considérations géométriques à des calculs d'algèbre linéaire numérique. Cependant, elle souffre d'une restriction extrêmement forte qui impose de pouvoir représenter implicitement une des deux surfaces comme le déterminant d'une "matrice résultante" (matrice nécessairement carrée). En particulier, il faut demander que la paramétrisation de la surface en question ne

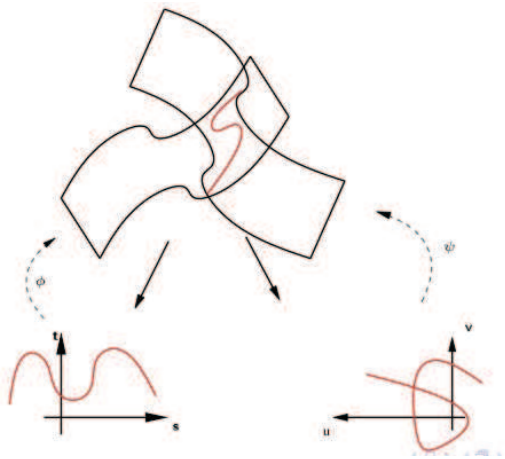


Figure 1: *Projections d'une courbe d'intersection*

possède pas de points de base. Il se trouve que l'utilisation des représentations implicites matricielles que nous avons introduites permet de lever cette restriction. On procède de la façon suivante.

Notons $\mathbb{M}(T_1, T_2, T_3, T_4)$ une matrice de représentation d'une des deux surfaces, disons $\mathcal{S}_1$. En substituant la paramétrisation $\phi(X_1 : X_2 : 1)$ de l'autre surface, disons $\mathcal{S}_2$, aux variables T_1, T_2, T_3, T_4 respectivement dans cette paramétrisation, on obtient une matrice polynomiale bivariée $\mathbb{M}(X_1, X_2)$. À ce stade, le lieu qui nous intéresse est, par propriété des représentations implicites matricielles,

$$\{(X_1, X_2) \in \mathbb{C}^2 \text{ tel que } \text{rang}(\mathbb{M}(X_1, X_2)) \text{ chute}\}. \quad (5)$$

On montre [BLB] que *ce lieu est en fait une courbe algébrique plane de* $\mathbb{C}^2$ (qui passe par tous les points de base de la paramétrisation ϕ), en particulier, il ne possède pas de point isolé. Cette propriété permet alors d'exprimer le lieu (5) par une équation implicite, comme dans l'approche Canny-Manocha, cela afin de pouvoir déterminer la topologie de ce lieu. Cette réduction est assez technique et nous renvoyons le lecteur à l'article [BLB] pour les détails. Elle consiste pour l'essentiel à linéariser la matrice $\mathbb{M}(X_1, X_2)$ en un pinceau bivarié $A + X_1 B + X_2 C$, où A, B, C sont des matrices à coefficients dans $\mathbb{R}$, puis d'adapter un algorithme de réduction de pinceau, appelé décomposition $\Delta W - 1$ et dû à V. Kublanovskaia [Kub99, KK96], pour arriver à un pinceau carré dont le déterminant fournit l'équation recherchée.

Avant de clore ce paragraphe, mentionnons une application particulièrement intéressante de cette approche du problème d'intersection entre deux surfaces paramétrées : la représentation du lieu d'auto-intersection d'une surface paramétrée (là encore, précisons que la façon de représenter l'auto-intersection d'une surface dépend fortement de l'utilisation poursuivie). En effet, rien n'empêche dans la méthode décrite ci-dessus de considérer que les deux surfaces $\mathcal{S}_1$ et

$\mathcal{S}_2$ sont identiques, ainsi que leur paramétrisation. Le lieu (5) décrit alors une courbe dans l'espace des paramètres de cette surface que est en correspondance avec le lieu d'auto-intersection au travers de sa paramétrisation. Nous reviendrons de manière un peu plus précise sur ce procédé dans le paragraphe 5.3.

5. Représentations implicites matricielles pour les courbes paramétrées

Le concept de représentation implicite matricielle que nous avons introduit pour les surfaces paramétrées est en fait plus général et peut s'appliquer à divers types d'objets paramétrés. Dans le cadre de la modélisation géométrique, les courbes paramétrées sont intensivement utilisées, notamment par le fait que de nombreuses surfaces paramétrées sont définies comme des familles de courbes paramétrées. Dans ce qui suit, nous introduisons donc le concept de représentation implicite matricielle pour une courbe $\mathcal{D}$ donnée par une paramétrisation en coordonnées homogènes

$$\begin{array}{rcl} \phi : \mathbb{P}^1_{\mathbb{R}} & \to & \mathbb{P}^3_{\mathbb{R}} \\ (s:t) & \mapsto & (f_1(s,t) : f_2(s,t) : f_3(s,t) : f_4(s,t)). \end{array} \quad (6)$$

Les polynômes $f_1(s,t), f_2(s,t), f_3(s,t), f_4(s,t)$ sont des polynômes homogènes dans $\mathbb{R}[s,t]$ sans facteur commun et de même degré $d \geq 1$.

5.1. Définition et propriétés

La construction d'une représentation implicite matricielle pour une courbe paramétrée est en tout point semblable à celle pour une surface paramétrée que nous avons détaillée précédemment. Notant toujours T_1, T_2, T_3, T_4 les coordonnées homogènes de l'espace projectif $\mathbb{P}^3$, on s'intéresse aux relations de degré ν, pour ν un entier naturel, des polynômes f_1, f_2, f_3, f_4, c'est-à-dire aux polynômes

$$\sum_{i=1}^{4} g_i(s,t) T_i \in \mathbb{R}[s,t][T_1, T_2, T_3, T_4]$$

tels que g_1, g_2, g_3 et g_4 sont des polynômes homogènes de degré ν qui satisfont à la propriété

$$\sum_{i=1}^{4} g_i(s,t) f_i(s,t) \equiv 0. \quad (7)$$

On note $\mathcal{R}_\nu$ l'ensemble de toutes ces relations qui est un $\mathbb{R}$-espace vectoriel de dimension finie. Une base de $\mathcal{R}_\nu$ s'obtient en résolvant le système linéaire induit par (7) que l'on écrit dans la base des monômes de degré $\nu + d$ en les variables s,t.

Pour tout entier $\nu \geq 0$, on définit ainsi la matrice $\mathbb{M}(\phi)_\nu$ par l'égalité

$$\begin{pmatrix} s^\nu & s^{\nu-1}t & s^{\nu-2}t^2 & \cdots & t^\nu \end{pmatrix} \mathbb{M}(\phi)_\nu = \begin{pmatrix} L^{(1)} & L^{(2)} & \cdots & L^{(n_\nu)} \end{pmatrix}$$

où $L^{(1)}, \ldots, L^{(n_\nu)}$ est une $\mathbb{R}$-base de $\mathcal{R}_\nu$. Les colonnes de la matrice $\mathbb{M}(\phi)_\nu$ sont donc formées par les coefficients de chaque relation $L^{(j)}$, $j = 1, \ldots, n_\nu$, dans la base des monômes en les variables s,t de degré ν.

On montre alors qu'à partir d'un certain degré ν_0, toute matrice $\mathbb{M}(\phi)_\nu$, avec $\nu \geq \nu_0$, vérifie les propriétés suivantes :

- elle est formée de $\nu+1$ lignes et $3(\nu+1)-d$ colonnes,
- ses entrées sont des formes linéaires en T_1, T_2, T_3, T_4 à coefficients dans $\mathbb{R}$,
- évaluée en un point $P \in \mathbb{P}^3 \setminus \mathcal{D}$, elle est de rang maximum égal à $\nu+1$,
- évaluée en un point $P \in \mathcal{D} \subset \mathbb{P}^3$, elle est de rang strictement plus petit que $\nu+1$.

Une telle matrice est qualifiée de représentation implicite matricielle de la courbe $\mathcal{D}$ car elle permet de la caractériser par une chute de rang. Il est ici tout à fait remarquable d'obtenir une représentation implicite de notre courbe avec une *unique* matrice. En effet, une représentation implicite classique de $\mathcal{D}$ nécessite pour sa part plusieurs (au moins deux) équations polynomiales.

D'un point de vue du calcul, c'est évidemment la matrice $\mathbb{M}(\phi)_{\nu_0}$ qu'il faut privilégier car c'est celle de plus petite taille. L'entier ν_0 est très simple à contrôler : on peut toujours choisir $\nu_0 = d-1$. Cependant, si l'on calcule le plus petit degré η tel qu'il existe une relation non nulle dans $\mathcal{R}_\eta$, alors on peut choisir $\nu_0 = d-1-\eta$.

Une fois formée, une représentation implicite matricielle d'une courbe paramétrée se manipule de la même façon qu'une représentation implicite matricielle d'une surface paramétrée. Ainsi, notant $\mathbb{M}(\phi)_\nu$, pour $\nu \geq \nu_0$, une représentation implicite matricielle de $\mathcal{D}$, tester si un point $P = (P_1 : P_2 : P_3 : P_4) \in \mathbb{P}^3$ donné appartient à la courbe $\mathcal{D}$ (intersection point/courbe) se fait en calculant le rang de $\mathbb{M}(\phi)_\nu(P)$ qui est une matrice à coefficients dans $\mathbb{R}$. On utilise une DVS afin de calculer un rang numérique approché. De plus, ce calcul de DVS permet également de résoudre le problème d'inversion, c'est-à-dire de déterminer un antécédent $(s_P : t_P)$ de P par ϕ lorsque celui-ci est unique. En effet, par construction même de cette matrice, on a l'égalité matricielle

$$\begin{pmatrix} s_P^\nu & s_P^{\nu-1}t_P & s_P^{\nu-2}t_P & \cdots & t_P^\nu \end{pmatrix} \times \mathbb{M}(\phi)_\nu(P) = 0.$$

De plus, le noyau de la transposée de la matrice $\mathbb{M}(\phi)_\nu(P)$ étant de dimension 1 dans ce cas, le vecteur formant la dernière ligne de la matrice U apparaissant dans une DVS de $\mathbb{M}(\phi)_\nu(P)$, i.e. $\mathbb{M}(\phi)_\nu(P) = U\Sigma V$, est une très bonne approximation d'un vecteur de la forme

$$c. \begin{pmatrix} s_P^\nu & s_P^{\nu-1}t_P & s_P^{\nu-2}t_P & \cdots & t_P^\nu \end{pmatrix}$$

où c est une constante non nulle. Par suite, on en extrait très simplement l'antécédent $(s_P : t_P)$ de P.

Example 2 Considérons la paramétrisation suivante :

$$\left\{\begin{array}{lcl} f_0(s,t) &=& 3s^4t^2-9s^3t^3-3s^2t^4+12st^5+6t^6,\\ f_1(s,t) &=& -3s^6+18s^5t-27s^4t^2-12s^3t^3+33s^2t^4+6st^5-6t^6,\\ f_2(s,t) &=& s^6-6s^5t+13s^4t^2-16s^3t^3+9s^2t^4+14st^5-6t^6,\\ f_3(s,t) &=& -2s^4t^2+8s^3t^3-14s^2t^4+20st^5-6t^6. \end{array}\right.$$

On peut vérifier que $\eta = 2$ et donc que l'on obtient une représentation implicite matricielle dès le degré 3. La matrice $\mathbb{M}(\phi)_3$ est donnée par

$$\left(\begin{array}{cccccc} T_1+T_2 & 0 & 3T_2-3T_3 & 0 & 2T_3-2T_4 & 0\\ -3T_1 & T_1+T_2 & -T_2-3T_3 & 3T_2-3T_3 & -2T_4 & 2T_3-2T_4\\ T_1 & -3T_1 & T_2+3T_3 & -T_2-3T_3 & T_4 & -2T_4\\ 0 & T_1 & 0 & T_2+3T_3 & 0 & T_4 \end{array}\right)$$

5.2. Intersection courbe/courbe

En plus de notre courbe $\mathcal{D}$ paramétrée par (6), supposons à présent donnée une deuxième courbe $\mathcal{C}$, distincte de $\mathcal{D}$, et paramétrée en coordonnées homogènes par

$$\begin{array}{rcl} \Psi : \mathbb{P}^1_{\mathbb{R}} & \to & \mathbb{P}^3_{\mathbb{R}}\\ (u:v) & \mapsto & (x(u,v):y(u,v):z(u,v):w(u,v)). \end{array}$$

Comme toujours, les polynômes $x(u,v), y(u,v), z(u,v)$ et $w(u,v)$ sont des polynômes homogènes dans $\mathbb{R}[u,v]$ sans facteur commun et de même degré. Usant d'une matrice de représentation pour la courbe $\mathcal{D}$, on peut déterminer l'ensemble des points d'intersection $\mathcal{D}\cap\mathcal{C}$ de manière robuste, exactement comme nous l'avons fait pour l'intersection courbe/surface, grâce aux outils de l'algèbre linéaire numérique. Nous répétons ici l'algorithme qui est identique à celui présenté au paragraphe 4.4.

Les données en entrée sont notre courbe $\mathcal{D}$ paramétrée par ϕ, ainsi qu'une matrice de représentation $\mathbb{M}(T_1,\ldots,T_4)$, et notre courbe $\mathcal{C}$ paramétrée par Ψ. On procède alors de la façon suivante :

1. On forme la matrice $\mathbb{M}(\Psi(u,1))$.
2. On forme les matrices compagnes A, B de la matrice $\mathbb{M}(\Psi(u,1))$.
3. On extrait la partie régulière $A'-uB'$ du pinceau $A-uB$.
4. On calcule l'ensemble des valeurs propres (généralisées) $\{u_1,\ldots,u_r\}$ du pinceau $A'-uB'$.
5. On renvoie l'ensemble de points $\{\Psi(u_1),\ldots,\Psi(u_r)\}$ qui correspond aux points d'intersection de $\mathcal{D}\cap\mathcal{C}$ (à l'exception du point $\Psi(1:0)$ qui pourrait éventuellement appartenir à $\mathcal{D}$).

5.3. Lieu d'auto-intersection d'une courbe paramétrée

Une application intéressante des représentations implicites matricielles pour notre courbe $\mathcal{D}$ paramétrée par ϕ est la détermination de ses points singuliers, si elle en possède. En effet, ces points sont obtenus en intersectant la courbe $\mathcal{D}$ avec elle-même. Afin d'être plus précis, rappelons tout d'abord ce que l'on désigne par un point singulier de la courbe $\mathcal{D}$.

Un point P sur la courbe $\mathcal{D}$ est dit singulier si l'espace tangent à $\mathcal{D}$ n'est pas une droite vectorielle. Il y a deux types de points singuliers : les singularités qui correspondent à un recoupement de la courbe avec elle-même, auquel cas ce point possède au moins deux antécédents distincts par ϕ, et les singularités qui sont locales au paramètre (noter que ces deux pathologies peuvent se cumuler). Plus concrètement, choisissons un plan $\mathcal{H}$ dans $\mathbb{P}^3$ passant par P et ne contenant pas notre courbe $\mathcal{D}$. Soit $H(T_1,T_2,T_3,T_4)=0$ une équation de ce plan. Alors, l'intersection de $\mathcal{H}$ et de $\mathcal{D}$ se voit algébriquement au travers du polynôme homogène de degré d dans $\mathbb{C}[s,t]$

$$H(f_1(s,t),f_2(s,t),f_3(s,t),f_4(s,t)) = \prod_{i=1}^{d}(t_i s - s_i t).$$

En effet, les d points $(s_i:t_i)\in\mathbb{P}^1_{\mathbb{C}}$, pas nécessairement distincts, sont tels que $\phi(s_i:t_i)\in\mathcal{H}\cap\mathcal{D}$. On définit alors la *multiplicité d'intersection* de $\mathcal{D}$ avec $\mathcal{H}$ au point P, que l'on note $i_P(\mathcal{H},\mathcal{D})$, comme le nombre de points $(s_i:t_i)$, $i=1,\ldots,d$, tels que $\Psi(s_i:t_i)=P$. À partir de là, on définit la *multiplicité du point* P de $\mathcal{C}$, que l'on note $m_P(\mathcal{C})$, comme le minimum des multiplicités d'intersection $i_P(\mathcal{D},\mathcal{H})$ lorsque $\mathcal{H}$ parcourt tous les plans ne contenant pas $\mathcal{D}$ et passant par le point $P\in\mathcal{D}$, minimum qui est atteint pour un choix suffisamment général de $\mathcal{H}$. Finalement, le point P est appelé un point singulier de $\mathcal{D}$ si et seulement si $m_P(\mathcal{D})\geq 2$.

Soit $\mathbb{M}(\phi)_\nu$, $\nu\geq\nu_0$, une représentation implicite matricielle de $\mathcal{D}$. On montre alors la propriété suivante [BLB10] : pour tout point $P\in\mathbb{P}^3_{\mathbb{C}}$ et tout entier $\nu\geq\nu_0$ on a

$$\text{rang}\,\mathbb{M}(\phi)_\nu(P) = \nu+1-m_P(\mathcal{C}).$$

Par conséquent, une représentation implicite matricielle de $\mathcal{D}$ permet de stratifier les points de $\mathbb{P}^3_{\mathbb{C}}$ en termes de leur multiplicité par rapport à la courbe $\mathcal{C}$.

Il est également possible de déterminer explicitement tous les points singulier de $\mathcal{D}$. En effet, on peut substituer la paramétrisation de $\mathcal{D}$ aux variables T_1,T_2,T_3,T_4 dans la matrice $\mathbb{M}(\phi)_\nu$ pour obtenir la matrice

$$\mathbb{M}(s,t) := \mathbb{M}(\phi)_\nu(f_1(s,t),f_2(s,t),f_3(s,t),f_4(s,t)).$$

Cette opération revient géométriquement à intersecter la courbe $\mathcal{D}$ avec elle-même (noter qu'un tel procédé n'a aucun sens avec une représentation implicite de $\mathcal{D}$ classique par des équations polynomiales puisqu'il conduit aux équations inexploitables $0=0$, alors qu'il prend tout son sens avec une représentation implicite matricielle). À partir de là, on peut appliquer un algorithme complètement similaire à celui utilisé pour l'intersection courbe/courbe. Soulignons que l'usage des notions de rang numérique et de DVS permet de déterminer de manière robuste si un point est très proche d'être singulier.

6. Conclusion

Cet article présente un nouveau concept de représentation implicite d'une courbe ou d'une surface paramétrée. Cette représentation consiste en une matrice dont les entrées sont des formes linéaires en les coordonnées de $\mathbb{R}^3$. Elle caractérise une courbe ou une surface par une propriété de chute de rang. Très simple à calculer, elle s'avère être, en complément d'une paramétrisation, un outil intéressant pour les problèmes d'intersection. Son intérêt principal est notamment de transformer ces problèmes d'intersection en des problèmes d'algèbre linéaire numérique pour lesquels nous disposons d'algorithmes puissants et robustes pour les résoudre (décomposition en valeurs singulières, calcul de valeurs et vecteur propres généralisés). Ainsi, dans le cadre plus particulier du lancer de rayons sur une surface paramétrée, cette nouvelle approche pourrait permettre d'améliorer la robustesse des méthodes existantes dans des situations singulières.

Références

[ACGS07] ARULIAH D. A., CORLESS R. M., GONZALEZ-VEGA L., SHAKOORI A. : Geometric applications of the bezout matrix in the lagrange basis. In *Proceedings of the 2007 international workshop on Symbolic-numeric computation* (London, Ontario, Canada, 2007), ACM, pp. 55–64.

[BC05] BUSÉ L., CHARDIN M. : Implicitizing rational hypersurfaces using approximation complexes. *J. Symbolic Comput.. Vol. 40*, Num. 4-5 (2005), 1150–1168.

[BCD03] BUSÉ L., COX D., D'ANDREA C. : Implicitization of surfaces in $\mathbb{P}^3$ in the presence of base points. *J. Algebra Appl.. Vol. 2*, Num. 2 (2003), 189–214.

[BCJ09] BUSÉ L., CHARDIN M., JOUANOLOU J. : Torsion of the symmetric algebra and implicitization. *Proceedings of the American Mathematical Society. Vol. 137*, Num. 06 (février 2009), 1855–1865.

[BD07] BUSÉ L., DOHM M. : Implicitization of bihomogeneous parametrizations of algebraic surfaces via linear syzygies. In *ISSAC 2007*. ACM, New York, 2007, pp. 69–76.

[BDD09a] BOTBOL N., DICKENSTEIN A., DOHM M. : Matrix representations for toric parametrizations. *Comput. Aided Geom. Design. Vol. 26*, Num. 7 (2009), 757–771.

[BDD09b] BOTBOL N., DOHM M., DICKENSTEIN A. : Matrix representations for toric parametrizations. arXiv :0807.4802 ; to appear in Computer Aided Geometric Design, 2009.

[BJ03] BUSÉ L., JOUANOLOU J.-P. : On the closed image of a rational map and the implicitization problem. *J. Algebra. Vol. 265*, Num. 1 (2003), 312–357.

[BLB] BUSÉ L., LUU BA T. : The surface/surface intersection problem by means of matrix based representations. Preprint HAL inria-00620947.

[BLB10] BUSÉ L., LUU BA T. : Matrix-based Implicit Representations of Rational Algebraic Curves and Applications. *Computer Aided Geometric Design. Vol. 27*, Num. 9 (2010), 681–699.

[CGZ00] COX D., GOLDMAN R., ZHANG M. : On the validity of implicitization by moving quadrics of rational surfaces with no base points. *J. Symbolic Comput.. Vol. 29*, Num. 3 (2000), 419–440.

[Gan66] GANTMACHER F. R. : *Théorie des matrices. Tome 2 : Questions spéciales et applications*. Traduit du Russe par Ch. Sarthou. Collection Universitaire de Mathématiques, No. 19. Dunod, Paris, 1966.

[GVL96] GOLUB G. H., VAN LOAN C. F. : *Matrix computations*, third ed. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 1996.

[KD06] KHETAN A., D'ANDREA C. : Implicitization of rational surfaces using toric varieties. *J. Algebra. Vol. 303*, Num. 2 (2006), 543–565.

[KK96] KUBLANOVSKAYA V., KHAZANOV V. : Spectral problems for pencils of polynomial matrices. methods and algorithms v. *Journal of Mathematical Sciences. Vol. 79(3)* (1996), 1048–1076.

[Kub99] KUBLANOVSKAYA V. : Methods and algorithm of solving spectral problems for polynomial matrices and rational matrix. *Journal of Mathematical Sciences. Vol. 96(3)* (1999), 3085–3287.

[LBBM09] LUU BA T., BUSÉ L., MOURRAIN B. : Curve/surface intersection problem by means of matrix representations. In *SNC* (Kyoto, Japan, 2009), Kai H., Sekigawa H., (Eds.), ACM Press, pp. 71–78.

[MC91] MANOCHA D., CANNY J. : A new approach for surface intersection. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications* (Austin, Texas, United States, 1991), ACM, pp. 209–219.

[SC95] SEDERBERG T., CHEN F. : Implicitization using moving curves and surfaces. In *Proceedings of SIGGRAPH* (1995), vol. 29, pp. 301–308.

Jointures de surfaces canal par des cyclides de Dupin le long de cercles donnés

Lucie Druoton [1,2] Lionel Garnier [3] et Rémi Langevin [2]

[1]CEA, DAM, Valduc, F-21120 Is Sur Tille
[2]IMB, UMR CNRS 5584, Université de Bourgogne, faculté Mirande, 21000 Dijon
[3]LE2I, UMR CNRS 5158, Université de Bourgogne, faculté Mirande, 21000 Dijon

Résumé

Dans quelques cas particuliers, il est possible de réaliser une jointure G^1 de deux surfaces canal par une cyclide de Dupin et de nombreux auteurs ont travaillé sur ce sujet. Dans cet article, nous abordons le cas général, en imposant les cercles de jointures, ce qui nous oblige à utiliser deux morceaux de cyclides de Dupin. Afin de simplifier le problème, nous travaillons dans l'espace des sphères où il suffit de joindre, sur une quadrique de dimension 4*, deux courbes de façon G^1 pour obtenir une jointure G^1 des surfaces dans l'espace à trois dimensions. Nous abordons aussi la jointure d'un cercle et d'une sphère le long d'un cercle donné par une cyclide de Dupin en utilisant la notion de faisceaux de sphères sur la quadrique précitée.*

Mots-clés : Cyclides de Dupin, espace des sphères, jointures G^1

1 Introduction

Les cyclides de Dupin, inventées par P. Dupin en 1822 [Dup22], ont été introduites en CAO par R. Martin en 1982 [Mar82]. De nombreux mathématiciens ont étudié leurs propriétés géométriques, citons entre autres G. Darboux [Dar87, Dar17], A. Forsyth [For12] et A. Cayley [Cay73]. Aujourd'hui, elles sont très largement utilisées pour effectuer des jointures 3D entre deux primitives, en se plaçant uniquement dans l'espace euclidien usuel [AD97a, AD97b, DMP93, Pra90, GFN04, Gar07]. La majorité des algorithmes développés ne traitent que les cas particuliers où il suffit de joindre des courbes par deux arcs de cercles. Dans le cas où ces conditions ne sont pas réalisées, M. Pratt réalise la jointure entre deux cônes ou deux cylindres ayant des axes sécants en utilisant deux cyclides de Dupin [Pra90]. Nous généralisons ces jointures aux surfaces canal et dans le cas de cônes ou de cylindres, nous nous affranchissons de la condition d'intersection des axes. W. Boehm [Boe90], quant à lui, peut réaliser des jointures entre deux cônes ou cylindres n'ayant pas des axes coplanaires, mais il ne choisit pas les cercles de jointures sur ces quadriques et impose le cercle de jointures sur les cyclides.

Ces conditions limitent évidemment le nombre de constructions possibles et le choix des cercles de jointures sur les surfaces initiales n'est pas possible. Une autre solution, développée depuis très peu de temps, consiste à utiliser la représentation des cyclides de Dupin et des surfaces canal dans l'espace des sphères [DGL11a]. Bien que plusieurs représentations, essentiellement équivalentes, soient possibles (cf. M. Berger [Ber78, BG92], M. Paluszny [PB98], U. Hertrich-Jeromin [HJ03], T. Cecil [Cec92]), nous utilisons celle de R. Langevin, J. O'Hara [LO10] et P. Walczak [LW08] qui est une quadrique de dimension 4 de l'espace de Lorentz de dimension 5. Dans cet espace, une cyclide de Dupin est représentée par deux coniques particulières : deux cercles, un cercle et une hyperbole ou un cercle et une parabole. Ces arcs de coniques sont l'intersection d'une quadrique canonique que nous notons Λ^4 et de 2-plans affines. Il est ainsi possible d'étudier des propriétés de jointure 3D (principalement de continuité) en se plaçant sur la quadrique Λ^4 où les calculs sont simplifiés par l'utilisation d'équations de degré 2 remplaçant les équations de cyclides de degré 4 dans l'espace euclidien de dimension 3.

L'article est organisé comme suit : après un bref état de l'art sur l'espace des sphères et la représentation des cyclides de Dupin dans cet espace, nous étudions, dans la section 3,

les positions relatives de deux sphères sur Λ^4. Dans le quatrième paragraphe, nous réalisons une jointure entre deux cercles, dont l'un est sur une sphère donnée, par une cyclide de Dupin. Avant de conclure et de donner nos perspectives de travail, dans la section 5, nous réalisons une jointure entre deux surfaces canal par deux cyclides de Dupin.

2 Les cyclides de Dupin dans l'espace des sphères

Nous allons rappeler le minimum nécessaire à la compréhension des algorithmes donnés dans cet article. Pour plus de détails sur la théorie mathématique, le lecteur pourra se reporter à [HJ03, Cec92, LW08, DGL11a].

$\mathscr{E}_3$ est l'espace affine à trois dimensions d'espace vectoriel attaché $\overrightarrow{\mathscr{E}_3}$ muni de la forme quadratique $\mathscr{Q}_3$, définie positive, suivante :

$$\forall \overrightarrow{v}\,(x;y;z) \in \overrightarrow{\mathscr{E}_3},\ \mathscr{Q}_3(\overrightarrow{v}) = x^2+y^2+z^2$$

2.1 L'espace de Lorentz

Soit $\overrightarrow{\mathrm{L}_{4,1}}$ l'espace vectoriel de dimension 5 de base $(\overrightarrow{e_i})_{i\in[\![0,4]\!]}$, muni de la forme de Lorentz, bilinéaire symétrique définie et de signature $(4;1)$:

$$\begin{array}{rccc} \mathscr{L}_{4,1}: & \overrightarrow{\mathrm{L}_{4,1}}\times\overrightarrow{\mathrm{L}_{4,1}} & \longrightarrow & \mathbb{R} \\ & (\overrightarrow{u};\overrightarrow{v}) & \longmapsto & -x_0y_0+\sum_{i=1}^{4}x_iy_i \end{array} \qquad (1)$$

où $\overrightarrow{u}\,(x_0;\ldots;x_4)$ et $\overrightarrow{v}\,(y_0;\ldots;y_4)$ appartiennent à $\overrightarrow{\mathrm{L}_{4,1}}$. La forme quadratique définie associée à $\mathscr{L}_{4,1}$ est notée $\mathscr{Q}_{4,1}$.

Soit $\mathrm{L}_{4,1}$ l'espace affine, associé à l'espace vectoriel $\overrightarrow{\mathrm{L}_{4,1}}$, muni du repère $(O_5,\overrightarrow{e_0},\overrightarrow{e_1},\ldots,\overrightarrow{e_4})$ où O_5 a pour coordonnées $(0;0;0;0;0)$. Dans cet espace, trois quadriques jouent un rôle important.

2.2 Les quadriques particulières de $\mathrm{L}_{4,1}$

2.2.1 La « sphère » de rayon 0

Dans $\mathrm{L}_{4,1}$, C_l est la « sphère » de rayon nul† définie par :

$$\mathrm{C}_l = \left\{M\in\mathrm{L}_{4,1}\mid \mathscr{Q}_{4,1}\left(\overrightarrow{O_5M}\right)=0\right\} \qquad (2)$$

et nous pouvons distinguer différents types de vecteurs et plans dans $\mathrm{L}_{4,1}$, tableaux 1 et 2.

2.2.2 Le paraboloïde P isométrique à $\mathscr{E}_3$

Nous devons construire l'espace euclidien $\mathscr{E}_3$ ambiant comme une sous-variété de $\mathrm{L}_{4,1}$ afin de manipuler simultanément les sphères et les plans contenus dans $\mathscr{E}_3$ et les points de $\mathscr{E}_3$. Il faut, pour cela, construire une isométrie entre $\mathscr{E}_3$ et

† Le cône de lumière avec un œil euclidien.

Type de vecteur $\overrightarrow{v}$	
Espace	$\mathscr{Q}_{4,1}(\overrightarrow{v})>0$
Lumière	$\mathscr{Q}_{4,1}(\overrightarrow{v})=0$
Temps	$\mathscr{Q}_{4,1}(\overrightarrow{v})<0$

Table 1: *Les trois types de vecteurs $\overrightarrow{v}$ de $\overrightarrow{L_{4,1}}$.*

Type	Plan
Espace	Tous les vecteurs sont de type espace
Temps	Contient deux vecteurs lumière indépendants ou contient au moins un vecteur temps
Lumière	Plan parallèle à un plan tangent à C_l

Table 2: *Trois types de plans de $\overrightarrow{L_{4,1}}$ et $L_{4,1}$.*

le paraboloïde P, figure 1, section du cône C_l par un hyperplan affine $\mathscr{H}$ particulier de $\mathrm{L}_{4,1}$ [DGL11a]. Les formules de passage entre $\mathscr{E}_3$ et P ne sont pas données ici, le lecteur peut se reporter à [DGL11a].

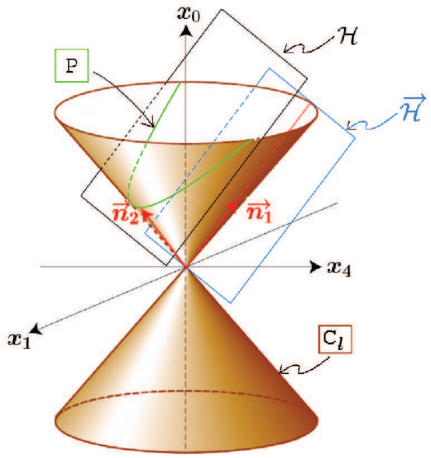


Figure 1: *Construction du paraboloïde P de dimension 3 isométrique à $\mathscr{E}_3$. L'hyperplan $\overrightarrow{\mathscr{H}}$ est tangent à C_l. Chaque vecteur lumière $\overrightarrow{n}$, non colinéaire à $\overrightarrow{n_1}$, correspond à un point de $\mathscr{E}_3$ via le paraboloïde P : ce point est l'intersection de la droite $(O;\overrightarrow{n})$ et de P.*

2.2.3 La « sphère » de rayon 1

Dans $\mathrm{L}_{4,1}$, Λ^4, qui représente l'espace des sphères orientées et des plans orientés de $\mathscr{E}_3$, est la « sphère » unitaire‡ définie par :

$$\Lambda^4 = \left\{M\in\mathrm{L}_{4,1}\mid \mathscr{Q}_{4,1}\left(\overrightarrow{O_5M}\right)=1\right\} \qquad (3)$$

‡ Un hyperboloïde à une nappe avec un œil euclidien.

Les formules de passage entre ces sphères, ces plans et Λ^4 ne sont pas données ici, le lecteur peut se reporter à [DGL11a]. Notons cependant que les deux points de Λ^4 correspondant à la même sphère non orientée ou au même plan non orienté sont symétriques par rapport à l'origine. Ainsi, dans cet article, le terme sphère désigne aussi bien une sphère orientée qu'un plan orienté de $\mathscr{E}_3$.

2.3 Faisceaux linéaires de sphères et correspondance dans Λ^4

Dans [LW08], les auteurs ont rappelé que tout faisceau linéaire de sphères de $\mathscr{E}_3$ est représenté dans $L_{4,1}$ par la section de la quadrique Λ^4 par un 2-plan $\mathscr{P}$ contenant l'origine. Selon le type du plan $\mathscr{P}$, nous avons différents types de faisceaux de sphères, tableau 3.

Type de $\mathscr{P}$	Faisceau de sphères	Figure
Espace	à base cercle	2
Temps	à points limites	3
Lumière	tangentes en un point (via le paraboloïde P)	4

Table 3: *Différents type de faisceaux de sphères appartenant à la section de Λ^4 par un 2-plan $\mathscr{P}$ passant par l'origine.*

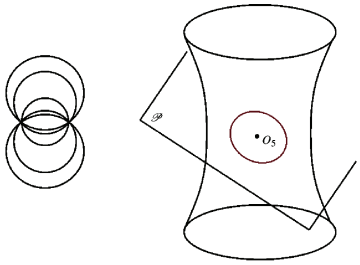


Figure 2: *Coupe plane d'un faisceau de sphères à base cercle (toute les sphères contiennent ce cercle) et la courbe correspondante dans Λ^4. Le 2-plan $\mathscr{P}$ est de type espace.*

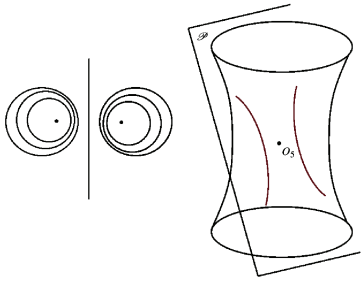


Figure 3: *Coupe plane d'un faisceau de sphères à points limites et la courbe correspondante dans Λ^4. Le 2-plan $\mathscr{P}$ est de type temps.*

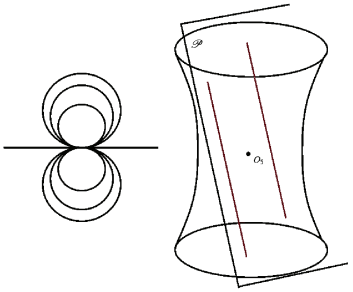


Figure 4: *Coupe plane d'un faisceau de sphères tangentes en un point et les deux droites lumière correspondantes dans Λ^4. Le 2-plan $\mathscr{P}$ est de type lumière.*

2.4 Généralités sur les surfaces canal

Une surface canal Γ est engendrée par une famille à un paramètre de sphères orientées $\mathtt{S}(t)$. Chaque sphère de cette famille est tangente à la surface canal Γ en un cercle appelé **cercle caractéristique**. Dans Λ^4, une famille à un paramètre des sphères est représentée par une courbe $t \mapsto \gamma(t)$. Pour déterminer le cercle caractéristique de la sphère $\mathtt{S}(t_0)$, il suffit de considérer la sphère $\dot{\mathtt{S}}(t_0)$ correspondant à l'intersection entre Λ^4 et la droite passant par O_5 et de vecteur directeur $\overrightarrow{\dot{\gamma}(t_0)}$. Notons que les sphères $\mathtt{S}(t_0)$ et $\dot{\mathtt{S}}(t_0)$ sont orthogonales, figure 7. Dans la suite, afin de ne pas alourdir le texte, un cercle caractéristique est associé directement à ce vecteur tangent.

2.5 Cyclides de Dupin

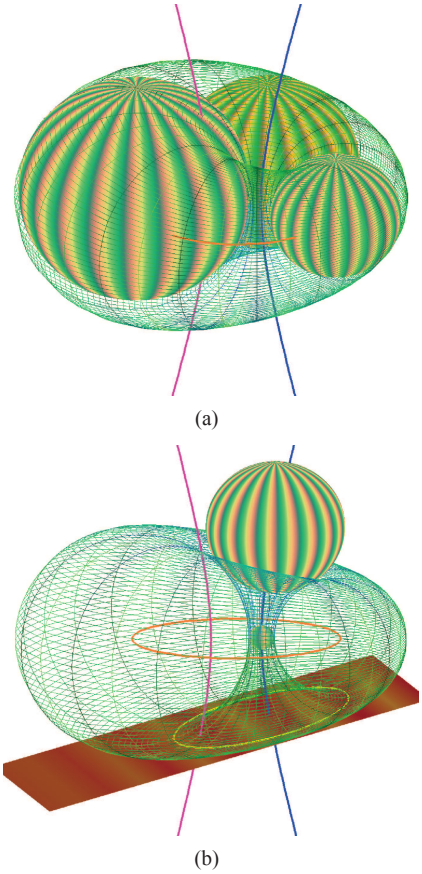
Une cyclide de Dupin est une surface canal particulière : elle est l'enveloppe de deux familles à un paramètre de sphères orientées [Dar87], figure 5.

Lorsque la cyclide de Dupin n'est pas dégénérée, les lieux des centres des sphères sont une ellipse et une hyperbole situées dans deux plans orthogonaux, elle dépend de trois paramètres a, c et μ et il existe un repère dans lequel son équation paramétrique est la suivante [For12, Gar07] :

$$F_{a,c,\mu}(\theta,\psi) = \begin{pmatrix} \dfrac{\mu(c - a\cos\theta\cos\psi) + b^2\cos\theta}{a - c\cos\theta\cos\psi} \\ \dfrac{b\sin\theta\,(a - \mu\cos\psi)}{a - c\cos\theta\cos\psi} \\ \dfrac{b\sin\psi\,(c\cos\theta - \mu)}{a - c\cos\theta\cos\psi} \end{pmatrix}$$

où $b = \sqrt{a^2 - c^2}$ et $\theta \in [0; 2\pi]$, $\psi \in [0; 2\pi]$.

Les cyclides de Dupin dégénérées sont celles dont les centres d'une des familles de sphères ne sont pas sur une conique propre et nous pouvons citer comme exemples, les tores, les cônes et les cylindres de révolution. Cependant, il est facile de passer d'une cyclide de Dupin dégénérée



(a) (b)

Figure 5: *Une cyclide de Dupin non dégénérée, enveloppe de deux familles de sphères à un paramètre. (a) : les sphères sont centrées sur une ellipse. (b) : les sphères sont centrées sur une hyperbole.*

à une cyclide de Dupin non dégénérée par une inversion [Gar07, PG10].

Une cyclide de Dupin est représentée dans l'espace des sphères Λ^4 par deux courbes[§], « cercles » ou un « cercle » et une « droite », obtenus comme sections de Λ^4 par deux 2-plans affines orthogonaux [DGL11a], figure 6. De plus, ces plans sont tous deux orthogonaux à la droite des centres des « cercles », qui passe par l'origine. Notons que lorsque nous parlons d'orthogonalité dans $L_{4,1}$, il s'agit toujours de l'orthogonalité relative à la forme de Lorentz $\mathcal{Q}_{4,1}$.

D'un point de vue euclidien, nous pouvons différencier trois types de courbes γ (pour la forme de Lorentz) selon les types de plans les contenant, tableau 4 et cette nature permet de connaître le nombre de point(s) singulier(s) de la cyclide, tableau 5.

§ Chaque courbe correspond à l'une des familles de sphères dont la cyclide de Dupin est l'enveloppe.

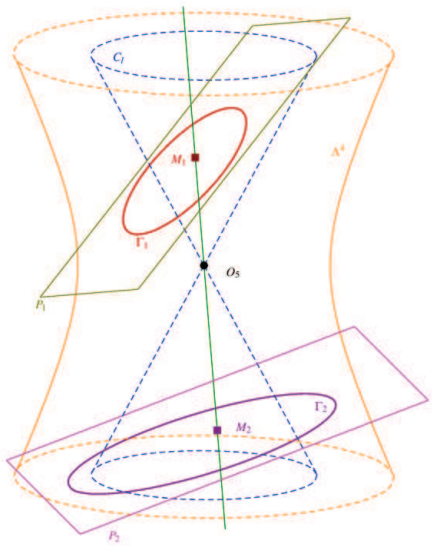


Figure 6: *Représentation des deux familles de sphères d'une cyclide de Dupin dans* Λ^4.

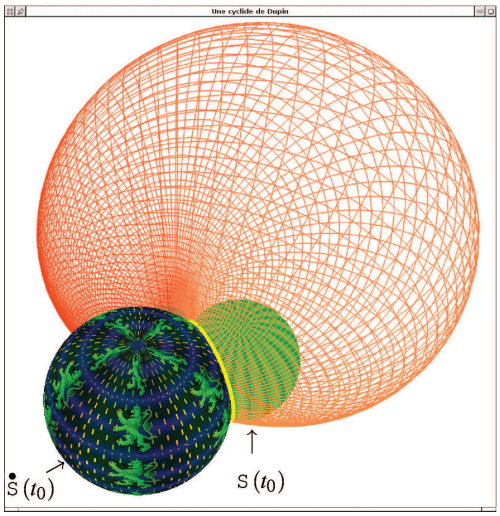


Figure 7: *Représentation d'un cercle caractéristique d'une cyclide comme intersection de deux sphères* $S(t_0)$ *et* $\dot{S}(t_0)$ *orthogonales.*

3 Positions entre sphères de $\mathscr{E}_3$ via leur représentation dans Λ^4

Considérons une sphère orientée S de $\mathscr{E}_3$ correspondant à un point σ de Λ^4. Notons $\sigma^- = -\sigma$, le point symétrique de σ par rapport à l'origine O_5, correspondant à l'autre orientation de la sphère S, i.e :

$$\overrightarrow{O_5\sigma^-} = -\overrightarrow{O_5\sigma}$$

Type de plan	Nature de γ	Vue euclidienne
Espace	« cercle »	cercle
Lumière	« droite »	parabole
Temps	« cercle »	hyperbole équilatère

Table 4: *Nature de la courbe en fonction du type du plan coupant* Λ^4.

Vue euclidienne	Nombre de point(s) singulier(s)
deux cercles	0
un cercle et une parabole	1
un cercle et hyperbole équilatère	2

Table 5: *Nombres de points singuliers d'une cyclide de Dupin en fonction de la nature des courbes la représentant dans* Λ^4.

L'intersection de l'hyperplan $T_\sigma\Lambda^4$ (resp. $T_{\sigma^-}\Lambda^4$) tangent à Λ^4 en σ (resp.σ^-) avec Λ^4 est une « sphère »[¶] $\mathscr{C}_\sigma$ (resp. $\mathscr{C}_{\sigma^-}$) de dimension 3, de centre σ (resp. σ^-) et de rayon nul, figure 8. Par analogie avec ce qui se passe dans $\mathscr{E}_3$ du point de vue euclidien, nous pouvons définir trois espaces délimités par $\mathscr{C}_\sigma$, tableau 6.

σ_x est	Condition	Nature de $\overrightarrow{\sigma\sigma_x}$
à l'intérieur de $\mathscr{C}_\sigma$	$\mathscr{Q}_{4,1}\left(\overrightarrow{\sigma\sigma_x}\right) < 0$	Temps
sur $\mathscr{C}_\sigma$	$\mathscr{Q}_{4,1}\left(\overrightarrow{\sigma\sigma_x}\right) = 0$	Lumière
à l'extérieur de $\mathscr{C}_\sigma$	$\mathscr{Q}_{4,1}\left(\overrightarrow{\sigma\sigma_x}\right) > 0$	Espace

Table 6: *Définition de l'intérieur et de l'extérieur de la « sphère »* $\mathscr{C}_\sigma$*, de centre* σ *et de rayon nul, de dimension* 3*, définie par :* $\mathscr{C}_\sigma = \Lambda^4 \cap T_\sigma\Lambda^4$.

En comparant $|\mathscr{L}_{4,1}\left(\overrightarrow{O_5\sigma}, \overrightarrow{O_5\sigma_x}\right)|$ à 1, nous allons caractériser l'intérieur et l'extérieur de $\mathscr{C}_\sigma \cup \mathscr{C}_{\sigma^-}$. Ainsi, nous pourrons connaître l'intersection dans $\mathscr{E}_3$ entre les deux sphères définies par σ et τ :

Théorème 1 : Positions relatives de deux sphères

Soient $\mathtt{S}$ et $\mathtt{S}_x$ deux sphères orientées de $\mathscr{E}_3$. Soient σ et σ_x leurs représentations dans Λ^4.

Alors, nous avons les trois propositions suivantes :

- $\mathtt{S} \cap \mathtt{S}_x$ est un cercle ssi $\left|\mathscr{L}_{4,1}\left(\overrightarrow{O_5\sigma}, \overrightarrow{O_5\tau}\right)\right| < 1$
- $\mathtt{S}$ et $\mathtt{S}_x$ sont tangentes ssi $\left|\mathscr{L}_{4,1}\left(\overrightarrow{O_5\sigma}, \overrightarrow{O_5\tau}\right)\right| = 1$
- $\mathtt{S} \cap \mathtt{S}_x = \emptyset$ ssi $\left|\mathscr{L}_{4,1}\left(\overrightarrow{O_5\sigma}, \overrightarrow{O_5\tau}\right)\right| > 1$

Démonstration : Soit $\mathscr{P}$ le 2-plan défini par O_5, σ et τ.

[¶] Avec un œil euclidien, nous avons un cône.

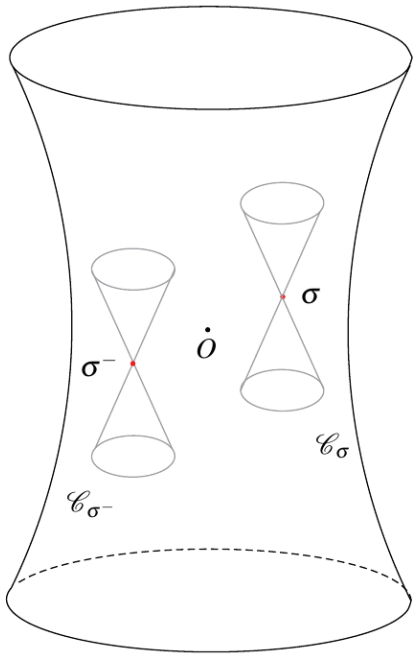


Figure 8: *« Sphères »* $\mathscr{C}_\sigma$ *et* $\mathscr{C}_{\sigma^-}$ *de centre* σ *et* σ^- *et de rayon nul :* $\mathscr{C}_\sigma = \Lambda^4 \cap T_\sigma\Lambda^4$ *et* $\mathscr{C}_\sigma^- = \Lambda^4 \cap T_{\sigma^-}\Lambda^4$.

Nous considérons le vecteur $\overrightarrow{u}$ de $\overrightarrow{\mathscr{P}}$, pseudo-unitaire (s'il n'est pas de type lumière), orthogonal à la droite $(O_5\tau)$. Ainsi :

$$\exists(\alpha;\beta) \in \mathbb{R}^2 \mid \overrightarrow{O_5\sigma} = \alpha\,\overrightarrow{O_5\tau} + \beta\,\overrightarrow{u} \tag{4}$$

d'où :

$$1 = \mathscr{Q}_{4,1}\left(\overrightarrow{O_5\sigma}\right) = \alpha^2 + \beta^2\,\mathscr{Q}_{4,1}\left(\overrightarrow{u}\right) \tag{5}$$

puisque, d'une part, σ et τ appartiennent à Λ^4 et d'autre part, $\overrightarrow{u}$ est $\mathscr{L}_{4,1}$-orthogonal à $\overrightarrow{O_5\tau}$. Pour les mêmes raisons, nous avons :

$$\mathscr{L}_{4,1}\left(\overrightarrow{O_5\sigma}, \overrightarrow{O_5\tau}\right) = \alpha \tag{6}$$

ce qui permet de comparer $|\alpha|$ à 1, tableau 7.

Type de $\overrightarrow{u}$	$\mathscr{Q}_{4,1}\left(\overrightarrow{u}\right)$	Conclusion
Temps	-1	$\lvert\alpha\rvert > 1$
Lumière	0	$\lvert\alpha\rvert = 1$
Espace	1	$\lvert\alpha\rvert < 1$

Table 7: *Comparaison de* $|\alpha|$ *à* 1 *à partir de la formule (5).*

De plus, à partir de la formule (4) nous avons :

$$\begin{aligned}\mathscr{Q}_{4,1}\left(\overrightarrow{\sigma\tau}\right) &= \mathscr{Q}_{4,1}\left((1-\alpha)\,\overrightarrow{O_5\tau} + \beta\,\overrightarrow{u}\right)\\ &= (1-\alpha)^2\,\mathscr{Q}_{4,1}\left(\overrightarrow{O_5\tau}\right) + \beta^2\,\mathscr{Q}_{4,1}\left(\overrightarrow{u}\right)\end{aligned}$$

ce qui conduit, pour les mêmes raisons que précédemment, à la relation fondamentale suivante :

$$\mathcal{Q}_{4,1}\left(\overrightarrow{\sigma\tau}\right) = (1-\alpha)^2 + \beta^2\, \mathcal{Q}_{4,1}\left(\overrightarrow{u}\right) \qquad (7)$$

qui permet de déduire la nature du vecteur $\overrightarrow{\sigma\tau}$, tableau 8.

Type de $\overrightarrow{u}$	$\mathcal{Q}_{4,1}(\overrightarrow{u})$	α	Type de $\overrightarrow{\sigma\tau}$
Temps	-1	$\lvert\alpha\rvert > 1$	Temps
Lumière	0	$\lvert\alpha\rvert = 1$	Lumière
Espace	1	$\lvert\alpha\rvert < 1$	Espace

Table 8: *Type de la droite $(\sigma\tau)$ à partir de la formule (7) et du tableau 7.*

Comme les propriétés donnés dans le tableau 3 sont toujours valides, d'après le tableau 8 et la formule (6), nous avons le résultat qu'il fallait démontrer. □

Notons que les valeurs absolues nous permettent de nous affranchir de l'orientation de la sphère S puisque nous avons :

$$\mathcal{L}_{4,1}\left(\overrightarrow{O_5\sigma^-}, \overrightarrow{O_5\tau}\right) = -\mathcal{L}_{4,1}\left(\overrightarrow{O_5\sigma}, \overrightarrow{O_5\tau}\right)$$

4 Jointure G^1 entre une sphère le long d'un cercle et un autre cercle

Dans tout le reste de cet article, nous ne considérons que des cercles caractéristiques sur les cyclides de Dupin (i.e. nous ne nous intéressons pas aux cercles de Villarceau).

Ce travail provient en fait de deux contextes : le premier consiste à choisir un cercle caractéristique C_1 sur une surface canal et à joindre de façon G^1 cette surface canal, le long de ce cercle, avec un cercle C par une cyclide de Dupin. La donnée du cercle C_1 sur la surface canal impose la sphère S contenant le cercle C. Le second consiste à joindre des cyclides de Dupin le long de cercles donnés et dans ce cas, nous avons une famille à un paramètre de couples de cyclides de Dupin. Le choix d'une sphère contenant un des cercles revient à privilégier un couple de cyclides de Dupin dans cette famille. La condition d'existence est la suivante :

Théorème 2 : Jointure de deux cercles par une cyclide
Soit deux cercles C et C_1.
Il est possible de trouver une cyclide de Dupin telle que C et C_1 soient deux cercles caractéristiques de cette cyclide si et seulement si les deux cercles sont soit coplanaires, soit cosphériques (i.e. il existe une sphère S_0 contenant les deux cercles C et C_1).

Démonstration : L'assertion est évidente en considérant les cercles parallèles sur un tore (croisé, à collier nul ou à collier). Par inversion, nous obtenons tous les cas. En particulier, il est possible d'intervertir les rôles des cercles parallèles et des cercles méridiens [PG10]. □

Dans l'espace des sphères, les deux courbes représentant les faisceaux de sphères à bases C et C_1 se coupent en deux points qui correspondent aux deux sphères orientées définies par la sphère S_0.

La figure 9 illustre le théorème 2 dans le cas d'un tore à collier.

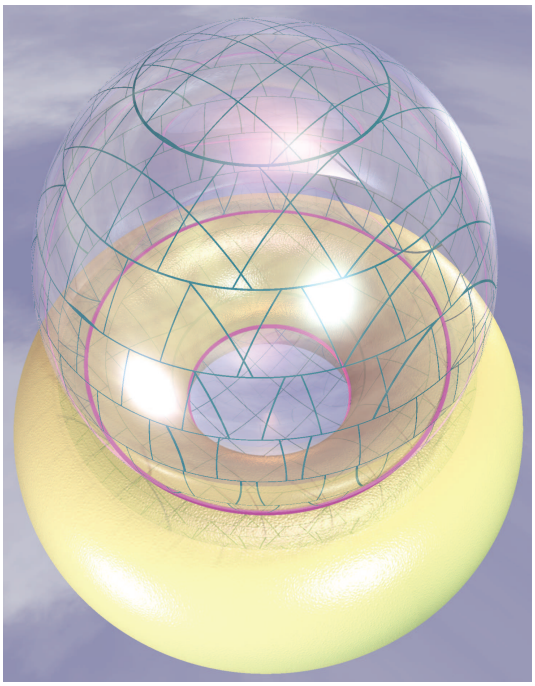


Figure 9: *Deux cercles cosphériques sur un tore à collier.*

L'algorithme 1 permet de trouver la cyclide de Dupin, tangente à la sphère S le long du cercle C telle que le cercle C_1 soit un de ses cercles caractéristiques, figure 13. Le principe est le suivant :

- à la sphère S correspond le point σ de Λ^4 ;
- au cercle C sur la sphère S correspond un vecteur $\overrightarrow{v_0}$;
- le cercle C_1 engendre un cercle $t \mapsto \gamma(t)$ dans un 2-plan de type espace contenant O_5. Ce cercle Γ_1 représente le faisceau des sphères à base le cercle C_1, figure 10.

Trouver une cyclide de Dupin répondant à notre problème revient, dans Λ^4, à trouver le réel t_0 tel que la courbe obtenue comme section de Λ^4 par le plan affine engendré par $\gamma_1(t_0)$, σ et $\overrightarrow{v_0}$ ait pour vecteur tangent, en σ, un vecteur colinéaire à $\overrightarrow{v_0}$. Notons que nous avons l'avantage de travailler avec un « cercle » Γ d'un 2-plan $\mathscr{P}$: si nous notons Ω son centre alors le vecteur $\overrightarrow{u}$ de $\overrightarrow{\mathscr{P}}$ et tangent à Γ au point σ si et seulement si nous avons :

$$\mathcal{L}_{4,1}(\overrightarrow{\Omega\sigma}, \overrightarrow{u}) = 0$$

Notons que si E est un ensemble de points ou de vecteurs, $Aff(E)$ est l'espace affine engendré par les éléments de E.

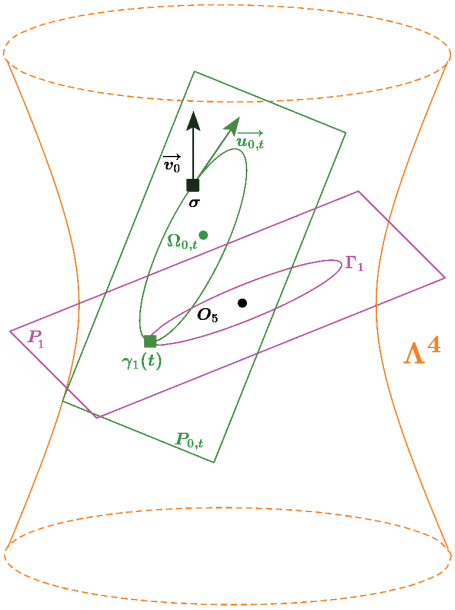


Figure 10: *Illustration de l'algorithme 1 dans l'espace des sphères. Nous cherchons la valeur de t qui permet d'obtenir un vecteur tangent au « cercle » (obtenu comme section de Λ^4 par le plan affine défini par $\gamma(t)$, σ et $\vec{v_0}$) en σ colinéaire à $\vec{v_0}$.*

Concernant la détermination des paramètres de la cyclide de Dupin à partir d'une de ses représentations dans l'espace des sphères, le lecteur peut se reporter à [DGL11a]. Une autre possibilité concernant la visualisation est la construction des cercles caractéristiques en utilisant une méthode de subdivision. [DGL11b]

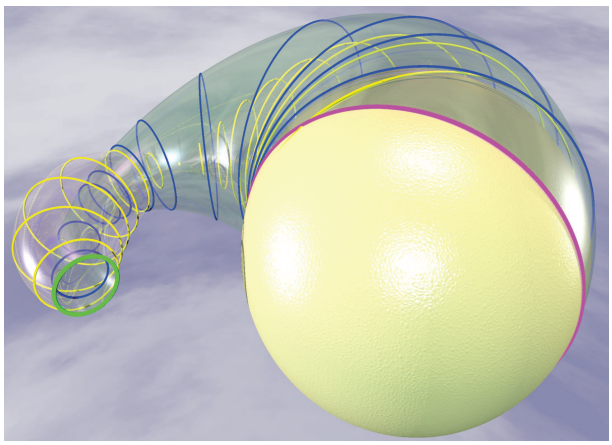


Figure 11: *Exemple de jointure entre deux cercles dans $\mathscr{E}_3$ par deux morceaux de cyclide de Dupin imposée par une sphère tangente en l'un des deux cercles.*

Algorithme 1 Jointure G^1 entre un cercle sur une sphère et un autre cercle.

Entrée : Le cercle C sur la sphère S et le cercle C_1
Condition : Les cercles C et C_1 sont cosphériques.

1. Calcul de σ correspondant à S dans Λ^4
2. Calcul de $\vec{v_0}$ associé à C dans $\overrightarrow{L_{4,1}}$
3. Calcul de Γ_1 correspondant au faisceau de sphères de $\mathscr{E}_3$ à base le cercle C_1 dans Λ^4
4. Soit $\gamma_1(t) \in \Gamma_1$
5. Détermination du Plan $P_{0,t} = Aff(\sigma, \vec{v_0}, \overrightarrow{\sigma\gamma_1(t)})$
6. Détermination du « cercle » $\Gamma_{0,t} = P_{0,t} \cap \Lambda^4$
7. Calcul du centre $\Omega_{0,t}$ du « cercle » $\Gamma_{0,t}$
8. Calcul de $\vec{u_{0,t}}$, vecteur tangent à $\Gamma_{0,t}$ en σ tel que :
$$\mathscr{L}_{4,1}(\overrightarrow{\Omega_{0,t}\sigma}, \vec{u_{0,t}}) = 0$$
9. Détermination de l'une des deux valeurs t_0 tel que les vecteurs $\vec{u_{0,t_0}}$ et $\vec{v_0}$ soient colinéaires

Sortie : Une courbe Γ_{0,t_0} représentant une famille de sphères qui a pour enveloppe la cyclide de Dupin effectuant la jointure entre C et C_1 tangente à S en C.

5 Jointure G^1 entre deux surfaces canal par deux cyclides de Dupin le long de deux cercles

5.1 Principe et algorithme

Nous souhaitons joindre de manière G^1 deux surfaces canal quelconques $Surf_1$ et $Surf_2$ le long de deux cercles caractéristiques C_1 et C_2 par deux morceaux de cyclides de Dupin.

Dans Λ^4, la surface canal $Surf_1$ (resp. $Surf_2$) est représentée par la courbe γ_1 (resp. γ_2) dont l'extrémité de jointure est le point σ_1 de γ_1 (resp. σ_2 de γ_2). σ_1 (resp.σ_2) représente la sphère tangente S_1 (resp. S_2) à la surface canal $Surf_1$ (resp. $Surf_2$) le long du cercle C_1 (resp. C_2). Nous pouvons calculer le vecteur $\vec{v_1}$ (resp. $\vec{v_2}$) tangent à γ_1 (resp. γ_2) en σ_1 (resp. σ_2) représentant le cercle caractéristique C_1 (resp. C_2). Le travail consiste à joindre les deux couples points-vecteurs $(\sigma_1; \vec{v_1})$ et $(\sigma_2; \vec{v_2})$ par deux arcs de « cercles », dans une section de Λ^4 par un espace affine de dimension 3, qui se recollent de façon G^1 en un point σ. Du point de vue euclidien, ces sections peuvent être soit une sphère S^2 où le travail a déjà été fait [Smu04, Sha87], un cylindre, un hyperboloïde à une nappe Λ^2 ou un hyperboloïde à deux nappes $\mathbb{H}^2$.

Soit σ un point de Λ^4. Soit $P_{1,\sigma} = Aff(\sigma_1, \sigma, \vec{v_1})$ et $P_{2,\sigma} = Aff(\sigma_2, \sigma, \vec{v_2})$. Les deux courbes suivantes :

$$\Gamma_{1,\sigma} = \Lambda^4 \cap P_{1,\sigma} \qquad \Gamma_{2,\sigma} = \Lambda^4 \cap P_{2,\sigma}$$

représentent chacune une cyclide de Dupin $Cycl_1$ et $Cycl_2$. La cyclide de Dupin $Cycl_1$ (resp. $Cycl_2$) joint de manière G^1

la surface canal $Surf_1$ (resp. $Surf_2$) le long de C_1 (resp. C_2) et la sphère S correspondant à σ le long d'un cercle $C_{\sigma,1}$ (resp. $C_{\sigma,2}$). En général, $C_{\sigma,1}$ et $C_{\sigma,2}$ ne représentent pas le même cercle sur S. Dans ce cas, la jointure entre les deux cyclides n'est pas G^1. Il est possible d'obtenir une jointure G^1 en raccordant chacune des deux cyclides de Dupin sur une même sphère [LSD*].
Pour assurer cette G^1-continuité, nous imposons que les cercles $C_{\sigma,1}$ et $C_{\sigma,2}$ soient confondus. Cette condition impose que les vecteurs $\overrightarrow{u_{1,\sigma}}$ (resp. $\overrightarrow{u_{2,\sigma}}$) tangents à $\Gamma_{1,\sigma}$ (resp. $\Gamma_{2,\sigma}$) en σ soient colinéaires.

Dans l'algorithme 2, nous obtenons un degré de liberté concernant le choix de la solution σ_0.

Notons que si nous changeons l'orientation de la sphère S dans $\mathscr{E}_3$, la jointure entre les deux cyclides dans $\mathscr{E}_3$ est toujours G^1 tandis que la jointure des deux courbes $\Gamma_{1,\sigma}$ et $\Gamma_{2,\sigma}$ n'est même plus G^0 dans Λ^4 : une courbe a pour extrémité σ alors que l'autre a pour extrémité $\sigma^- = -\sigma$.

La figure 12 illustre une jointure G^1 entre les surfaces $Surf_1$, $Cycl_1$, $Cycl_2$ et $Surf_2$, dans Λ^4, en utilisant l'algorithme 2, le résultat dans $\mathscr{E}_3$ est montré par la figure 13.

Algorithme 2 Jointure G^1 entre deux surfaces canal le long de deux cercles donnés par deux morceaux de cyclides de Dupin

Entrée : Les cercles C_1 et C_2 donnés sur les surfaces canal $Surf_1$ et $Surf_2$

1. Calcul de la sphère S_1 (resp. S_2) tangente à $Surf_1$ (resp. $Surf_2$) en C_1 (resp. C_2)
2. Calcul de σ_1 et σ_2 correspondant à S_1 et S_2 dans Λ^4.
3. Calcul de $\overrightarrow{v_1}$ et $\overrightarrow{v_2}$ associés à C_1 et C_2 dans $\overrightarrow{L_{4,1}}$
4. Soit $\sigma \in \Lambda^4$, calcul des 2-plans $P_{1,\sigma} = Aff(\sigma_1, \sigma, \overrightarrow{v_1})$ et $P_{2,\sigma} = Aff(\sigma_2, \pm\sigma, \overrightarrow{v_2})$ en fonction de σ
5. Calcul, en fonction de σ, des « cercles » :
$$\Gamma_{1,\sigma} = P_{1,\sigma} \cap \Lambda^4$$
$$\Gamma_{2,\sigma} = P_{2,\sigma} \cap \Lambda^4$$
6. Calcul de $\Omega_{1,\sigma}$ et $\Omega_{2,\sigma}$, centres des cercles Lorentz $\Gamma_{1,\sigma}$ et $\Gamma_{2,\sigma}$
7. Calcul de $\overrightarrow{u_{1,\sigma}}$ et $\overrightarrow{u_{2,\sigma}}$, vecteurs tangents en σ à $\Gamma_{1,\sigma}$ et $\Gamma_{2,\sigma}$ en résolvant le système suivant :
$$\begin{cases} \mathscr{L}_{4,1}(\overrightarrow{\Omega_{1,\sigma}\sigma}, \overrightarrow{u_{1,\sigma}}) = 0 \\ \mathscr{L}_{4,1}(\overrightarrow{\Omega_{2,\sigma}\sigma}, \overrightarrow{u_{2,\sigma}}) = 0 \end{cases}$$
8. Détermination de σ_0 tel que les vecteurs $\overrightarrow{u_{1,\sigma_0}}$ et $\overrightarrow{u_{2,\sigma_0}}$ soient colinéaires.

Sortie : Les deux courbes Γ_{1,σ_0} et Γ_{2,σ_0} de Λ^4 correspondant aux deux cyclides de Dupin effectuant une jointure G^1 entre les surfaces canal $Surf_1$ et $Surf_2$ le long des cercles C_1 et C_2.

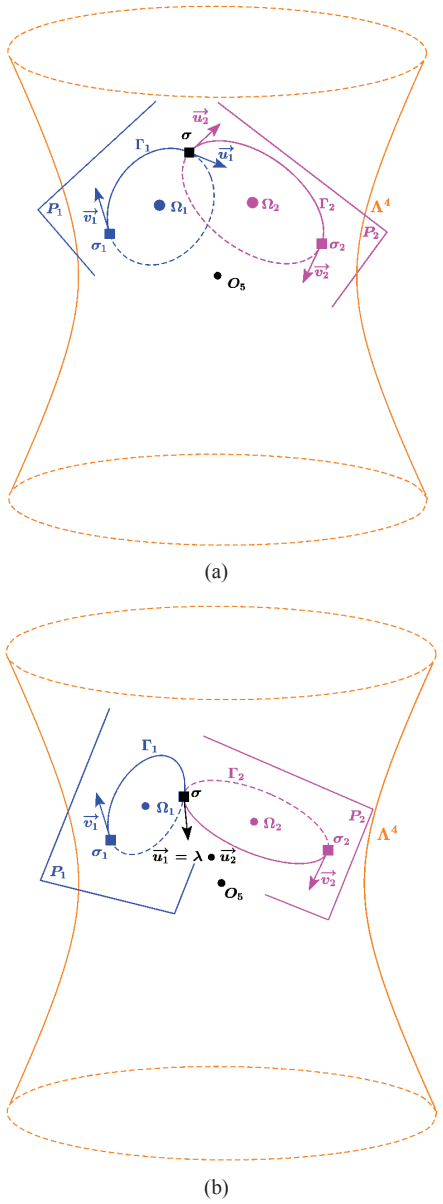


Figure 12: *Illustration de l'algorithme 2 dans l'espace des sphères. (a) Cas général, les deux vecteurs tangents au point σ ne sont pas colinéaires. (b) Jointure G^1, le point σ est calculé afin que les deux vecteurs tangents au point σ soient colinéaires.*

5.2 Exemple numérique

Dans $\mathscr{E}_3$, muni du repère orthonormé direct $(O_3, \overrightarrow{\imath}, \overrightarrow{j}, \overrightarrow{k})$, nous souhaitons joindre deux cylindres Cyl_1 et Cyl_2. Dans la suite toutes les valeurs numériques approchées sont données au centième près. Le cylindre Cyl_1, de rayon $R_1 \simeq 8,007$, de hauteur 4, est placé dans la scène

en utilisant la matrice de la transformation affine [Gar07] suivante :

$$\begin{pmatrix} -0,996 & 0,084 & 0,000 & -15,035 \\ 0,083 & 0,993 & 0,084 & 0,050 \\ -0,007 & -0,083 & 0,996 & 10,996 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Nous voulons joindre Cyl_1 le long du cercle C_1 tangent à la sphère S_1 de centre :

$$O_1\,(-15,035; 0,050; 10,996)$$

et de rayon $r_1 \simeq 8,007$. Le point σ_1 de Λ^4 correspondant à S_1 a pour coordonnées :

$$(17,725; -1,878; 0,006; 1,373; 17,600)$$

tandis que le vecteur $\overrightarrow{v_1}$ correspondant à C_1 a pour composantes :

$$(-2,097; -0859; -1,318; 0,000; -2,062)$$

Le cylindre Cyl_2, de rayon $R_2 \simeq 3,940$, de hauteur 5, est placé dans la scène en utilisant la matrice de la transformation affine suivante :

$$\begin{pmatrix} 0,365 & -0,330 & 0,870 & 5,606 \\ 0,645 & -0,584 & -0,492 & 8,142 \\ -0,671 & -0,742 & 0,000 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Nous voulons joindre Cyl_2 le long du cercle C_2 tangent à la sphère S_2 de centre :

$$O_2\,(5,000; 8,485; 0,000)$$

et de rayon $r_2 \simeq 3,940$. Le point σ_2 de Λ^4 correspondant à S_2 a pour coordonnées :

$$(10,250; 1,250; 2,121; 0,000; 10,000)$$

et le vecteur $\overrightarrow{v_2}$ correspondant à C_2 a pour composantes :

$$(18,016; 0,000; 0,138; 1,638; 18,016)$$

En utilisant l'algorithme 2, nous trouvons σ de coordonnées :

$$(4,410; -1,402; 1,080; 0,000; 4,354)$$

qui correspond à la sphère S de centre :

$$O_S\,(-7,071; 6,928; 0,000)$$

et de rayon $r_S \simeq 6,414$. Les deux courbes :

$$\Gamma_1 = \Lambda^4 \cap Aff(\sigma_1, \sigma, \overrightarrow{v_1})$$

et :

$$\Gamma_2 = \Lambda^4 \cap Aff(\sigma_2, \sigma, \overrightarrow{v_2})$$

correspondent aux deux cyclides Dup_1 et Dup_2.

Les paramètres de la cyclide de Dupin Dup_1 sont :

$$a_1 \simeq 10,507; \quad c_1 \simeq -1,504; \quad \mu_1 \simeq 6,503$$

tandis que la matrice de la transformation affine permettant de la représenter est :

$$\begin{pmatrix} -0,716 & 0,000 & 0,699 & -7,516 \\ -0.696 & -0,084 & -0,713 & 7,364 \\ 0,059 & -0,996 & 0,060 & 10,381 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Les paramètres de la cyclide de Dupin Dup_2 sont :

$$a_2 = 10; \quad c_2 = 2; \quad \mu_2 = 5$$

et toute la scène est représentée dans son repère.

En ne conservant que les parties utiles de ces cyclides pour la jointure, nous obtenons l'illustration de la figure 13.

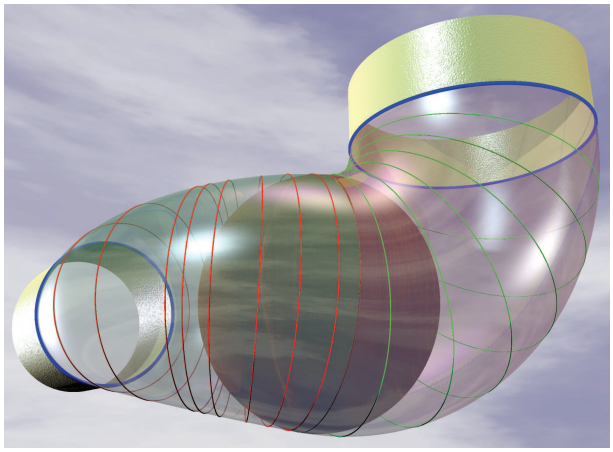


Figure 13: *Exemple de jointure entre deux cylindres dans $\mathscr{E}_3$ par deux morceaux de cyclides de Dupin, contrairement à W. Boehm, nous imposons les cercles bleus sur les cylindres.*

6 Conclusion et perspectives

Nous avons présenté deux algorithmes. Le premier traite de jointures G^1 entre un cercle sur une sphère et un autre cercle par un morceau de cyclide de Dupin. Le second, plus élaboré, permet de réaliser des jointures entre deux surfaces canal quelconques par deux morceaux de cyclide de Dupin. La résolution de ce dernier problème n'est pas directe dans l'espace usuel de dimension 3, c'est pourquoi nous nous sommes placés dans l'espace des sphères contenu dans l'espace de Lorentz de dimension 5. Dans cet espace, il suffit de considérer uniquement des courbes planes et des vecteurs tangents. En effet, la recherche de deux cyclides de Dupin effectuant la jointure entre deux surfaces canal revient à déterminer deux cercles pour la forme de Lorentz tous deux tangents au même vecteur en un même point. Par rapport

à ce qui existe, nous réalisons ces jointures sans conditions initiales sur les deux surfaces canal (condition d'intersection des axes par exemple). Nos algorithmes permettent de prendre en entrée des cercles caractéristiques sur les deux surfaces canal à joindre et non pas d'imposer le cercle de jointure entre les deux cyclides.

A l'avenir, nous souhaitons construire des triangles 3D à bords circulaires passant par trois points donnés. Pour ce faire, nous utiliserons soit des cercles caractéristiques, soit des cercles de Villarceau en passant dans l'espace des sphères. Nous souhaitons aussi réaliser des jointures entre différentes surfaces canal en utilisant des triangles ou des carreaux de cyclides de Dupin. Etant donné qu'il existe des algorithmes de subdivisions de cyclides de Dupin, nous pourrons obtenir des subdivisions de surfaces dont les sommets n'ont pas la même connectivité.

Références

[AD97a] ALLEN S., DUTTA D. : Cyclides in pure blending I. *Computer Aided Geometric Design. Vol. 14*, Num. 1 (1997), 51–75. ISSN 0167-8396.

[AD97b] ALLEN S., DUTTA D. : Cyclides in pure blending II. *Computer Aided Geometric Design. Vol. 14*, Num. 1 (1997), 77–102. ISSN 0167-8396.

[Ber78] BERGER M. : *Géométrie 2*, 2ème ed., vol. 5. Cedic-Nathan, 1978.

[BG92] BERGER M., GOSTIAUX B. : *Géométrie différentielle : variétés, courbes et surfaces*, 2ème ed. PUF, avril 1992.

[Boe90] BOEHM W. : On cyclides in geometric modeling. *Computer Aided Geometric Design. Vol. 7*, Num. 1-4 (juin 1990), 243–255.

[Cay73] CAYLEY A. : On the cyclide. *Quartely Journal of Pure and Applied Mathematics. Vol. 12* (1873), 148–165.

[Cec92] CECIL T. : *Lie sphere geometry*. Universitext, 1992.

[Dar87] DARBOUX G. : *Leçons sur la Théorie Générale des Surfaces*, vol. 1. Gauthier-Villars, 1887.

[Dar17] DARBOUX G. : *Principes de géométrie analytique*. Gauthier-Villars, 1917.

[DGL11a] DRUOTON L., GARNIER L., LANGEVIN R. : Les cyclides de Dupin et l'espace des sphères. *Revue Electronique Francophone d'Informatique Graphique. Vol. 5*, Num. 1 (2011), 41–59.

[DGL11b] DRUOTON L., GARNIER L., LANGEVIN R. : Modélisation irérative de carreaux de cyclides de dupin. In *G.T.M.G.* (Grenoble, 30 -31 mars 2011), pp. 11 – 20.

[DMP93] DUTTA D., MARTIN R. R., PRATT M. J. : Cyclides in surface and solid modeling. *IEEE Computer Graphics and Applications. Vol. 13*, Num. 1 (janvier 1993), 53–59.

[Dup22] DUPIN C. P. : *Application de Géométrie et de Méchanique à la Marine, aux Ponts et Chaussées, etc.* Bachelier, Paris, 1822.

[For12] FORSYTH A. R. : *Lecture on Differential Geometry of Curves and Surfaces*. Cambridge University Press, 1912.

[Gar07] GARNIER L. : *Mathématiques pour la modélisation géométrique, la représentation 3D et la synthèse d'images*. Ellipses, 2007. ISBN : 978-2-7298-3412-8.

[GFN04] GARNIER L., FOUFOU S., NEVEU M. : Blending of surfaces of revolution and planes by Dupin cyclides. In *Geometric Modeling and Computing [Proc. book of the 8th SIAM Conference on Geometric Design and Computing, Seattle, USA,Nov. 2003]* (Nashville, USA, 2004), Nashboro Press.

[HJ03] HERTRICH-JEROMIN U. : Introduction to Mobius differential geometry. *London Mat. Soc. Lecture note, Cambridge University Press. Vol. xii* (2003), 300.

[LO10] LANGEVIN R., O'HARA J. : Conformal arclength via osculating circles. *Commentarii Mathematici Helvetici. Vol. 85* (2010).

[LSD*] LANGEVIN R., SIFRE J., DRUOTON L., GARNIER L., PALUSZNY M. : Gluing Dupin cyclides along circles, finding a cyclide given three contact conditions. Version manuscrite.

[LW08] LANGEVIN R., WALCZAK P. : Conformal geometry of foliations. *Geom Dedicata. Vol. 132*, Num. 5 (2008), 135–178.

[Mar82] MARTIN R. R. : *Principal patches for computational geometry*. PhD thesis, Engineering Department, Cambridge University, 1982.

[PB98] PALUSZNY M., BOEHM W. : General cyclides. *Computer Aided Geometric Design. Vol. 15*, Num. 7 (1998), 699–710.

[PG10] PUECH L., GARNIER L. : Construction de triangles rectangles 3D sur des tores à collier et des cyclides de Dupin en anneau. In *Journées A.F.I.G* (Novembre 2010), Université de Bourgogne, pp. 173–182.

[Pra90] PRATT M. J. : Cyclides in computer aided geometric design. *Computer Aided Geometric Design. Vol. 7*, Num. 1-4 (1990), 221–242.

[Sha87] SHARROCK T. J. : Biarcs in three dimensions. In *Proceedings of the 2nd Conference on the Mathematics of Surfaces* (1987), Martin R., (Ed.), Clarendon Press New York, NY, USA, pp. 395–411. ISBN :0-19-853619-4.

[Smu04] SMUTNY J. : *Global radii of curvature and the biarc approximation of spaces curves : In pursuit of ideal knot shapes*. PhD thesis, EPF Lausanne, 2004.

Détection de similarités de Surfaces Paramétriques

Quoc-Viet DANG, Sandrine MOUYSSET, Géraldine MORIN

IRIT-ENSEEIHT
Université de Toulouse

Résumé

Notre travail détermine des similarités locales sur des surfaces paramétriques, en particulier, pour des surfaces de B-Splines ou NURBS. Des parties de la surface similaires à une isométrie près sont identifiées. La détection de similarités dans des objets 3D maillés a été récemment étudiée dans la littérature en vue d'applications telles que l'alignement, la segmentation, l'édition de forme, ou encore la complétion de parties cachées. Une méthode de classification des paires de points dans un espace de transformation est utilisée. Nous adaptons cette approche, en améliorant la classification par une approche spectrale. Nous appliquons ensuite les résultats obtenus à l'édition de surfaces en liant les points de contrôle correspondants aux partie similaires. Ainsi, nous obtenons une édition cohérente du modèle.

Mots-clés : Modélisation géométrique, isométrie, détection de similarités, classification spectrale, édition de forme

1. Introduction

La similarité est un phénomène fréquent dans la nature, dans les objets synthétiques ou dans l'architecture. C'est une caractéristique essentielle à laquelle les artistes s'attachent dans leurs oeuvres, que les concepteurs de forme 3D doivent régler dans leurs conceptions, et aussi auquel le visuel humain se concentre lors de la perception de beauté. Détecter ces similarités dans des modèles existants est donc une tâche importante, et conduit à de très nombreuses applications pour l'édition de formes, la complétion de partie manquantes ou encore la compression de modèles 3D. Par exemple, Chaouch et al. ont utilisé la réflexion comme une caractéristique pour aligner les modèles 3D [CVB08], Li [LYW*11] propose un système de complétion de données manquantes d'un modèle de crâne basé sur la symétrie, Mitra [MGP06] produit les modèles 3D édités en déformant les parties similaires à partir des originaux de ces modèles. En effet, la détection de similarités dans des objets 3D est un axe de recherche actuel et largement abordé [MGP06, LoE06, PSG*06, LCDF10]. D'autre part, les surfaces à poles, et en particulier les B-splines ou plus généralement les NURBS sont des modèles importants et passés dans les standards. Leur forme paramétrique permet d'accéder naturellement à des informations différentielles sur la surface.

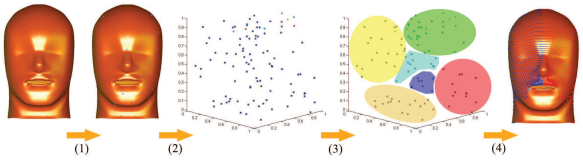


Figure 1: Pipeline – (1) échantillonage, (2) passage en espace de transformation, (3) classification, (4) validation

Les contributions de ce travail sont les suivantes. Tout d'abord, pour orienter les vecteurs des repères caractéristiques sur un point de la surface, nous proposons alors une simple méthode basée sur l'analyse du voisinage. Nous distinguons les isométries directes et indirectes. Les isométries indirectes sont ramenées au cas direct afin de simplifier le traitement données et d'obtenir des résultats plus cohérents. De plus, contrairement aux approches de l'état de l'art, notre approche de classification utilise une méthode spectrale, une heuristique non-supervisée ayant la capacité de grouper automatiquement des classes sans réglage des paramètres globaux nécessaire à l'algorithme de *Mean Shift*. A notre connaissance, il n'y a aucun recherche dédiée à

la détection de similarités dans les modèles paramétriques comme des B-Splines ou des NURBS.
Une première section détaille l'état de l'art et la section 3 suivante fixe les notations et détaille l'approche de Mitra et al. [MGP06] que nous avons suivie et modifiée. La section 4 détaille le calcul de la signature et l'orientation des vecteurs du repère correspondant. Section 5 présente la méthode de classification originale proposée. Finalement, section 6 présente les résultats obtenus.

2. État de l'art

Détection de similarités est un sujet étudié largement dans la domaine d'images numériques en 2D et aussi mais aussi en 3D pour des objets représentés par un maillage. Zabrodsky et al. [ZPA95] a proposé la *distance symétrique* comme une métrique mesurant la symétrie présente dans des objets 2D ou 3D. Cette distance est définie par le carré de la distance moyenne nécessaire de déplacement les points de l'objet original pour obtenir l'objet symétrique correspondant. Sun et ses collègues [SS97] convertissent le problème de détection de symétries en corrélation de l'image Gaussienne, les symétries rotationelle et bilatérale sont déterminées en utilisant l'histogramme des orientations. Kazhdan et al. [KCD*04] introduisent un descripteur de symétrie qui représente les symétries d'un objet 3D par rapport à tous les plans possibles passant par le barycentre de l'objet. Podolak et al. [PSG*06] généralisent cette approche pour rechercher les symétries d'un objet 3D associée à un plan arbitraire. Cette mesure était utilisée pour définir deux notions : les axes principaux de symétrie sont l'ensemble des normales des meilleurs plans de symétrie. Le centre de symétrie est l'intersection de ces trois plans.
D'autres travaux récents [Low03,LoE06,MGP06,PMW*08] utilisent une nouvelle approche dans la détection de symétrie. Cette approche utilisent des points caractéristiques. A chaque point de l'objet est associé certaines caractéristiques. Les points de mêmes caractéristiques sont mis en correspondance pour former un vote quantifiant la transformation entre ces points. Ces votes sont ensuite accumulées dans l'espace de Hough afin de déterminer les groupes dominants qui représentent potentiellement des couples de régions similaires. Cet approche efficaces et robustes sont le point de départ de notre recherche.

3. Recherche de similarité : pipeline

Étant donnée χ, une surface NURBS en produit tensoriel de bidegré (p,q) associée aux vecteurs de noeuds $\mathbf{u} = \{u_0,\ldots,u_n\}$ et $\mathbf{v} = \{v_0,\ldots,v_m\}$ et réseau de points de contrôle $\mathbf{P}$ de poids w, définie par l'équation :

$$\chi^w(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)P_{ij}^w. \qquad (1)$$

Notre travail vise à identifier des régions de cette surfaces similaires à une isométrie près. Les isométries peuvent être directes ou indirectes mais nous ne considérons pas les homothéthies, ni plus généralement les similitudes. Nous suivons et modifierons le cheminement de la méthode proposée par Mitra et al. dans [MGP06] pour les objets 3D maillés. Cette section consiste à rappeler les différentes étapes de leur algorithme basé sur un "système de votes". Cet algorithme consiste en quatre étapes comme décrit dans la figure 1.

1. **Echantillonage et signatures** A chaque point de la surface peut être associée une signature caractérisée par les propriétés locales de courbure (détaillée en section 4.1). Tout d'abord, une grille de points est obtenue en échantillonnant les paramètres du modèle en u et v. Ensuite, nous sélectionnons aléatoirement un nombre prédéfini de points sur cette grille comme points d'échantillonnage.
2. **Paires dans l'espace des transformations** Parmi les points échantillonnés sur la surface, ceux de même signature sont mis en correspondance pour former une paire. Étant donnés une paire $(\mathbf{p_i},\mathbf{p_j})$ de la surface paramétrique, la transformation de $\mathbf{p_i}$ à $\mathbf{p_j}$ est définie par $\mathbf{T_{ij}} \in \mathbb{R}^6$, dont $\mathbf{T_{ij}} = (R_{ij}^x, R_{ij}^y, R_{ij}^z, t_{ij}^x, t_{ij}^y, t_{ij}^z)$, où $[R_{ij}^x, R_{ij}^y, R_{ij}^z]$ sont les trois angles d'Euler dérivés de la matrice de rotation R_{ij} et $[t_{ij}^x, t_{ij}^y, t_{ij}^z]$ sont les trois composantes de la translation $t_{ij} = p_j - R_{ij} * p_i$. A chaque paire considérée correspond une transformation T_{ij}, un point de l'espace de transformation appelé Γ.
3. **Classification** Dans l'espace Γ, chacune de ces transformations T_{ij} représente une transformation entre deux points. Les paires de points de même transformation peuvent potentiellement correspondre à deux régions similaires de la surface. Les méthodes de segmentation classiques comme K-means ou les recherches de K plus proches voisins sont utilisées pour résoudre ce problème. Cependant, nous n'avons pas à priori aucune connaissance sur le nombre de régions similaires, ni sur le nombre de points proches d'un point donné dans l'espace de transformation . Nous proposons plutôt d'utiliser une classification spectrale non supervisée, qui permet de déterminer automatiquement les classes de façon stable sans avoir à connaître le nombre de classes recherchées. Nous détaillerons cet algorithme dans la section 5.
4. **Validation** Chacune des classes obtenues par la classification se compose de points qui sont idéalement dans deux régions similaires. Néanmoins, la cohérence spatiale entre les deux paires de points n'est pas assurée : c'est le but de cette dernière étape, dite de validation. La vérification est effectuée par un processus d'expansion de région. Étant donné C_k une classe de points dans l'espace des transformations, une pairs de points (p_i, p_j) est sélectionnée aléatoirement. En appliquant la transformation euclidienne, les coordonnées euclidiennes de 8 points proches de p_i sont ensuite comparées avec celles de p_j. Si cette comparaison est satisfaite à un seuil près, le processus est appliqué itérativement pour chacun des 8

voisins de p_i. Ce processus est répété jusqu'à ce que tous les points dans la région aient été visités. Nous conservons cette étape de validation.

La prochaine section détaille le calcul de la signature en un point pour les surfaces paramétriques considérées.

4. Calcul de signature et orientation

Dans notre étude la signature en un point de la surface paramétrique est un repère définit par la normale et les directions principales.

4.1. Cas des B-Spline et NURBS

Le calcul des signatures d'un point spécifique sur une surface B-Spline ou NURBS est basé sur les propriétés locales différentielles ; leur calcul pour les surfaces NURBS est décrit par Farin [Far92]. La *première forme fondamentale* est définie par :

$$ds^2 = Edu^2 + 2Fdudv + Gdv^2 \tag{2}$$

$$\text{où} \begin{cases} E = & E(u,v) = & \chi_u \chi_u \\ F = & F(u,v) = & \chi_u \chi_v \\ G = & G(u,v) = & \chi_v \chi_v \end{cases}$$

Le discriminant

$$D = \left\| \chi_u \wedge \chi_v \right\| = \sqrt{EG - F^2} \tag{3}$$

Dans l'équation (2), les dérivées partielles χ_u et χ_v en un point $\mathbf{x}$ engendrent le plan tangent à la surface en $\mathbf{x}$. Alors, le vecteur unitaire normal

$$n = \frac{\chi_u \wedge \chi_v}{\left\| \chi_u \wedge \chi_v \right\|} = \frac{1}{D} (\chi_u \wedge \chi_v) \tag{4}$$

qui, avec les vecteurs non normalisés χ_u, χ_v, forme un repère affine d'origine $\mathbf{x}$.
La courbure d'une région de la surface est basée sur la *deuxième forme fondamentale* :

$$\kappa \cos\phi ds^2 = Ldu^2 + 2Mdudv + Ndv^2 \tag{5}$$

$$\text{où} \begin{cases} L = & \chi_{uu} \cdot n \\ M = & \chi_{uv} \cdot n \\ N = & \chi_{vv} \cdot n \end{cases}$$

L'équation (5) exprime que, pour une direction donnée du/dv dans le plan u,v et pour un angle donné ϕ, la *deuxième forme fondamentale*, avec la *première forme fondamentale*, nous permet de calculer la courbure κ d'une courbe tracée sur une surface et dont la tangente a cette direction.
Nous introduisons deux matrices symétriques :

$$\mathcal{F}_1 = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \text{ et } \mathcal{F}_2 = \begin{pmatrix} L & M \\ M & N \end{pmatrix} \tag{6}$$

Puisque χ_u et χ_v sont linéairement indépendantes, $\mathcal{F}_1$ est toujours inversible. Donc, les courbures principales sont des valeurs propres de la matrice $\mathcal{F}_1^{-1}\mathcal{F}_2$. De plus, cette matrice possède toujours des valeurs propres réelles. Les deux valeurs propres κ_1, κ_2 sont les deux courbures principales et les deux vecteurs propres $V_1 = (\xi_1, \eta_1)^T$, $V_2 = (\xi_2, \eta_2)^T$ définissent les deux directions principales $t_1 = \xi_1 \chi_u + \eta_1 \chi_v$, $t_2 = \xi_2 \chi_u + \eta_2 \chi_v$.

Théorème 4.1 Etant données κ_1 et κ_2 les courbures principales en un point $\mathbf{p}$ d'une région de la surface χ. Alors :
- $\kappa_1, \kappa_2 \in \mathbb{R}$
- si $\kappa_1 = \kappa_2 = \kappa$, toutes les tangentes à p sont des vecteurs principaux, et le point p est appelé *point ombilic*.
- si $\kappa_1 \neq \kappa_2$, les deux vecteurs principaux correspondants sont perpendiculaires l'un à l'autre.

La signature en chaque point correspond à un repère orthonormé affine dont l'origine est ce point et les vecteurs sont le vecteur normal et les deux vecteurs principaux. Dans le cas des points ombilics les directions des axes sont mal définies, et donc nous ne considérerons pas ces points particuliers. Pour les autres points (non ombilics), les directions du repère sont bien définies. Dans le prochain paragraphe, nous expliquons comment nous déterminons le sens des vecteurs, et le calcul de la transformation, quelque peu différent de celui proposé par Mitra et al. [MGP06].

4.2. Orientation du voisinage

Etant donnés deux points p_i et p_j du modèle paramétrique avec leurs repères orthonormaux. Puisque les vecteurs tangents s'orientent vers le même sens, les repères correspondants ne sont pas cohérents pour certains cas de similarité, l'isométrie indirecte en particulier. Nous proposons alors un simple méthode pour régler ce problème.
Supposons que le repère orthonormal du point p_i est inchangé après la modification, mais p_j. Nous pouvons observer qu'il existe quatre possibilités d'orientation différents pour le repère des tangentes au point p_j. Notre méthode a pour but de déterminer l'orientation des tangentes de p_j la plus apte par rapport à celle de p_i. Cette méthode se compose des étapes suivantes :

1. Pour chaque point p_i et p_j, déterminer les huit points voisins qui le connectent, il existe alors un système de vecteurs d'origine à chaque point.
2. Projeter les vecteurs associés à p_i sur le plan tangent $(\overrightarrow{t^i_{min}}, \overrightarrow{t^i_{max}})$. Passer les vecteurs projetés en repère orthonormal engendré par ces deux vecteurs tangents.
3. Procéder l'étape 2 pour chacune de possibilité de l'orientation de deux tangentes $\overrightarrow{t^j_{min}}$ et $\overrightarrow{t^j_{max}}$ du point p_j.
4. Pour chaque système, calculer les courbures principales des huit voisins dans l'ordre correspondant. Le passage en repère orthonormal engendré par les deux vecteurs tangents nous permet d'arranger les voisins dans l'ordre. Par définition, le premier point du système associe au vecteur qui est situé dans la région positive du repère et

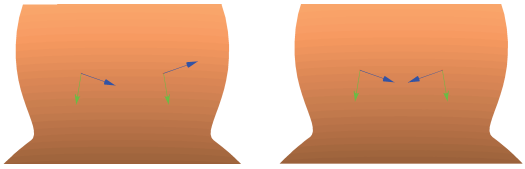


Figure 2: Avant modification (à gauche) le sens des vecteurs est donné par la paramétrisation, après modification (à droite) le sens des vecteurs est déterminé par le voisinage.

qui est au plus proche de l'orientation de $\overrightarrow{t_{min}}$. Les ordres restants sont déterminés en tournant autour de l'origine du repère depuis $\overrightarrow{t_{min}}$ vers $\overrightarrow{t_{max}}$.

5. Pour chacune des possibilités de voisins associées au point p_j, comparer les courbures correspondantes avec celles des voisins de p_i, la possibilité choisie est alors celle qui correspond à la comparaison la plus exacte.

La figure 2 représente le résultat de la modification de l'orientation du repère tangent entre les deux points symétriques sur un modèle NURBS.

5. Classification spectrale

L'étape de classification des paires de points dans un espace de transformation Γ est basée sur une approche spectrale, à savoir la classification spectrale. Le but est de regrouper les points/paires correspondant à des transformations proches (étape 3 de la section 3).

5.1. Méthode

Introduite par *Ng, Jordan, Weiss* [NJW02], la méthode de classification spectrale consiste à extraire les vecteurs propres associés aux plus grandes valeurs propres d'une matrice affinité gaussienne normalisée. Ces vecteurs propres constituent un espace propre de dimension réduite dans lequel les données projetées sont regroupées par classe. L'algorithme associé, défini par l'Algorithme 1, comprend peu d'étapes et peut être facilement codé sur Matlab. Nous rappelons que que la norme de la transformation $T \in \Gamma$ est définie par [MGP06] :

$$\|T\|^2 = \beta_1\|R\|_2^2 + \beta_2\|T\|_2^2,\ \forall T \in \Gamma, \tag{7}$$

où $\|.\|_2$ est la norme euclidienne et β_i pour $i = \{1,2\}$ sont des poids servant à ajuster l'influence relative de chaque composante dans la transformation.

Cette méthode est principalement basée sur la mesure d'affinité gaussienne, son paramètre et ses éléments spectraux. Elle présente l'intérêt d'utiliser les propriétés inhérentes aux noyaux de Mercer permettant de projeter implicitement les données dans un espace de grande dimension dans lequel les données seront linéairement indépendantes. Ainsi,

Algorithm 1 Classification spectrale

Input : Ensemble $S = \{T_{ij}\}_{i,j=1..n} \in \Gamma$, nombre de classes k.

1. Construction de la matrice affinité $A \in \mathbb{R}^{n\times n}$ définie par :
$$A_{il} = \begin{cases} \exp\left(-\frac{\|T_{ij}-T_{lm}\|^2}{(\sigma/2)^2}\right) \text{ si } (ij) \neq (lm), \\ 0 \text{ sinon.} \end{cases} \tag{8}$$
2. Construction de la matrice normalisée : $L = D^{-1}A$ avec $D_{i,i} = \sum_{r=1}^{n} A_{ir}, \forall i \in \{1,..n\}$.
3. Assembler la matrice $X = [X_1X_2..X_k] \in \mathbb{R}^{n\times k}$ formée à partir des k plus grands vecteurs propres de L.
4. Construction de la matrice Y formée en normalisant les lignes de X.
5. Traiter chaque ligne de Y comme un point de $\mathbb{R}^k$ et les classer en k classes via la méthode *K-means*.
6. Assigner le point original T_{ij} à la classe t si et seulement si la ligne i de la matrice Y est assignée à la classe t.

des classes de formes arbitraires (notamment des domaines non convexes) peuvent être définies. De plus, l'algorithme dépend uniquement de deux paramètres à savoir le paramètre d'affinité gaussienne et le nombre de classe k. Pour rendre cette méthode totalement non supervisée, nous utilisons une heuristique pour définir chaque paramètre [MNRG11].

5.2. Paramètre d'affinité

L'expression de l'affinité gaussienne, définie par l'équation (8), dépend donc d'un paramètre σ. Or, σ influe sur la séparabilité des données dans l'espace de projection spectrale. En effet, d'après *Ng, Jordan, Weiss* [NJW02], σ contrôle la similarité entre les données et conditionne la qualité des résultats. Pour garder l'efficacité de la méthode, une approche globale est privilégiée où le paramètre est fonction des distances entre les points et de la dimension du problème :

$$\sigma = \frac{max_{(ij)\neq(lm)}\|T_{ij}-T_{lm}\|^2}{2n^{1/p}}. \tag{9}$$

Cette heuristique intègre la notion de densité de points dans l'ensemble des données p-dimensionnelles et l'équation (9) donne un seuil à partir duquel des points sont considérés comme proches.

5.3. Nombre de classes

Pour déterminer le nombre de classes k, la matrice affinité gaussienne A est à nouveau exploitée. Pour une valeur de k, la matrice affinité est ordonnancée par classe. Une matrice par bloc est ainsi définie : les blocs hors diagonaux représentent les affinités entre les classes et les blocs diagnonaux les affinités intra-classes. A partir de cette structure bloc,

Figure 3: Les modèles d'expérimentation

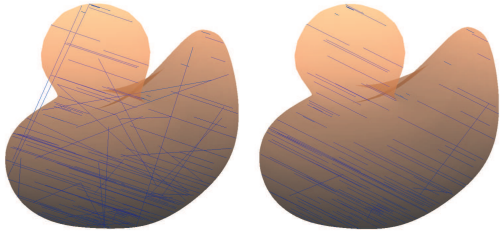


Figure 4: Comparaison entre la méthode Mean-shift (à gauche) et Spectrale (à droite)

nous pouvons évaluer un ratio moyen entre tous les blocs hors diagonaux et les blocs diagonaux en norme de Frobenius. Ainsi, parmi les valeurs de k, le minimum de ce ratio définit le nombre optimale de classe k. En effet, ce minimum correspond au cas où l'affinité entre les classes est la plus faible et l'affinité au sein des classes est la plus forte.

6. Expérimentation

Nous avons implémenté le pipeline décrit dans la section 3 pour détecter les similarités dans deux modèles NURBS synthétisés qui représentent les similarités.
La figure 4 visualise la comparaison entre les deux méthode de classification : Mean-shift [MGP06] et notre approche de Spectrale [MNRG11]. Les lignes reliant deux points de même signature représentent les points dans l'espace de transformation. Nous pouvons observer qu'il y a des bruités dans la classification Mean-shift, et que la méthode spectrale réduit ces bruités. Après l'étape de validation, les deux régions similaires sont détectées comme dans la figure 5.

7. Conclusion

Nous avons présenté un algorithme de recherche de parties similaires dans des surfaces paramétriques B-splines ou NURBS. Cet algorithme suit l'approche proposée par Mitra et al. [MGP06]. Nous proposons une mise en correspondance des points, i.e. une transformation, originale puisqu'elle se base sur une étude du voisinage des points permettant de trouver le sens des vecteurs caractéristiques. Nous

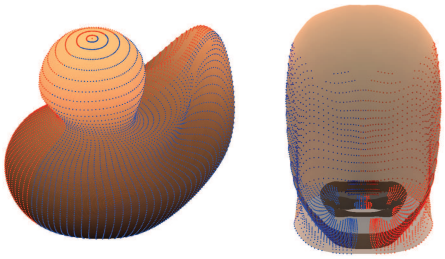


Figure 5: Deux régions symétriques sont détectées

traitons aussi les transformations indirectes de façon à se ramener à un cas tout à fait similaire à celui des transformations directes. Finalement, l'étape de classification est faite par une méthode spectrale non supervisée, permettant d'éviter le choix de paramètres inhérents à la méthode de K-means. Par la suite, nous souhaitons généraliser cette classification pour unifier les étapes de classification et de validation.

Références

[CVB08] CHAOUCH M., VERROUST-BLONDET A. : *A Novel Method for Alignment of 3D Models*. Research Report RR-6408, INRIA, 2008.

[Far92] FARIN G. : *Courbes et surfaces pour la CGAO - conception géométrique assistée par ordinateur*. Masson, 1992.

[KCD*04] KAZHDAN M., CHAZELLE B., DOBKIN D., FUNKHOUSER T., RUSINKIEWICZ S. : A reflective symmetry descriptor for 3d models, 2004.

[LCDF10] LIPMAN Y., CHEN X., DAUBECHIES I., FUNKHOUSER T. : Symmetry factored embedding and distance. *ACM Trans. Graph.. Vol. 29* (July 2010), 103 :1–103 :12. Another approach for detecting symmetry.

[LoE06] LOY G., OLOF EKLUNDH J. : Detecting symmetry and symmetric constellations of features. In *In ECCV* (2006), pp. 508–521.

[Low03] LOWE D. G. : Distinctive image features from scale-invariant keypoints, 2003.

[LYW*11] LI X., YIN Z., WEI L., WAN S., YU W., LI M. : Cultural heritage : Symmetry and template guided completion of damaged skulls. *Comput. Graph.. Vol. 35* (August 2011), 885–893.

[MGP06] MITRA N. J., GUIBAS L., PAULY M. : Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH). Vol. 25*, Num. 3 (2006), 560–568.

[MNRG11] MOUYSSET S., NOAILLES J., RUIZ D., GUIVARCH R. : On a strategy for spectral clustering with parallel computation. *High Performance Computing for Computational Science–VECPAR 2010* (2011), 408–420.

[NJW02] NG A., JORDAN M., WEISS Y. : On spectral clustering : Analysis and an algorithm. *Advances in neural information processing systems. Vol. 2* (2002), 849–856.

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L. J. : Discovering structural regularity in 3d geometry, 2008.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T. : A planar reflective symmetry transform for 3d shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH). Vol. 25*, Num. 3 (July 2006).

[SS97] SUN C., SHERRAH J. : 3-d symmetry detection using the extended gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 19* (1997), 164–168.

[ZPA95] ZABRODSKY H., PELEG S., AVNIR D. : Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 17* (1995), 1154–1166.

Calcul de témoins pour la résolution de systèmes de contraintes géométriques

Arnaud Kubicki [1], Dominique Michelucci [1] et Sebti Foufou [1,2]

[1]Université de Bourgogne
[2]Qatar University

Résumé

In geometric constraint solving, the constraints are represented with an equation system $F(U,X) = 0$, where X denotes the unknowns and U denotes a set of parameters. The target solution for X is noted X_T, the associated parameters are U_T. A witness is a couple (U_W, X_W) such that $F(U_W, X_W) = 0$. The witness is not the target solution, but they share the same combinatorial features, even when the witness and the target lie on two distinct connected components of the solution set of $F(U,X) = 0$. Thus a witness enables the qualitative study of the system : the detection of over- and under-constrained systems, the decomposition into irreducible subsystems, the computation of subsystems boundaries.
This paper investigates the witness computation in various configurations. The witness computation will be studied under several numerical methods : Newton iterations from random seeds, the BFGS method and the Nelder-Mead simplex method.The robustness and performances of these methods will be analyzed and compared.

En résolution de contraintes géométriques, les contraintes sont représentées par un sytème d'équations $F(U,X) = 0$, où X est l'ensemble des inconnues et U celui des paramètres. La solution recherchée pour X est notée X_T, les paramètres associés U_T . Un témoin est constitué du couple (U_W, X_W) tel que $F(U_W, X_W) = 0$, en général (U_W, X_W) est différent de (U_T, X_T). Le témoin n'est pas la solution recherchée mais partage les mêmes propriétés combinatoires, y compris si le témoin et la solution sont dans deux composantes connexes différentes dans l'ensemble des solutions de $F(U,X) = 0$. Ainsi, un témoin permet une étude qualitative du système : la détection des systèmes sous-contraints ou sur-contraints, la décomposition en systèmes irréductibles et le calcul des frontières des sous-systèmes.
Nous étudions ici le calcul de témoins pour différentes configurations. Le calcul de témoins sera étudié avec différentes méthodes : la méthode de Newton avec initialisation aléatoire, la méthode BFGS et la méthode de Nelder-Mead (simplexe "marchant")La robustesse et la performance de ces méthodes seront analysées et comparées.

Mots-clés : Résolution de contraintes géométriques, Calcul de témoins, Algorithmes d'analyse numérique

1. Introduction

Dans le domaine de la conception assistée par ordinateur, la résolution de systèmes de contraintes géométriques permet de générer des formes qui satisfont des relations spatiales données par l'utilisateur : au lieu de donner explicitement les coordonnées des points, lignes, plans ou autres objets géométriques, l'utilisateur dessine une esquisse sur laquelle il spécifie, avec des outils dédiés, des relations géométriques entre les objets : des distances, des angles, des incidences et ainsi de suite. Une configuration est un modèle géométrique composée des entités contraintes ou d'un sous-ensemble de

celles-ci. Les solutions se composent des coordonnées de ces objets géométriques, c'est à dire d'une configuration qui satisfait les contraintes.

Un système de contraintes géométriques se compose d'un triplet $\mathcal{S} = (C,X,U)$ où X est l'ensemble des inconnues (objets géométriques), U l'ensemble des paramètres (valeur des contraintes et coordonnées données par l'utilisateur) et C l'ensemble des contraintes. En ce qui concerne les logiciels de résolution d'équations, on représente le système de contraintes géométriques comme un ensemble d'équations F et le processus de résolution consiste alors à rechercher les racines du sytème $F(U,X) = 0$.

Un système de contrainte géométrique peut être sous-contraint (il y a une infinité de solution parce qu'il n'y a pas assez de contraintes), sur-contraint (il n'y a aucune solution à cause de dépendances directes, structurelles, indirectes ou des incohérences) ou bien contraint (il y a un ensemble fini non-vide de solution). Un système est dit consistant sur-contraint quand il est génériquement sur-contraint mais que les valeurs des paramètres sont telles qu'il y a des solutions. Remarquons qu'un système rigide est sous-contraint puisque ces solutions sont invariantes par translation ou rotation : pour cette raison, on considère les systèmes bien contraint *modulo* les transformations rigides. Pour une définition plus formelle d'un système de contraintes géométriques et des niveaux de constriction, le lecteur peut se référer à [MT10].

La résolution de contraintes géométriques est un domaine de recherche actif depuis plusieurs années [BR98, GM06]. Un état de l'art des techniques de base largement utilisées pour résoudre des systèmes de contraintes géométriques 2D et 3D est disponible dans [HJA05]. Un état de l'art des techniques de décompositions se trouve dans [JTNM06].

Pour tenir compte des spécificités des systèmes de contraintes géométriques, les algorithmes de résolution proposés, appelés solveurs, varient selon les aspects suivants :
- les étapes de résolutions (détections de dépendances, correction, décomposition, résolution de sous-systèmes *etc.*),
- les techniques de résolution sous-jacentes (méthodes classiques de l'analyse numérique, *e.g.* Newton-Raphson, méthodes de graphe, méthodes probabilistes, *etc.*),
- les interactions avec l'utilisateur en cours de résolution (algorithmes autonomes et semi-autonomes) ;
- les solutions fournies à l'utilisateur (une configuration, plusieurs ou toutes).

Détecter les dépendances directes et indirectes dans un système de contraintes géométriques est un problème difficile pour lequel de nombreux algorithmes échouent. En particulier, les solveurs basés sur les graphes sont incapables de détecter les dépendances non-structurelles qui sont la conséquence de théorèmes de géométrie. La méthode du témoin proposée dans [MF06] contourne habilement les limitations des méthodes de graphes et détecte toutes les dépendances entre les contraintes. Le calcul d'une base des déplacements infinitésimaux d'un témoin typique est expliqué dans [MF09], ainsi que l'utilisation d'une telle base pour interroger le témoin et détecter toutes les dépendances. D'autres problèmes importants en résolution de contraintes géométriques, comme la détection du problème bien contraint maximum ou du calcul d'une base bien-contrainte d'un système sur-contraint consistant sont abordées dans [MST*10] et utilisent la méthode du témoin.

Cependant, le plus gros inconvénient de la méthode du témoin est l'obtension d'un témoin typique, c'est à dire une solution d'un système de contraintes géométriques, avec les mêmes ensembles C et U mais des valeurs différentes des paramètres. Le témoin doit être typique, c'est à dire qu'il doit avoir les mêmes propriétés combinatoires que les solutions. La plupart du temps, l'esquisse dessinée par l'utilisateur est un témoin, mais quand le système comporte beaucoup de contraintes d'incidence (que doit satisfaire le témoin) ou quand l'utilisateur place les objets dans une configuration singulière, on peut avoir besoin de calculer un témoin.

Cet article complète la méthode du témoin en comparant différentes manières de calculer un témoin typique : itération de Newton avec initialisation aléatoire, la méthode BFGS, la méthode du simplexe de Nelder et Mead, les projections itérées et les méthodes par intervalles.

Un solveur par intervalle complet, robuste et avec certification est décrit dans [FMF09] et utilisé dans [MST*10]. Si un tel solveur ne trouve aucune solution, cela prouve la non-existence de solutions. Seuls les solveurs complets garantissent cette propriété et ce degré de robustesse. Cependant, la plupart du temps, l'utilisation de ce type de solveur est excessive : le système considéré est souvent très sous-contraint avec quelques sous-systèmes localement sur-contraints et redondants donc des méthodes plus simples et plus rapides pourraient suffire.

L'article est organisé comme suit : la partie §2 rappelle la définition d'un témoin et montre en quoi le calcul est important et parfois difficile. La partie §3 montre un ensemble de méthodes que l'on a implémentées pour pouvoir comparer leur performance dans la génération de témoins. Les parties §4 et §5 présentent des exemples 2D et 3D avec le calcul des témoins respectifs. La partie §8 étudie, à travers deux exemples, l'utilisation de la méthode du témoin pour détecter les dépendances entre les valeurs des paramètres. Pour finir, la partie §9 donne quelques pistes pour compléter ce travail.

2. La méthode du témoin

Soit un système d'équation représentant des contraintes géométriques, où les contraintes sont représentées par un ensemble d'équations $F(U,X) = 0$, où U représente un ensemble de paramètres dont les valeurs associées aux solutions cherchées sont $U_T = \rho(U)$, et X représente les inconnues. La solution cherchée pour X est notée X_T. Un témoin est un couple (U_W,X_W) tel que $F(U_W,X_W) = 0$; la plu-

part du temps, U_W et U_T sont différents (de même que X_W et X_T), donc le témoin (U_W, X_W) n'est pas la solution désirée, mais il partage les mêmes propriétés combinatoires que (U_T, X_T), même quand le témoin et la solution cherchée n'appartiennent pas à la même composante connexe de l'espace des solutions de $F(U,X) = 0$. Ainsi un témoin permet l'étude qualitative du système : la détection des systèmes sous ou sur-contraints, et la décomposition en sous-systèmes irréductibles. Ces aspects sont détaillés dans les articles [MF06, MF07, MF09, MST*10]. De plus, le témoin peut être calculé dans $\mathbb{C}$ car la détection des dépendances se fondent sur un calcul de rang.

Le principal avantage de la méthode du témoin, comparé aux autres méthodes combinatoires (méthodes de graphe) est que le témoin peut détecter toutes les dépendances entre contraintes, et pas seulement les plus simples (dépendances structurelles), qui sont déjà détectées par les méthodes à base de graphe, mais aussi les dépendances non-structurelles qui sont la conséquence de propriétés ou théorèmes géométriques, connus ou non.

Donc un témoin typique doit pouvoir être calculé. Nous devons répondre à une première question : quel système allons nous résoudre ?

Une première idée est de résoudre un système de contraintes géométriques et de considérer le système $F(U,X) = 0$, où à la fois U et X sont considérés comme des inconnues. C'est la méthode utilisée dans [MST*10], où le système est résolu par analyse d'intervalle.

Une seconde idée est de résoudre directement le système cible $F(U_T, X) = 0$. Ce système est en général plus contraint que le système $F(U,X) = 0$, on peut donc faire l'hypothèse qu'il sera plus difficile à résoudre.

Une autre idée est possible : un témoin doit au moins vérifier les contraintes d'incidences et cela est souvent suffisant. Cela n'est pas suffisant quand un même paramètre est utilisé plusieurs fois pour par exemple spécifier un triangle isocèle ou équilatéral. Bien sûr le système ainsi réduit est plus petit et plus facile à résoudre. Les contraintes d'incidence sont également appelées contraintes projectives (la géométrie projective étudie les propriétés projectives, c'est à dire qui sont invariantes par projection). En 2D, les contraintes d'incidence sont : 3 points alignés, 4 points cocycliques, la tangence, les incidences point-courbe. En 3D, il s'agit de la coplanarité de 4 points, 5 points sur une même sphère, *etc.*

Remarquons que les contraintes d'incidence peuvent être formulées en équations de différentes manières : *e.g.* la colinéarité de 3 points A, B, C en 2D peut être exprimée comme l'annulation du déterminant

$$\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} = 0$$

ou comme le sous-système :

$$\begin{cases} (x_A, y_A, 1) \,.\, (a,b,c) = 0 \\ (x_B, y_B, 1) \,.\, (a,b,c) = 0 \\ (x_C, y_C, 1) \,.\, (a,b,c) = 0 \\ a^2 + b^2 - 1 = 0 \end{cases}$$

où la ligne ABC a l'équation : $ax + by + c = 0$, et le vecteur (a,b) est normé. Cependant d'autres formulation sont possibles : coordonnées non cartésiennes, ou formulation ne dépendant pas des coordonnées (voir ci-dessous). Quelque soit la formulation, supprimer les équations dépendant des paramètres est le moyen le plus simple d'obtenir un système ne comportant que des contraintes projectives. Cette méthode parait raisonnable : les systèmes de contraintes d'incidence peuvent être sur-contraint (ce qui peut paraître non-intuitif), et un système sur-contraint redondant sera détecté en étudiant le témoin. Pour un système sur-contraint contradictoire, il n'y a pas de solution donc pas de témoin : un solveur complet (par exemple un solveur basé sur les intervalles) est nécessaire pour traiter ce cas.

Cependant, supprimer les équations dépendant des paramètres peut être excessif, puisqu'une combinaison linéaire de contraintes de distance (*i.e.* des équations impliquant des paramètres) implique parfois des contraintes projectives. Les deux exemples suivants montre ce problème en 2D et en 3D. En 2D, si 6 points M_i sont contraints par : $M_iF_1^2 + M_iF_2^2 = d^2$ (ou $M_iF_1^2 - M_iF_2^2 = d^2$), alors ils sont sur la même ellipse (ou la même hyperbole), alors que ce n'est pas vrai dans le cas général ; d'après le théorème de Pascal, si 6 points sont sur une même conique, alors les cotés opposés de l'hexagone $M_1, \ldots M_6$ se coupent en 3 points colinéaires (c'est vrai pour n'importe quelle permutation de $M_1, \ldots M_6$ bien sûr). Ainsi supprimer toutes les équations $M_iF_1^2 + M_iF_2^2 = d$ donne un système moins contraint (au sens projectif) que le système initial. En 3D, la colinéarité de 3 points A, B, C peut être définie par l'annulation du determinant de Cayley-Menger [MF04] :

$$\begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & D_{AB} & D_{AC} \\ 1 & D_{AB} & 0 & D_{BC} \\ 1 & D_{AC} & D_{BC} & 0 \end{vmatrix} = 0$$

où D_{AB}, D_{AC}, D_{BC} sont les carrés des distances AB, AC, BC. Clairement, supprimer ces contraintes (si les distances sont des paramètres, et le reste des inconnues) supprime également les contraintes d'incidence. Pour finir, mise à part les incidences, deux invariants numériques sont conservés en géometrie projective, en 2D le birapport réel des longueurs signées de quatre points colinéaires et le birapport complexe de quatres points coplanaires, avec coordonnées dans $\mathbb{C}$. Supprimer les équations impliquant de tels paramètres est une autre façon d'obtenir un système moins contraint (au sens projectif) que le système initial.

3. Méthodes

Cette partie présente cinq méthodes qui peuvent être utilisées pour générer un témoin : la méthode de Newton-Raphson, le simplexe de Nelder et Mead [NM65], la méthode BFGS, les projections itérées [PFTV92] et l'homotopie [LM96]. On présente aussi une méthode complète : la méthode de Newton par intervalles.

3.1. La méthode de Newton

Avec un bon point de départ, le principal avantage de la méthode de Newton-Raphson est sa vitesse de convergence. C'est une méthode aisée à programmer. Pour appliquer la méthode de Newton quand la matrice Jacobienne n'est pas inversible ou non carrée, on calcule une pseudo-inverse à l'aide d'une décomposition en valeurs singulières. On utilise la formule suivante :

$$X_{n+1} = X_n - J_n^{-1} F(X_n)$$

J_n^{-1} étant l'inverse (ou la pseudo-inverse) de la jacobienne de F au point X_n

3.2. La méthode du simplexe de Nelder et Mead

La méthode de Nelder-Mead (méthode NMS) [NM65] est une méthode de minimisation. Elle procède en deux étapes pour trouver un zéro de F : elle minimise $\|F\|$ puis elle vérifie que $\|F\|$ vaut zero (dans le but d'éviter les minima locaux).

Cette méthode permet d'éviter le calcul de la dérivée, mais n'est pas efficace en terme de nombre d'évaluation de la fonction (un nombre important d'itération est nécessaire). Étant donné un simplexe de départ (calculé à partir d'un point de départ aléatoire), l'algorithme calcule le minimum avec les étapes suivantes :

- il calcule le maximum de la fonction sur les sommets du simplexe, puis il prend le symétrique de ce simplexe par rapport à l'hyperplan opposé au sommet maximum.
- il peut agrandir le simplexe, ou bien
- il contracte le simplexe suivant une ou plusieurs directions

La combinaison de ces opérations fait "marcher" le simplexe jusqu'à un minimum local. Les détails de l'algorithme peuvent être trouvés dans [PFTV92].

3.3. La méthode BFGS

La méthode BFGS (Broyden, Fletcher, Goldfarb et Shanno) est aussi une méthode de minimisation. On utilise un procédé identique à la méthode de NMS. Cette méthode est une variante de la méthode de Newton, et nécessite le calcul du gradient pour construire une approximation de la matrice hessienne de $\|F\|$, *i.e.* la matrice H telle que :

$$\forall (i,j) \in [0,n-1]^2 \quad H(i,j) = \frac{\partial^2 \|F\|}{\partial x_i \partial x_j}$$

L'approximation est construite de façon à fournir une matrice définie positive ce qui garantit la descente vers un minimum local de $\|F\|$ [PFTV92]. Elle est mise à jour à chaque itération.

3.4. La méthode des projections itérées

L'idée de cette méthode est de projeter le point de départ sur les hypersurfaces définies par chaque équation. Soit $S_i = \{X | F_i(X) = 0\}$ la surface associée à l'équation $F_i(X) = 0$. Le premier point est projeté sur la surface d'équation S_1, le résultat est projeté sur S_2 et ainsi de suite. Avec cette méthode, le calcul de l'inverse de la jacobienne n'est pas nécessaire. Par contre on doit toujours calculer le gradient. Elle est facile à programmer mais la convergence n'est pas aussi rapide qu'avec la méthode de Newton.

3.5. Méthode de Newton par intervalles

La méthode de Newton par intervalles permet de trouver toutes les solutions d'un système d'équations comprises dans une boite. On se limite à un nombre fini d'itérations. Dans la suite, X désigne des vecteurs numériques et $[X]$ des vecteurs d'intervalles (*i.e.* des boites). Plutôt que d'inverser une matrice d'intervalles, on inverse la matrice jacobienne au centre de l'intervalle étudié. La formule utilisée est fondée sur l'itération de Newton. Soit $N : X \to X - J^{-1}F(X)$. La formule $[X_{n+1}] = [X_n] - J_n^{-1} F([X_n])$ évaluée par arithmétique d'intervalles naïve ne peut jamais donner une boite incluse dans X_n même quand N est contractante. Pour remédier à cela, on utilise l'évaluation centrée :

$$[X_{n+1}] = N(X_c) + \partial_X N([X_n])([X_n] - X_c)$$

X_c étant le centre de $[X_n]$ et $\partial_X N$ la différentielle de N. De plus, les solutions cherchées appartiennent aussi bien à X_n qu'à X_{n+1}, on choisit donc $[X_{n+1}] = (N_c([X_n])) \cap [X_n]$ où N_c désigne la version centrée de l'itération de Newton. Si on ne réduit pas assez l'intervalle étudié, on le divise puis on étudie chacune de ces parties séparément.

Cette méthode est utile, par exemple, pour détecter les cas où il n'existe pas de solutions.

3.6. L'homotopie

Le principe de la méthode par homotopie (appelée aussi continuation) est la suivante : soit un système bien contraint $F(X) = 0$, et soit X_0 la valeur associé à l'esquisse, c'est à dire la valeur initiale de X si l'on utilise la méthode de Newton. L'homotopie est définie ainsi :

$$H(t,X) = F(X) - (1-t)F(X_0)$$

et le lieu des $H(t,X) = 0$ est une courbe qui passe par X_0 en $t = 0$; on peut le montrer avec le théorème des fonctions implicites, ou le lemme de Sard. L'idée de l'algorithme est de suivre la courbe par un procédé de prédiction-correction (la méthode de Newton par exemple), ou une approximation

linéaire par morceaux. Si un point de la courbe homotopique avec $t = 1$ et $X = X_1$ est atteint, alors X_1 est la solution cherchée [SW05].

La méthode de l'homotopie peut être utilisée de deux façons pour résoudre des contraintes géométriques.

Pour la première approche, Lamure et Michelucci utilisent l'homotopie dans [LM95, LM96] comme variante de la méthode de Newton ; ils ont remarqué que les itérations de Newton ne convergent parfois pas vers la racine la plus proche intuitivement du point de départ, typiquement l'esquisse de l'utilisateur. À ce titre, la méthode par homotopie donne de meilleurs résultats que la méthode de Newton, ce qui est dû au fait que les bassins d'attraction pour la méthode de Newton sont fractals alors qu'ils ne le sont pas pour l'homotopie.

Dans la seconde méthode, Durand et Hoffmann utilisent l'homotopie dans [Dur98, DH00] pour calculer l'ensemble des racines d'un système d'équations algébriques. Le principe est de calculer un système facile à résoudre et de même degré algébrique (par exemple le même nombre de Bézout), de calculer les racines de celui-ci, et de suivre les courbes homotopiques associées dans l'espace des complexes de $t = 0$ à $t = 1$. Par exemple, si le système à résoudre comporte 2 équations polynômiales de degré 3 et 2 inconnues (une interpretation géométrique serait l'intersection de deux cubiques), alors un exemple de système de départ serait :

$$\begin{cases} (a_1x+b_1y+c_1)(a_2x+b_2y+c_2)(a_3x+b_3y+c_3) &= 0 \\ (a'_1x+b'_1y+c'_1)(a'_2x+b'_2y+c'_2)(a'_3x+b'_3y+c'_3) &= 0 \end{cases}$$

Les coefficients $a_i, b_i, c_i, a'_i, b'_i, c'_i$ sont des flottants quelconques (pseudo-aléatoires) et les racines sont facilement calculées par la résolution de 3^2 systèmes linéaires :

$$\begin{cases} a_ix+b_iy+c_i &= 0 \\ a'_jx+b'_jy+c'_j &= 0 \end{cases}$$

avec $i, j \in \{1,2,3\}$

Ainsi la méthode homotopique est un solveur complet dans le cas des systèmes algébriques. Dans le cas des systèmes non-algébriques, il n'y a pas d'algorithme connu pour construire un système de départ équivalent au système à résoudre. Pour cette raison, on préfère utiliser un solveur par intervalles quand on a besoin d'un solveur complet. De plus, il est plus facile de tenir compte des inégalités avec un solveur basé sur les intervalles. Pour finir, nous n'utilisons pas l'homotopie comme une variante de la méthode de Newton : cela a déjà été réalisé dans [LM95, LM96] et nous ne sommes pas dans un contexte interactif dans lequel l'utilisateur donne un point de départ raisonnable.

4. Résultats en 2D

Cette section donne quelques exemples de système de contraintes géométriques et les résultats respectifs de certaines des différentes méthodes mentionnées dans la partie 3, à savoir la méthode de Newton (et sa variante dans $\mathbb{C}$), la méthode NMS et la méthode BFGS.

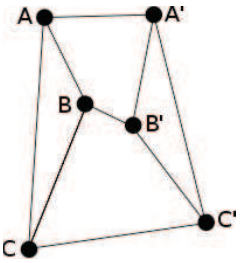


Figure 1: *Système rigide 2D fait de deux triangles liés par trois contraintes de distances*

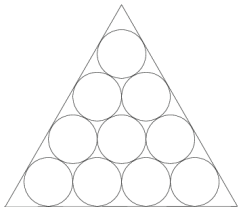


Figure 2: *Un exemple de pavage par des cercles. Pour simplifier, le triangle est équilatéral et tous les cercles ont le même rayon.*

4.1. Deux rigides

Cet exemple 2D est constitué de deux triangles ABC et $A'B'C'$; chacun d'entre eux est spécifié par la longueur de ces trois arêtes, voir Fig. 1. L'assemblage de ces deux triangles est rendu rigide par l'ajout de 3 contraintes de distances : AA', BB', CC'. Il suffit de prendre six points au hasard en 2D pour obtenir un témoin typique.

4.2. Pavage par des cercles

Le pavage par des cercles est une configuration en 2D constituée de cercles tangents les uns aux autres, voir la figure 2 pour un exemple. Le pavage par des cercles est une approximation d'une fonction analytique utile pour les systèmes embarqués et les marches aléatoires [Ste03]. Tout plan 2D, ou complexe simpliciel 2D, peut être representé par un continuum de pavage par des cercles : chaque sommet est representé par un cercle, et chaque arête AB dans le plan 2D, ou le complexe simpliciel 2D, signifie que les 2D cercles associés aux sommet A et B sont tangents par l'extérieur.

Le contour extérieur du pavage possède beaucoup de degrés de liberté. Par exemple il est possible de trouver un pavage de cercles dans lequels tous les cercles extérieurs sont tangents à un même cercle circonscrit. Cette propriété est l'analogie discrète du théorème de Riemann : il y a une transformation conforme entre tout disque ouvert (au sens topologique) et le disque unité ouvert. Certains algirithmes spécifiques [CS03] convergent de façon itérative vers le pavage de cercles de n'importe quel complexe simplicial 2D. Ces algorithmes sont suffisamment rapides pour permettre l'interaction, même avec des milliers de cercles. Ils sont bien plus rapides qu'un solveur générique.

Pour simplifier, nous allons paver des triangles avec des cercles, comme sur la figure 2. Soit k le nombre de cercles tangent à un coté du triangle. On génère le système de

méthode	taux de succés	temps total
Newton	100%	0.24 s
Newton dans $\mathbb{C}$	100 %	0.26 s
BFGS	100 %	3.88 s
NMS	0 %	3.88 s

Table 1: *problème des 2 rigides*

k	méthode	taux de succés	temps total
1	Newton	98%	0.42 s
	Newton dans $\mathbb{C}$	100 %	0.72 s
	BFGS	5 %	6.08 s
	NMS	0 %	1.82 s
2	Newton	68%	1.34 s
	Newton dans $\mathbb{C}$	82 %	1.42 s
	BFGS	0 %	14.7 s
	NMS	0 %	4.88 s
3	Newton	36%	3.3 s
	Newton dans $\mathbb{C}$	86 %	2.5 s
	BFGS	0 %	25.88 s
	NMS	0 %	10.56 s
4	Newton	14%	6.46 s
	Newton dans $\mathbb{C}$	78 %	5.06 s
	BFGS	5 %	45.62 s
	NMS	0 %	24.24 s
5	Newton	4%	11.46 s
	Newton dans $\mathbb{C}$	82 %	8.9 s
	BFGS	0 %	70.84 s
	NMS	0 %	47.96 s

Table 2: *Pavage par des cercles*

contraintes géométriques pour $k = 1,2,3,4,5$, où il y a 1, 3, 6, 10 et 15 cercles au total. Les paramètres sont les coordonnées des trois sommets du triangle, le rayon et les coordonnées des centres des cercles ; quand on calcule un témoin ce sont des inconnues. Nous ne vérifions pas que les cercles calculés sont à l'intérieur du triangle, ni qu'ils sont tangents sur leur extérieur puisque ce n'est pas requis pour un témoin.

Ici, l'algorithme le plus efficace est l'algorithme de Newton. On remarque également que celui-ci est beaucoup plus efficace dans les complexes pour ce problème (voir tableau 2).

4.3. La configuration de Desargues en 2D

Le théorème de Desargues affirme que si les points 2D o, p_i, q_i sont colinéaires pour $i = 0,1,2$, alors les points d'intersection $p_i p_{i+1 \bmod 3} \cap q_i q_{i+1 \bmod 3}$ sont également colinéaires. Ce théorème s'étend en 3D.

Le but de cet exemple est de trouver les points o, p_i, q_i d'une configuration de Desargues. Il y 9 alignements dus aux hypothèses et un dû à la conclusion. En fait, n'importe lequel de ces alignements est une conséquence des neuf autres. Le système est très sous-contraint (aucune distance et aucun angle n'est donné), mais sur-contraint (dans le sens les contraintes sont redondantes).

Deux possiblités nous sont offertes pour spécifier ces alignements :

(i). annuler le déterminant, ou bien

(ii). définir chaque droite par un triplet (a,b,c) avec $a^2+b^2 = 1$, et le point (x,y) est sur la droite si et seulement si $ax+by+c = 0$.

La formulation qui utilise les déterminants (cf tableau 4) donne de bien meilleurs résultats pour chacune des méthodes testées. Toutes les méthodes fournissent un témoin, la plus rapide étant la méthode de Newton dans $\mathbb{R}$.

4.4. Configuration de Pappus

Le théorème de Pappus affirme que si les points p_1, p_2, p_3 sont colinéaires et les points q_1, q_2, q_3 sont colinéaires, alors les 3 intersections $r_{ij} = p_i q_j \cap p_j q_i, i \neq j$ sont colinéaires.

Dans cet exemple, on cherche les points p_i, q_j, r_{ij} d'une configuration de Pappus ; les neuf alignements sont donnés, huit sont dus aux hypothèses et le neuvième est la consé-

méthode	taux de succés	temps total
Newton	91%	6.28 s
Newton dans $\mathbb{C}$	97 %	6.28 s
BFGS	0 %	24.56 s
NMS	0 %	16.78 s

Table 3: *Problème de Desargues (formulation avec des droites)*

méthode	taux de succés	temps total
Newton	100%	0.54 s
Newton dans $\mathbb{C}$	100 %	0.78 s
BFGS	100 %	12.08 s
NMS	100 %	1 s

Table 4: *Problème de Desargues (formulation avec des déterminants)*

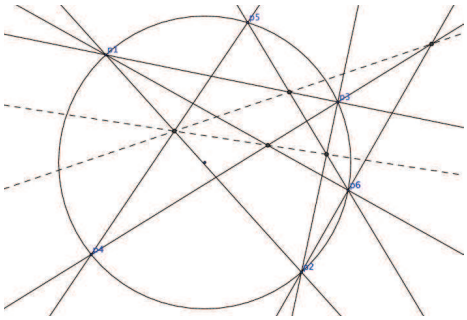


Figure 3: *Un exemple de configuration de Pascal.*

quence du théorème. Le système est trés sous-contraint (pas de contrainte d'angle ni de contraintes de distance), mais aussi sur-contraint car un alignement est la conséquence des huit autres. Compte tenue de la symétrie du problème, n'importe lequel de ces alignements peut être considéré comme une conséquence des huit autres.

Là aussi, la méthode de Newton s'avère la plus efficace. Les mêmes remarques que la partie 4.3 s'appliquent quant au choix de la représentation de ces contraintes.

4.5. Configuration de Pascal

D'aprés le théorème de Pascal, si six points sont sur une même conique, les cotés opposés de l'hexagone (pour n'importe quelle permutation des sommets) se coupent en trois points qui sont colinéaires. Une autre formulation est la suivante : si un hexagone en 2D (convexe ou non, avec auto-intersection ou non) $p_0p_1p_2p_3p_4p_5$ est tel que ses cotés opposés se coupent en trois points alignés $p_0p_1 \cap p_3p_4$, $p_1p_2 \cap p_4p_5$ et $p_2p_3 \cap p_5p_6$, alors la propriété est valide pour toutes les permutations des points $p_0p_1p_2p_3p_4p_5$.

Dans les tests, nous avons généré les contraintes pour chaque hypothèse du théorème, ainsi que la contrainte de la conclusion. Les contraintes d'alignement sont exprimées comme l'annulation du déterminant 4×4 des coordonnées homogènes des trois points considérés. Le système est redondant.

À l'exception de la méthode NMS, tous les algorithmes fournissent un témoin, la méthode de Newton étant la plus rapide.

5. Résultats en 3D

Dans cette partie, nous étudions des exemples de contraintes géométriques en 3D avec les résultats des différentes méthodes que nous avons programmées et testées (les mêmes que pour la partie 4.

5.1. Configuration de Desargues en 3D

La configuration de Desargues en 3D est similaire au problème 2D : les points 3D o, p_i, q_i sont colinéaires, pour $i = 0,1,2$, mais les plans $p_0p_1p_2$ et $q_0q_1q_2$ sont différents. Dans le cas général, ces 2 plans se coupent selon une droite, qui contient les trois points d'intersection $p_ip_{i+1 \bmod 3} \cap q_iq_{i+1 \bmod 3}$ entre les cotés homologues des triangles. Ainsi ces trois points d'intersection sont colinéaires. Il est possible de formuler le théorème de façon plus générale, pour tenir compte des cas dégénérés (les plans p_i et q_i sont parallèles ou identiques, ou bien l'une des droites $p_ip_{i+1 \bmod 3}$ est parallèle à son homologue $q_iq_{i+1 \bmod 3}$, etc.), mais on préfère ne pas les considérer pour simplifier.

Nous avons généré les contraintes spécifiant les hypothèses et la conclusion du théorème. Ces contraintes sont projectives, *i.e.* elles se composent uniquement de contraintes d'incidence : il n'y a donc pas de paramètre de distance ou d'angle.

Nous avons représenté les contraintes de colinéarité de trois points avec des determinants de deux façons différentes. La première formulation utilise l'annulation de deux determi-

méthode	taux de succés	temps total
Newton	100%	1.94 s
Newton dans $\mathbb{C}$	100 %	2.66 s
BFGS	17 %	6.62 s
NMS	0 %	13.48 s

Table 5: *Configuration de Pappus (droites)*

méthode	taux de succés	temps total
Newton	100%	0.26 s
Newton dans $\mathbb{C}$	100 %	0.2 s
BFGS	100 %	3.14 s
NMS	0 %	3.36 s

Table 6: *Configuration de Pappus (déterminants)*

nants :

$$\begin{vmatrix} x_1-x_0 & x_2-x_0 \\ y_1-y_0 & y_2-y_0 \end{vmatrix} = \begin{vmatrix} x_1-x_0 & x_2-x_0 \\ z_1-z_0 & z_2-z_0 \end{vmatrix} = 0$$

La deuxième utilise aussi la contrainte ci-dessous :

$$\begin{vmatrix} y_1-y_0 & y_2-y_0 \\ z_1-z_0 & z_2-z_0 \end{vmatrix} = 0$$

Le système est alors redondant pour chaque alignement. Pour ce problème, toutes les méthodes fournissent un témoin, avec un net avantage en terme de performance pour la méthode NMS.

5.2. Configuration de Beltrami

Soient $B_i, i = 0,1,2$ trois droites blanches non-coplanaires en 3D, et soient $N_j, j = 0,1,2$ trois droites 3D noires non-coplanaires telles que chaque paire droite blanche B_i droite noire N_j soient coplanaires (elles se croisent au point Q_{ij}). Une droite est dite noire (respectivement blanche) si elle coupe trois droites blanches (noires). Alors toutes les droites noires sont coplanaires aux droites blanches. On peut remarquer que les droites noires et blanches génèrent une surface réglée quadratique, un hyperboloïde à une nappe ou un paraboloïde hyperbolique.

Nous reformulons le théorème comme suit : soit 16 points en 3D P_{ij}, $i = 0,1,2,3$ et $j = 0,1,2,3$ tels que pour tout i, P_{i0}, P_{i1}, P_{i2} soient colinéaires, P_{i0}, P_{i1}, P_{i3} soient colinéaires, et symétriquement. N'importe lequel de ces 16 alignements est une conséquence des 15 autres.

Trouver un témoin n'est pas trivial dans ce cas : il ne suffit pas de choisir 16 points au hasard.

Nous avons utilisé deux formulations différentes pour générer ces 16 contraintes d'alignements. Dans la première, l'alignement de 3 points est exprimé par l'annulation de deux déterminants. Dans la seconde, chaque alignement est exprimé par l'annulation de trois déterminants (ce qui est redondant).

Toutes les méthodes fournissent un témoin. L'avantage est à la méthode NMS. Les méthodes sont également plus efficaces avec la formulation comportant uniquement deux déterminants.

5.3. Droite tangente à quatre sphères

Le problème 3D sous-jacent a été soumis et résolu pour la première fois par Hoffmann et Yuan dans [HY00] : calculer les droites tangentes à quatre sphères données.

Générer un témoin est évident : on génère au hasard une droite et quatre points (les centres des sphères), on déduit ensuite les quatre rayons qui portent la distance de ces points à la droite. La droite est représentée par son vecteur directeur et le point de la droite le plus proche de l'origine.

5.4. L'octaèdre

Les polytopes 3D (polyèdres convexes) sont complètement spécifiés par la donnée des longueurs des arrêtes et la coplanarité des points appartenant à la même face. En d'autres termes, le système est rigide : c'est le théorème de Cauchy.

Il existe un algorithme spécifique qui calcule la réalisation (des coordonnées consistantes pour les sommets) d'un polytope 3D, à partir de son graphe (appelé squelette ou 1-squelette) ; il calcule un plongement de Tutte du graphe du polytope. Étonnamment, les coordonnées des points sont des nombres rationnels et même entiers aprés mise à l'échelle. Cet algorithme est une preuve calculatoire du théorème de Steinitz : tout polytope convexe possède une réalisation entière quelque soit son graphe. Cet algorithme pourrait être utilisé pour calculer un témoin pour un polytope décrit par son graphe.

Pour un octaèdre, toutes les faces sont triangulaires. Il n'y a donc pas de contraintes de coplanarité. Générer un témoin est évident : il suffit de prendre 6 sommets au hasard, puis de prendre les longueurs des arêtes. Le polytope généré peut

méthode	taux de succés	temps total
Newton	100%	0.24 s
Newton dans $\mathbb{C}$	100 %	0.3 s
BFGS	100 %	4.72 s
NMS	0 %	4.46 s

Table 7: *Configuration de Pascal*

méthode	taux de succés	temps total
Newton	100%	3.14 s
Newton dans $\mathbb{C}$	100 %	3.44 s
BFGS	100 %	18.72 s
NMS	100 %	1.02 s

Table 8: *configuration de Desargues en 3D*

méthode	taux de succès	temps total
Newton	100%	1.72 s
Newton dans $\mathbb{C}$	100 %	1.72 s
BFGS	100 %	26.56 s
NMS	100 %	1.32 s

Table 9: *Configuration de Beltrami (2 determinants)*

méthode	taux de succés	temps total
Newton	100%	7.22 s
Newton dans $\mathbb{C}$	100 %	7.76 s
BFGS	100 %	44.3 s
NMS	100 %	1.96 s

Table 10: *Configuration de Beltrami (3 determinants)*

méthode	taux de succés	temps total
Newton	1%	2.26 s
Newton dans $\mathbb{C}$	1 %	2.2 s
BFGS	1 %	0.38 s
NMS	0 %	8.82 s

Table 11: *Droite tangente à 4 sphères*

méthode	taux de succés	temps total
Newton	100%	0.68 s
Newton dans $\mathbb{C}$	100 %	0.7 s
BFGS	30 %	9.88 s
NMS	0 %	11.16 s

Table 12: *Octaèdre*

être non-convexe mais ce n'est pas important : cela reste un témoin.

5.5. Octaèdre avec les determinants de Cayley-Menger

Les dix distances entre cinq points en 3D ne sont pas indépendantes : le déterminant de Cayley-Menger associé doit s'annuler. Ainsi les determinants de Cayley-Menger [MF04] fournissent une solution simple au problème de l'octaèdre, dit de la plateforme de Stewart.

On écrit les conditions de Cayley-Menger pour les sommets équatoriaux et le sommet nord, puis pour les quatres sommets équatoriaux et le sommet sud. Ces deux équations font appels à deux distances inconnues qui sont les distances entre deux sommets opposées de l'équateur. Les autres distances sont connues par hypothèse. Ce système est plus simple que le système naïf (qui contient initialement 30 inconnues et 12 équations). Malheureusement, on ne sait pas étendre ces conditions aux autres polytopes.

Il est curieusement plus aisé d'obtenir un témoin en utilisant le système naïf bien qu'avec la formulation de Cayley-Menger le système ne comporte que 2 équations. La méthode de Newton dans $\mathbb{C}$ s'avère être la plus efficaces pour cette configuration.

5.6. L'hexaèdre

Le cube est un hexaèdre : le graphe d'un hexaèdre est celui d'un cube. L'hexaèdre possède 6 faces quadrilatères (coplanaires). On peut le décrire complètement par les conditions de coplanarité et la longueur de ces 12 arêtes.

À cause des conditions de coplanarité, il ne suffit pas de choisir 8 points au hasard pour obtenir un témoin. Cependant on peut en trouver un facilement : le cube de Platon. C' est un témoin évident pour tous les hexaèdres, mais peut être pas un témoin typique.

Nous avons généré le système associé ; pour un témoin, les longueurs des 12 arêtes sont considérés comme des inconnues. La coplanarité de quatre sommets est spécifiée par l'annulation d'un déterminant 4 par 4 des coordonnées homogènes des quatres points concernés.

Ici encore, l'algorithme de Newton s'avère être le plus rapide.

5.7. L'icosaèdre

Le graphe de l'icosaèdre générique est le même que celui du solide de Platon associé. Il est constitué de 20 faces triangulaires, 12 sommets et 30 arêtes. Il n'y a pas de contraintes de coplanarité. Générer un témoin est trivial : il suffit de choisir 12 points au hasard. Il peut y avoir des auto-intersections et le polyèdre peut être concave mais il reste un témoin.

5.8. Le dodécaèdre

Le graphe considéré est celui du dodécaèdre régulier. Le dodécaèdre est fait de 12 faces pentagonales, 20 sommets et 30 arêtes. C'est le dual de l'icosaèdre. À cause des contraintes de coplanarité sur les sommets des faces, on ne peut pas se contenter de choisir des sommets au hasard pour obtenir un témoin. Il est en revanche possible de choisir au hasard 12 plans, et d'en déduire les sommets qui sot tous de degré 3. Le polyèdre obtenu ne sera probablement pas convexe et présentera des auto-intersections mais ce sera un témoin.

On génère le système de contraintes en considérant les longueurs des arêtes comme des inconnues. Les coplanarités de cinq points $ABCDE$ d'une même face sont spécifiées par l'annulation de deux déterminant 4 par 4, $|ABCD| = |ABCE| = 0$. Ces conditions de coplanarité sont indépendantes.

6. Synthèse des résultats obtenus

On utilise des points initiaux aléatoires pour les méthodes qui en ont besoin et l'on effectue 100 essais pour chaque méthode. L'initialisation est identique pour chaque méthode pendant le même essai. Le temps indiqué est le temps cumulé sur ces 100 essais. Le système $F(U,X) = 0$ est résolu en considérant les paramètres U comme des inconnues. Toutes les méthodes testées convergent si le point de départ est une racine, ou est suffisamment près d'une racine. La proximité des points de départ par rapport à une racine aide le solveur à converger vers cette racine, mais cette convergence dépend de la méthode choisie. Les bassins d'attraction pour une racine dépendent du solveur ; par exemple ceux de la méthode de Newton sont fractals, ceux de l'homotopie ont des frontières lisses.

Dans la plupart des cas, l'algorithme de Newton fourni un témoin et s'avère la plus rapide. Pour certains cas (problème de desargues en 3D et pavage par des cercles) l'utilisation de l'algorithme de Newton dans les complexes donne de meilleurs résultats.

La méthode NMS ne donne pratiquement aucun résultat : cela s'explique par le fait qu'elle se trouve souvent bloquée sur un minimum local non nul. La même remarque s'applique à la méthode BFGS même si celle-ci obtient de meilleurs résultats, en particulier pour les formulations à base de déterminants.

Généralement, lorsque la formulation des alignements utilisent des déterminants, les méthodes sont plus effficaces pour trouver un témoin (voir tableau 6 et tableau 4).

Curieusement le problème de la droite tangente à quatre sphères s'avère difficile pour chacune des méthodes testées. Même la méthode de Newton ne fournit qu'un seul témoin sur 100 essais. Nous n'avons pas trouvé d'explication.

méthode	taux de succés	temps total
Newton	89%	5.9 s
Newton dans $\mathbb{C}$	100 %	4.06 s
BFGS	80 %	129.16 s
NMS	0 %	54 s

Table 13: *Octaèdre (Cayley-Menger)*

méthode	taux de succés	temps total
Newton	100%	1.96 s
Newton dans $\mathbb{C}$	100 %	2.16 s
BFGS	14 %	57.42 s
NMS	0 %	34.06 s

Table 14: *Hexaèdre*

méthode	taux de succés	temps total
Newton	100%	2.24 s
Newton dans $\mathbb{C}$	100 %	2.34 s
BFGS	100 %	41.5 s
NMS	0 %	56.46 s

Table 15: *Icosaèdre*

méthode	taux de succés	temps total
Newton	100%	5.6 s
Newton dans $\mathbb{C}$	100 %	5.8 s
BFGS	100 %	182.38 s
NMS	100 %	3.02 s

Table 16: *Dodécaèdre*

Système	inconnues	équations	meilleure méthode	taux de succés	temps total
2 rigides	21	9	Newton	100%	0.24 s
Pavage de cercles ($k = 5$)	60	44	Newton (dans $\mathbb{C}$)	82%	8.9 s
Desargues 2D	30	20	Newton	100%	0.54 s
Pappus	45	36	Newton (dans $\mathbb{C}$)	100%	0.2 s
Pascal	22	11	Newton	100%	0.24 s
Desargues 3D	50	40	NMS	100%	1.02 s
Beltrami	48	48	NMS	100%	1.32 s
Droite tangente à 4 sphères	26	10	BFGS	1%	0.38 s
Octaèdre	30	12	Newton	100%	0.68 s
Octaèdre (avec Cayley Menger)	14	2	Newton (dans $\mathbb{C}$)	100%	4.06 s
Hexaèdre	36	18	Newton	100%	1.96 s
Icosaèdre	66	30	Newton	100%	2.24 s
Dodécaèdre	60	24	NMS	100%	3.02 s

7. MNP et témoin

La méthode du témoin étend la méthode numérique probabiliste (MNP).

La MNP est utilisée pour tester la rigidité d'un graphe non orienté, pour une dimension donnée d.

Le problème de la rigidité d'un graphe est le suivant. Un graphe non orienté est donné, ainsi qu'une dimension d, entière. Chaque sommet du graphe est représenté par un point aléatoire en dimension d. Les arêtes du graphe ont leurs longueurs fixées par les distances entre les points correspondants. Comme les sommets sont aléatoires, la probabilité de la colinéarité (pour $d > 1$) de 3 points ou davantage est nulle, de même que la coplanarité de 4 points ou davantage en dimension $d > 2$.

Est-il possible de déformer la configuration sans modifier les longueurs des arêtes du graphe ? Si oui, le graphe est flexible ; sinon, le graphe est rigide.

D'après la théorie de la rigidité, la réponse ne dépend que du graphe considéré et de la dimension d ; plus précisément, l'ensemble des contre-exemples (une configuration infinitésimalement flexible, comme un triangle plat, alors que le graphe est rigide) est de mesure nulle, et un tirage aléatoire des coordonnées des sommets du graphe a donc une probabilité nulle de trouver un tel contre-exemple.

Exemple : en 2D, deux triangles partageant une arête commune forment un graphe rigide ; ce même graphe n'est plus rigide en 3D, car les deux triangles peuvent pivoter autour de leur arête commune. Le graphe du tétraèdre est rigide en 3D.

Pour décider la rigidité d'un graphe donné de S sommets en dimension d, la MNP procède ainsi :

Elle associe au graphe et à la dimension d un système d'équations ; chaque sommet du graphe est représenté par un point en dimension d ; les inconnues du système sont les $d \times S$ coordonnées des sommets ; chaque équation fixe la distance entre les deux sommets d'une arête du graphe ; les paramètres du système sont les longueurs des arêtes du graphe.

Comme le système des contraintes ne comporte aucune contrainte d'incidence, mais seulement des contraintes de distances entre les sommets, trouver un témoin est facile : il suffit pour chaque sommet de tirer un point au hasard en dimension d ; les valeurs des paramètres (les longueurs des arêtes) s'en déduisent trivialement.

La MNP étudie ensuite le jacobien du système, au témoin, exactement comme le fait la méthode du témoin.

La MNP est plus puissante que les méthodes fondées sur les graphes ; ainsi, elle détecte la flexibilité de la double banane. Cette flexibilité échappe aux méthodes à base de graphes, qui généralisent, commodément (mais indûment), la condition de Laman, décident que la double banane est rigide.

Mentionnons que la MNP décide, en temps polynomial, de la rigidité des graphes en toute dimension, alors que la caractérisation combinatoire (*i.e.* en termes de graphe) de la rigidité n'est connue qu'en 2D. Cette condition combinatoire en 2D a été donnée par Laman, un ingénieur en structures.

Il est tentant de généraliser la MNP à d'autres systèmes de contraintes géométriques. Pour étudier un système $F(X) = 0$ (les paramètres U disparaissent, remplacés par leurs valeurs), la MNP étendue choisit un point X_r au hasard (qui a donc une probabilité nulle d'être solution de $F(X) = 0$) et étudie le jacobien en ce point. Si le jacobien n'est pas de plein rang en un point aléatoire, alors le jacobien est singulier partout avec probabilité 1, et les équations sont donc dépendantes avec probabilité 1 ; soit elles sont contradictoires, soit elles sont redondantes. La MNP ne peut décider entre ces deux possibilités.

Comme la MNP étendue étudie le jacobien en un point qui n'est pas solution du système considéré, les dépendances plus subtiles, qui ne se produisent que pour les solutions, lui échappent : on dit que la MNP fournit un témoin faible. La MNP étendue ne détecte pas que certaines contraintes géométriques sont des conséquences des autres contraintes. Tout théorème géométrique (Pappus, Pascal, Desargues, etc) peut être utilisé pour produire un système redondant, ou contradictoire, de contraintes : les conditions du théorème sont imposées comme des contraintes, de même que la conclusion du théorème (alors le système obtenu est redondant), ou la négation de la conclusion du théorème (alors le système obtenu est contradictoire).

Malgré cette limitation de la MNP, il n'en reste pas moins que les dépendances qu'elle détecte sont réelles. Elle est simple, rapide, et on aurait grand tort de ne pas l'utiliser : quand il n'y a pas de témoin disponible, la MNP étendue permet de s'assurer de l'indépendance des équations sur un point aléatoire, *i.e.* presque partout.

La MNP est bien sûr un précurseur de la méthode du témoin. Cette dernière détecte toutes les redondances, contrairement à la MNP, mais a besoin d'un témoin.

On peut envisager d'utiliser la MNP pour décomposer les systèmes de contraintes, quand un témoin n'est pas disponible. Cela n'a pas été fait pour l'instant. A priori, cela paraît facile, puisque les deux méthodes ne se différencient que par les exemples qu'elles considèrent : un point aléatoire, non solution, pour la MNP et la MNP étendue, et un témoin (un point solution) pour la méthode du témoin.

8. Détection de dépendance parmi les paramètres

Les paramètres devraient être indépendants, au moins dans un système de contraintes correct. De telles dépendances n'empêchent pas le calcul d'un témoin. La méthode du témoin,*i.e.* en étudiant le jacobien du système au témoin, permet aussi de détecter la dépendance entre paramètres. Nous illustrons ce fait avec les deux exemples suivants.

Premier exemple, le système $x - u = x - v = 0$ possède 1

inconnue x et 2 paramètres u, v. Un témoin pour ce système est facile à calculer. Les paramètres u et v sont clairement dépendants : $u = v$. Remarquons que si u et v deviennent des inconnues, le système reste correct. Si l'on renomme v en y, le système devient $x - u = x - y = 0$ ce qui est pertinent.

Deuxième exemple, le système $x+y=u, x+z=v, y-z=w$ possède 3 inconnues x, y, z. Un témoin pour ce système peut être calculé aisément. Les paramètres u, v, w sont dépendants : $w = u - v$. Ici encore, si l'on considère les paramètres comme des inconnues, les trois équations sont indépendantes puisque le jacobien contient la sous-matrice 3×3 identité lorsque l'on dérive relativement à u, v et w :

$$J = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

La méthode du témoin détecte les dépendances en considérant le jacobien pour les variables x, y, z (et en éliminant les variables u, v, w) :

$$J = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

La sous-matrice 3×3 est de rang 2 car la dernière ligne est la différence des deux autres.

Le rang peut être différent selon que les colonnes associées aux paramètres sont prises en compte ou non, ce qui permet la détection des dépendances entre paramètres. Rappelons que chaque ligne contient la différentielle d'une équation, et chaque colonne est associée à une variable, *i.e.* une inconnue ou un paramètre.

9. Conclusion

Cet article compare différentes méthodes pour calculer un témoin. Pour chaque méthode, les points de départ ont été choisis aléatoirement. Nous n'avons pas essayé de décomposer les systèmes. Parmi ces méthodes numériques, l'algorithme de Newton-Raphson a donné les meilleurs résultats. Dans la plupart des cas, il converge vers un témoin dans un nombre raisonnable d'itérations.

La conclusion propose la stratégie suivante : en premier on essaye la méthode de Newton en partant de 50 points aléatoires ; si tous les essais échouent, on utilise un solveur complet comme par exemple un solveur par intervalle. Ce dernier est un ordre de grandeur plus lent que la méthode de Newton mais il est sûr et complet.

On peut compléter ce travail de différentes façons. On peut envisager de partir d'une solution dégénérée (tous les sommets sont égaux) et faire une marche aléatoire selon l'espace tangent de la variété solution. On peut également tester une version amortie de la méthode de Newton. Enfin on peut essayer d'utiliser des méthodes de décomposition avant de calculer un témoin, par exemple en utilisant un témoin faible ou une méthode de graphe.

Références

[BR98] BRÜDERLIN B., ROLLER D. (Eds.) : *Geometric Constraint Solving and Applications*. Springer, 1998.

[CS03] COLLINS C., STEPHENSON K. : A circle packing algorithm. *Computational Geometry : Theory and Applications. Vol. 25*, Num. 3 (2003), 233–256.

[DH00] DURAND C. B., HOFFMANN C. M. : A systematic framework for solving geometric constraints analytically. *Journal of Symbolic Computation. Vol. 30*, Num. 5 (2000), 493–519.

[Dur98] DURAND C. B. : *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Purdue University, West Lafayette, IN, U.S.A., 1998.

[FMF09] FÜNFZIG C., MICHELUCCI D., FOUFOU S. : Nonlinear systems solver in floating point arithmetic using LP reduction. In *SPM '09 : Proceedings of the SIAM/ACM joint conference on Geometric and Physical Modeling* (San Francisco, CA, U.S.A., 2009), pp. 123–134.

[GM06] GAO X.-S., MICHELUCCI D. : Special issue on geometric constraints, guest editors' foreword. *International Journal of Computational Geometry and Applications. Vol. 16*, Num. 5–6 (2006), 377–378.

[HJA05] HOFFMANN C. M., JOAN-ARINYO R. : A brief on constraint solving. *Computer-Aided Design and Applications. Vol. 2*, Num. 5 (2005), 655–663.

[HY00] HOFFMANN C. M., YUAN B. : On spatial constraint solving approaches. In *ADG '00 : Proceedings of the 3rd International Workshop on Automated Deduction in Geometry* (Zürich, Switzerland, 2000), vol. 2061 de *Lecture Notes in Artificial Intelligence*, Springer, pp. 1–15.

[JTNM06] JERMANN C., TROMBETTONI G., NEVEU B., MATHIS P. : Decomposition of geometric constraint systems : a survey. *International Journal of Computational Geometry and Applications. Vol. 16*, Num. 5,6 (2006), 379–414.

[LM95] LAMURE H., MICHELUCCI D. : Solving geometric constraints by homotopy. In *SMA '95 : Proceedings of the 3rd ACM symposium on Solid Modeling and Applications* (Salt Lake City, Utah, USA, 1995), ACM Press, pp. 263–269.

[LM96] LAMURE H., MICHELUCCI D. : Solving geometric constraints by homotopy. *IEEE Transactions on Visualization and Computer Graphics. Vol. 2*, Num. 1 (1996), 28–34.

[MF04] MICHELUCCI D., FOUFOU S. : Using Cayley-Menger determinants for geometric constraint solving. In

SMA '04 : Proceedings of the 9th ACM symposium on Solid Modeling and Applications (Genoa, Italy, 2004), Eurographics Association, pp. 285–290.

[MF06] MICHELUCCI D., FOUFOU S. : Geometric constraint solving : The witness configuration method. *Computer-Aided Design. Vol. 38*, Num. 4 (2006), 284–299.

[MF07] MICHELUCCI D., FOUFOU S. : Detecting all dependences in systems of geometric constraints using the witness method. In *ADG '06 : Proceedings of the 6th international workshop on Automated Deduction in Geometry* (Pontavedra, Spain, 2007), vol. 4869 de *Lecture Notes in Artificial Intelligence*, Springer, pp. 98–112.

[MF09] MICHELUCCI D., FOUFOU S. : Interrogating witnesses for geometric constraint solving. In *SPM '09 : Proceedings of the SIAM/ACM joint conference on Geometric and Physical Modeling* (San Francisco, CA, U.S.A., 2009), ACM, pp. 343–348.

[MST*10] MICHELUCCI D., SCHRECK P., THIERRY S. E. B., FÜNFZIG C., GÉNEVAUX J.-D. : Using the witness method to detect rigid subsystems of geometric constraints in CAD. In *SPM '10 : Proceedings of the ACM Conference on Solid and Physical Modeling* (Haïfa, Israël, September 2010), ACM, pp. 91–100.

[MT10] MATHIS P., THIERRY S. E. B. : A formalization of geometric constraint systems and their decomposition. *Formal Aspects of Computing. Vol. 22*, Num. 2 (2010), 129–151.

[NM65] NELDER J. A., MEAD R. : A simplex method for function minimization. *The Computer Journal. Vol. 7*, Num. 4 (1965), 308–313.

[PFTV92] PRESS W. H., FLANNERY B. P., TEUKOLSKY S. A., VETTERLING W. T. : *Numerical Recipes in C, 2nd Edition*. Cambridge University Press, 1992.

[Ste03] STEPHENSON K. : Circle packing : A mathematical tale. *Notices of the ACM. Vol. 50*, Num. 11 (2003), 1376–1388.

[SW05] SOMMESE A. J., WAMPLER C. W. I. : *The numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.

Une approche par décomposition et reparamétrisation de systèmes de contraintes géométriques.

Rémi Imbach [1], Pascal Mathis [1] et Pascal Schreck [1]

[1] Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, UMR 7005

Résumé

La décomposition des systèmes de contraintes est un avatar de la stratégie diviser pour régner *qui est indispensable dans le cadre de la résolution de systèmes de contraintes géométriques en CAO. Diverses méthodes tirant partie de l'invariance par déplacement ont été décrites dans la littérature, mais malheureusement, elles n'arrivent pas à décomposer des systèmes relativement standards, surtout en 3D. Une idée assez récente consiste à conjuguer résolution formelle et résolution numérique d'une part en modifiant le système de contraintes original pour le rendre soluble par une méthode constructive, et, d'autre part, en appliquant une méthode numérique pour rattraper les contraintes non prises en compte dans le système modifié. Dans la lignée de ces travaux, nous présentons deux nouvelles avancées. La première concerne la manière de modifier le système en tenant compte d'une méthode de décomposition et dont l'objectif est de minimiser le nombre de contraintes modifiées par composante irréductible. La seconde consiste à concevoir une méthode numérique qui tire parti du contexte géométrique pour rattraper des contraintes de manière robuste et pour produire toutes les solutions possibles.*

Mots-clés : Systèmes de contraintes géométriques, analyse des degrés de liberté, systèmes de contraintes paramétrés, résolution numérique.

1. Introduction

La résolution de contraintes géométriques est une problématique dérivée des CSP (Constraint Satisfaction Problems) qui peut s'appliquer à divers domaines comme l'enseignement assisté par ordinateur, la robotique, la chimie moléculaire ou encore la CAO. C'est ce dernier domaine qui nous intéresse dans cet article.

L'objet du dessin technique est de concevoir des objets ou des scènes géométriques dans un sens assez large qui comprend, entre autres, les mécanismes, les pièces mécaniques et les ensembles architecturaux. Dans tous les cas, il s'agit de dessiner une esquisse représentant la forme de l'objet qui sert de support à la cotation : en termes plus modernes, l'esquisse définit la topologie de l'objet dessiné tandis que les cotes qui lui sont appliquées définissent sa géométrie. Les outils spécialisés dans le traitement des contraintes géométriques extraient de cette esquisse cotée plusieurs informations dont les contraintes d'incidence, provenant de la topologie, et des contraintes métriques provenant de la cotation (voir exemple Figure 1). Le tout forme un système de contraintes géométriques, ou GCS (Geometric Constraint System) en abrégé. En ce sens, la résolution de contraintes géométriques a sa place en modélisation géométrique entre la modélisation déclarative, une esquisse cotée est en effet une spécification déclarative d'un objet, et des approches plus impératives comme la modélisation à base topologique ou la CSG. La manipulation de tels objets définis par des contraintes passe souvent par un *solveur* qui traduit la spécification déclarative en une représentation informatique plus facilement exploitable. Cette représentation peut prendre la forme

- soit d'un ensemble de valeurs numériques traduisant la géométrie d'un objet particulier, on parle alors de solveur numérique [LGL81, LM95, Mic03]. De tels solveurs utilisent une méthode itérative comme celle de Newton-Raphson ou la méthode par continuation. Elles sont générales, mais il est difficile d'obtenir toutes les solutions.
- soit d'une procédure pour construire de tels objets, on parle alors de solveur constructif [Brü93, DMS97, JAS97]. Les solveurs constructifs sont beaucoup moins

généraux, mais en revanche, ils permettent dans la plupart des cas de parcourir l'espace des solutions, ce qui est intéressant lorsqu'on veut obtenir "la solution proche de l'esquisse".

En CAO, des esquisses cotées relativement simples peuvent comporter des centaines de contraintes et, même si il y a eu beaucoup de progrès dans le domaine des solveurs, il est crucial de décomposer un système de contraintes en sous-systèmes plus petits et plus faciles à résoudre. Diverses approches ont été étudiées, parmi les plus connues on trouve [LM96, DMS97, HLS97, TSM*11, Zha11]. Malgré des degrés de sophistication divers, on rencontre des systèmes de contraintes qu'on ne peut pas décomposer avec ces méthodes. C'est typiquement le cas des systèmes définissant des polyèdres en 3D, mais aussi de systèmes de contraintes "ordinaires" en 2D.

Des travaux assez récents ont proposé de contourner le problème en modifiant le système de contraintes à résoudre en enlevant des contraintes et en les remplaçant par d'autres pour maintenir la rigidité : le système transformé est résolu de manière constructive, et on rattrape les contraintes supprimées du système de départ en utilisant une méthode numérique. On parle parfois de système pseudo-décomposable et de pseudo-décomposition. L'approche décrite dans [GHY02] utilise une technique employant la *force brute* via des essais systématiques de remplacement de contraintes et échantillonnage d'une courbe. Elle n'est utilisable que pour des petits systèmes 3D où seule une contrainte est remplacée. Plus récemment, [FS08] décrit une approche plus générale pour résoudre une plus grande famille de cas en 2D ou en 3D.

Nous proposons ici des travaux en cours de développement qui portent à la fois sur la manière de transformer le système initial S en un système S' soluble par une méthode constructive, et sur une approche numérique pour résoudre le système résultant de la solution formelle de S' et des contraintes retirées de S. Pour le premier point, notre approche se distingue des approches précédentes qui tentaient de minimiser le nombre de contraintes à changer sans tenir compte d'aucune méthode de décomposition [†]. L'idée consiste ici à modifier le système de contraintes en même temps qu'on essaye de le décomposer : nous étudions un moyen de rendre minimal le nombre de contraintes à changer *par composante rigide du système modifié*. Pour le moment, la méthode de décomposition que nous utilisons est très simple, il s'agit d'une propagation des degrés de liberté pour former des clusters à la manière d'Hoffmann *et al.* [HLS97]. En ce qui concerne les méthodes numériques, nous étudions la manière d'adapter des méthodes générales de résolution d'équations et de suivi de courbes en utilisant les particularités du cadre géométrique. L'objectif est d'utiliser des critères géométriques lus sur la figure pour rendre plus fiables les méthodes de suivi de chemins et notamment prévoir les changements de branche. Nous étudions d'autre part des méthodes numériques pour obtenir toutes les solutions du système.

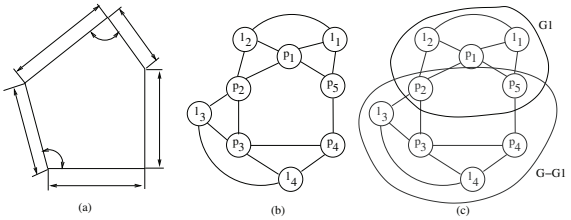


Figure 1: *Un GCS donné sous forme d'esquisse cotée et le graphe correspondant.*

La suite de cet article est organisée de la manière suivante. La section 2 introduit les notions de bases concernant les systèmes de contraintes et leur décomposition. La section 3 présente la méthode de reparamétrisation associée à la méthode des lieux. La section 4 présente en détail notre algorithme de décomposition. La section 5 explique comment nous réalisons la résolution numérique. La section 6 conclut.

2. Définitions préliminaires

2.1. GCS et graphes de contraintes

Un GCS est un ensemble de contraintes portant sur des primitives géométriques (points, droites, cercles, etc). Chaque primitive possède un degré de liberté (*DOF* pour *Degree Of Freedom*) correspondant au nombre minimum de paramètres permettant de la définir. À chaque contrainte est associé un degré de restriction (*DOR* pour *Degree Of Restriction*) qui représente le nombre de degrés de liberté qu'elle ôte aux primitives. Par exemple, dans l'espace, un point et un plan ont chacun 3 DOF alors qu'une droite en possède 4. Une contrainte de distance supprime 1 degré de liberté, une contrainte de type "être le milieu de de deux points" supprime les 3 degrés de liberté d'un point. En dimension d, avec P l'ensemble des primitives et C l'ensemble des contraintes, les GCS bien-contraints [‡] vérifient la relation suivante :

$$\sum_{p\in P} DOF(p) - \sum_{c\in C} DOR(c) = \binom{d+1}{2}$$

En CAO, les primitives sont souvent des points et droites en 2D, des points et plans en 3D ainsi que des cercles ou sphères de rayon fixé. Par ailleurs les types de primitives utilisés restreignent un seul degré de liberté ce qui explique que

†. Évidemment, un telle méthode pouvait être mise en œuvre, après modification du système, par le solveur constructif employé

‡. on dit aussi iso-rigides : ils contiennent le nombre minimum de contraintes pour définir une figure rigide

la relation précédente se trouve généralement sous la forme : $2n-3=c$ en 2D et $3n-6=c$ en 3D pour n primitives et c contraintes. Les constantes 3 et 6 correspondent aux degrés de liberté en translation et rotation des objets.

Un GCS peut être représenté par un graphe de contraintes $G=\langle V,E\rangle$ dans lequel les sommets correspondent aux primitives et les arêtes aux contraintes. Afin de simplifier le discours mais sans perte de généralité, nous considèrerons des contraintes de degré 2 et des contraintes de degré 1. Cela nous permet notamment de conserver le formalisme des graphes et d'éviter celui des hypergraphes, rendu nécessaire pour des contraintes impliquant plus de 2 primitives. L'extension aux hypergraphes est toutefois immédiate.

Si v est un sommet de G, nous noterons $DOF(v)$ le degré de liberté de la primitive associée alors que $DEG(v)$ désignera le degré du sommet dans le graphe.

Remarquons que dans la terminologie des graphes, le problème illustré par la figure 4 est habituellement noté $K_{3,3}$. Il s'agit en effet du graphe biparti complet à deux ensembles de 3 sommets.

La figure 1(a) représente un GCS en 2D avec 5 points, 5 distances et 2 contraintes d'angle. À droite, se trouve le graphe correspondant. Les sommets étiquetés p_i correspondent aux points et ceux étiquetés l_i aux droites. Les arêtes liant deux points schématisent des distances imposées, les arêtes point-droite des contraintes d'incidence et les arêtes entre droites des contraintes d'angle.

2.2. Décomposition et sous-graphes

Nous commençons par rappeler les notions élémentaires de la théorie des graphes. Sauf mention explicite du contraire, nous considérons des graphes non orientés. Soit $G=\langle V,E\rangle$ un graphe, on note $E=E[G]$ l'ensemble de ses arêtes et $V=V[G]$ l'ensemble des ses sommets. Si $V_1\subset V$, on note $G[V_1]$ le graphe induit ne contenant que les arêtes de G ayant leurs deux extrémités dans V_1. On définit de manière similaire le sous-graphe induit par un ensemble d'arêtes $E_1\subset E$ dont chaque sommet est incident à au moins une arête de E_1 et noté $G[E_1]$.

Soit un graphe G et un sous-graphe G_1 avec $E_1=E[G_1]$, on définit la différence $G-G_1$ comme étant égale à $G[V_2]$ avec $V_2=V[G[E-E_1]]$. Pour deux sous-graphes G_1 et G_2 de G, on définit l'intersection $G_{1\cap 2}=G[V_1\cap V_2]$. On sait ([MT10]) que si G_1 représente un GCS bien-contraint, il en est de même pour $G_2=G-G_1$ si et seulement si $G_{1\cap 2}$ est bien contraint et non réduit à un point en 2D et à deux points en 3D. La figure 1(c) représente un graphe G, un sous-graphe G_1 et $G_2=G-G_1$. $G_{1\cap 2}$ contient deux sommets $p2$ et $p5$ non reliés. Avec ces notations, on appelle sommet intérieur d'un sous-graphe G_1 un sommet de $G_1-G_{1\cap 2}$.

Nous supposons maintenant que G correspond à un GCS bien-contraint, que G_1 est un sous-graphe de G et que $G_2=G-G_1$. Alors, si G_1 est bien contraint et que $G_{1\cap 2}$ ne l'est pas, on peut le compléter par des informations tirées de G_1. Par exemple, dans la figure 1(b), dès que le GCS correspondant à G_1 est résolu, on peut calculer la distance $d=p2$-$p5$ et compléter $G_{1\cap 2}$ par une contrainte de distance liant ces deux points dans $G_{1\cap 2}$. Ce système complété est appelé le bord de G_1 dans G et noté $B_{G_1/G}$ [MT10].

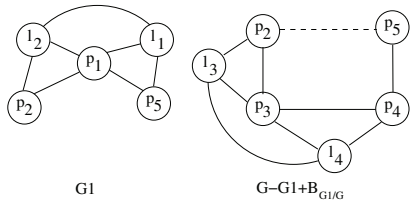


Figure 2: *Décomposition*

En termes de graphes, la decomposition d'un GCS correspondant au graphe G est une suite finie $G_1,G_2,\ldots,G_n$ où :
- G_1 correspond à un GCS bien-contraint,
- $G_2,\ldots,G_n$ est une décomposition de $G-G_1+B_{G_1/G}$

La figure 2 montre une décomposition du graphe de l'exemple précédent. Il faut noter qu'une telle décomposition n'est en général pas unique. On dit qu'elle est maximale lorsque la longueur de la suite de sous-graphes est maximale.

3. Méthode des lieux et reparamétrisation

La méthode par intersection des lieux géométriques, en abrégé méthode des lieux ou LIM (*Locus Intersection Method*), traduit chaque contrainte $c(p_i,p_j)$, où p_i (resp. p_j) est une primitive inconnue (resp. connue) en le lieu géométrique de p_i induit par $c(p_i,p_j)$. Les primitives inconnues sont construites en faisant l'intersection de ces lieux. Si un GCS peut être résolu sans changement par une telle suite d'intersections de lieux géométriques, une propagation des degrés de liberté dans le graphe correspondant fournit cette suite de constructions. LIM a un coût quadratique, mais une faible puissance de résolution. Nous détaillons cette méthode à la suite et nous montrons comment la reparamétrisation permet d'étendre largement la classe des GCS solubles.

3.1. LIM et plans de construction

La propagation des degrés de liberté peut être réalisée selon deux stratégies. La première, appelée rétro-propagation procède par déconstruction du graphe : un des sommets v satisfaisant $DEG(v)=DOF(v)$ est retiré du graphe avec les arêtes ayant ce sommet comme extrémité. Si après avoir répété cette étape autant de fois qu'il est possible, le graphe est réduit à un *repère*, ensemble de primitives déterminant un repère affine de l'espace, la résolution est un succès. Par exemple, dans le plan, si le graphe est réduit à un couple de sommets adjacents, la propagation s'achève.

La seconde stratégie, ou propagation avant, consiste à choisir un ensemble de primitives formant un repère, et à marquer les sommets correspondants comme construits. Puis, si un sommet v est adjacent à $DOF(v)$ sommets déjà construits, il est marqué lui aussi. Cette opération est appliquée jusqu'à ce qu'aucun nouveau sommet ne puisse être marqué. Lorsque tous les sommets sont marqués, le GCS est résolu.

Considérons à présent la propagation implantée par l'algorithme 1 où $DEGM(v)$ représente le nombre de sommets marqués adjacents à un sommet v.

Algorithme 1 Propagation avant des degrés de liberté

Entrées: $G = \langle V, E \rangle$: graphe de contraintes non-vide

1: Choisir un repère et marquer ses sommets
2: Choisir un sommet non-marqué v tel que $DEGM(v) = DOF(v)$, marquer v
3: Retourner à 2 tant qu'il est possible de choisir un sommet

Son application à un GCS fournit un *plan de construction*, c'est à dire une suite d'instructions permettant de construire les primitives du problème comme intersections de lieux géométriques. Pour le problème illustré par la figure 3 avec comme repère $(p_0 p_1)$, il produit le plan de construction suivant :

p_0 = fix()
  //origine du repère
l_0 = hline(p_0)
  // droite horizontale passant par p_0
p_1 = intersection_cl(p_0, k_0, l_0)
  // intersection du cercle (centre : p_0, rayon : k_0) avec l_0
p_2 = intersection_cc(p_0, k_6, p_1, k_1)
  // intersection des cercles (p_0, k_6) et (p_1, k_1)
p_5 = intersection_cc(p_0, k_5, p_2, k_8)
...

Ces deux stratégies permettent de résoudre les mêmes problèmes et traduisent une constructibilité symbolique par LIM. Les solutions numériques sont produites dans un second temps en calculant ces intersections. Remarquons qu'un plan de construction peut être vu comme une fonction des paramètres du GCS, les k_i dans l'exemple précédent.

L'évaluation numérique du plan de construction conduit à un arbre dont les sommets de profondeur n correspondent à l'évaluation de la n-ème instruction du plan de construction. À chaque fois qu'une telle instruction donne k solutions, k fils sont ajoutés au sommet correspondant. Finalement, l'arbre a autant de feuilles qu'il y a de solutions au GCS, chaque solution étant décrite par une branche de l'arbre.

Remarquons que LIM sous-tend un arbre avec potentiellement plus de branches que n'en produit l'évaluation. En effet, l'évaluation du plan de construction sur certaines branches peut échouer, par exemple quand un point dans le plan est définit comme l'intersection de deux cercles qui ne se coupent pas.

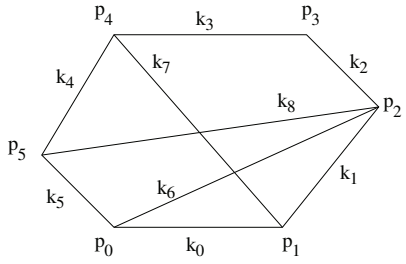


Figure 3: *Un GCS soluble par LIM. Les k_i sont les paramètres des distances*

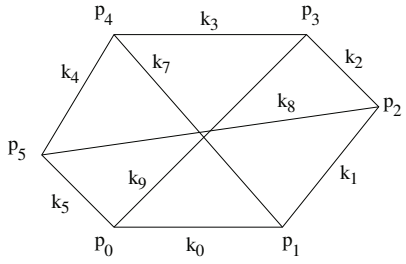


Figure 4: *Un problème non soluble par LIM*

Soit maintenant le GCS nommé $K_{3,3}$ et illustré par la figure 4 consistant à construire six points en connaissant neuf contraintes de distances entre ces points. Quel que soit le repère choisi, l'algorithme 1 échoue car chaque sommet a un degré dans le graphe strictement supérieur à son degré de liberté : les points p_i ont deux degrés de liberté mais sont contraints par trois distances.

L'approche par reparamétrisation consiste à remplacer un tel GCS par un autre, similaire, mais soluble par LIM.

3.2. Reparamétrisation

Reprenons l'exemple $K_{3,3}$ et supposons que le GCS de la figure 3 admette une solution s^* qui satisfasse la contrainte $distance(p_0, p_3) = k_9$. Alors s^* est aussi une solution du GCS $K_{3,3}$, car toutes ses contraintes sont satisfaites par s^*. Nous avons ainsi transformé le GCS G en un GCS G' en les mêmes inconnues en remplaçant certaines contraintes ($distance(p_0, p_3) = k_9$ dans l'exemple ci-dessus) par d'autres ($distance(p_0, p_2) = k_6$), avec G' soluble par LIM. Un plan de construction peut alors être formé pour G'.

Les paramètres des contraintes ajoutées (angles ou distances) sont appelées *paramètres guides* (k_6 dans l'exemple

précédent). La suite de la méthode consiste à rechercher des valeurs pour les paramètres guides de sorte que les solutions de G' soient aussi des solutions de G : c'est l'étape de résolution numérique.

[GHY02] décrit une procédure par rétro-propagation calculant à partir d'un GCS bien contraint G un nouveau GCS G' soluble par LIM. Cette méthode est réécrite en propagation avant dans l'optique d'une décomposition expliquée plus bas. La procédure s'arrête dès que le GCS induit par le sous-graphe des sommets marqués est bien contraint et contient au moins un sommet intérieur.

Notre méthode de reparamétrisation agit, elle, par propagation avant. Elle est formalisée par l'algorithme 2. À chaque itération, plusieurs sommets peuvent êtres choisis ; l'algorithme n'est donc pas déterministe et peut amener à plusieurs reparamétrisations différentes. Pour le GCS $K_{3,3}$, une de ces reparamétrisations possibles est le GCS de la figure 3.

Algorithme 2 Reparamétrisation

Entrées: $G = \langle V, E \rangle$: graphe de contraintes non-vide

1: Choisir un repère et marquer ses sommets
2: Choisir un sommet non encore marqué v tel que $DEGM(v) > 0$ et $DEGM(v) - DOF(v)$ est minimum
3: **Si** DEGM(v) < DOF(v)
   ajouter DOF(v)-DEGM(v) contraintes
   Si DEGM(v) > DOF(v)
   enlever DEGM(v)-DOF(v) contraintes
4: Marquer v
5: Soit Vm l'ensemble des sommets marqués
   Arrêter si $G[Vm]$ est bien contraint et contient un sommet intérieur
   sinon retourner à 2

3.3. Résolution numérique

On considère maintenant le plan de construction Cp du GCS reparamétré G' comme une fonction des paramètres guides $k^+ = (k_0^+, ..., k_{d-1}^+)$, et on le note $Cp(k^+)$ par omission des autres paramètres ([IMS11]).

Pour une figure produite par $Cp(k^+)$, on teste la satisfaction des contraintes supprimées en définissant la fonction $F : \mathbb{R}^d \to \mathbb{R}^d$ par $F = (F_0, ..., F_{d-1})$ où $F_i(k^+) = f_i^-(Cp(k^+)) - k_i^-$. $f_i^-(Cp(k^+))$ correspond à l'évaluation de la contrainte supprimée c_i^- de paramètre k_i^-.

En d'autres termes, $F(k^+)$ évalue, sur une figure produite par le plan de construction $Cp(k^+)$ évalué pour les paramètres $k^+ = (k_0^+, ..., k_{d-1}^+)$ les d contraintes supprimées et s'annule si elles sont satisfaites.

Les valeurs numériques des paramètres guides $(h_0, ..., h_{d-1})$ qui sont solutions de $F(h) = 0$ peuvent être obtenues en échantillonnant une boîte de $\mathbb{R}^d$ ([GHY02])

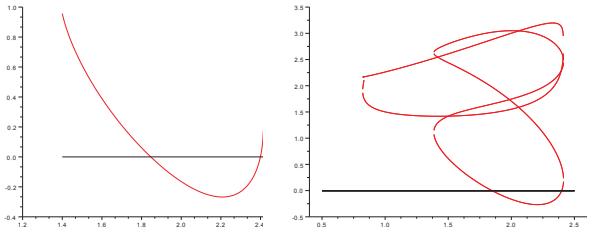


Figure 5: *À gauche : échantillonnage d'une branche du plan de construction. À droite : échantillonnage de toutes les branches du même plan de construction*

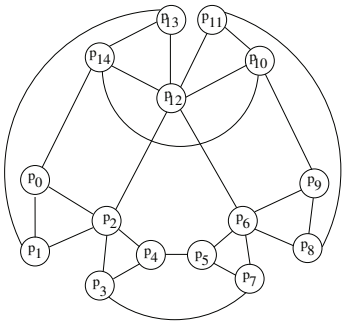


Figure 6: 15 *primitives et* 27 *contraintes. Deux contraintes sont remplacées pour résoudre le problème par LIM :*(p_1p_{13}) *et* (p_8p_{11})

et en calculant $F(x_0, ..., x_{d-1})$ sur chacune des branches du plan de construction de G' pour chaque échantillon $(x_0, ..., x_{d-1})$. La figure 5 illustre un tel échantillonnage dans le cas de la reparamétrisation de $K_{3,3}$.

Considérons à présent le GCS du plan dont le graphe de contrainte est donné par la figure 6, impliquant 15 primitives et 27 contraintes. Sa reparamétrisation amène à remplacer deux contraintes par deux autres dans le meilleur des cas et le problème est résolu en échantillonnant une fonction $F : \mathbb{R}^2 \to \mathbb{R}^2$, ce qui devient coûteux. En fait, ce coût explose quand le nombre de contraintes remplacées augmente, et il est encore aggravé par la croissance exponentielle du nombre de branches du plan de construction en fonction du nombre d'instructions.

L'idée proposée ici pour gérer cette explosion du coût consiste à choisir les contraintes à remplacer de telle sorte que le GCS G' soit décomposable en sous-systèmes bien contraints selon une méthode de décomposition donnée. Par exemple, le problème de la figure 6 peut-être transformé en le graphe de la figure 7 : trois contraintes ont été remplacées, mais le GCS induit par ce graphe est décomposable en trois sous-systèmes, solubles indépendamment, contenant chacun

un seul paramètre guide, et dont les plans de construction contiennent beaucoup moins de branches que le problème global.

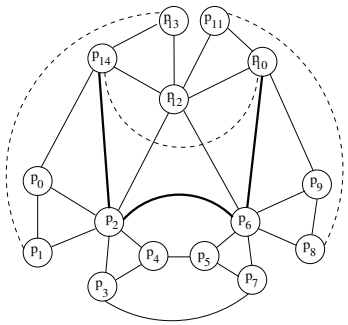


Figure 7: *Une reparamétrisation permettant la décomposition : les arêtes en pointillés sont les contraintes supprimées, celles en gras les contraintes ajoutées au GCS initial.*

Dans [FS08], la méthode de Newton-Raphson est appliquée à la fonction $F : \mathbb{R}^d \to \mathbb{R}^d$. Le principal problème vient alors du fait que F est seulement définie sur un sous-ensemble de $\mathbb{R}^d$, et la méthode de Newton peut en sortir ce qui la fait échouer. Nous utilisons ici une méthode par continuation pour trouver les zéros de F. La section 5 y est consacrée, et montre également comment le problème du domaine de définition est résolu.

3.4. Complétude et correction

Soit v une solution de G et $(f_0^+,...,f_{d-1}^+)$ la fonction évaluant les contraintes $(c_0^+,...,c_{d-1}^+)$ ajoutées dans G', il existe des paramètres $h = (h_0,...,h_{d-1})$ tels que $h_i = f_i^+(v)$. Comme v est solution de G, on a $f_i^-(v) = k_i^-$ pour $0 \leq i \leq d-1$, donc il existe une branche du plan de construction de G', noté Cp, sur laquelle $F(h) = 0$.

D'autre part, si h satisfait $F(h) = 0$ alors il existe une branche sur laquelle $Cp(h)$ est une solution v de G' vérifiant $f_i^-(Cp(k^+)) = k_i^-$ pour $0 \leq i \leq d-1$, donc v est solution de G.

4. Algorithme de décomposition avec reparamétrisation

4.1. Décomposition

Notre objectif est ici de lier propagation des degrés de liberté, reparamétrisation et décomposition de GCS.

L'algorithme suivant décompose le problème représenté par G en sous-problèmes éventuellement reparamétrés. Il met en jeu alternativement les deux algorithmes 1 et 2 vus plus haut.

L'utilisation d'une propagation avant des degrés de liberté impose la stratégie *bottom-up* suivie par cet algorithme.

Algorithme 3 Décomposition et reparamétrisation

Entrées: $G = \langle V, E \rangle$

Tant que tous les sommets ne sont pas marqués **faire**

- Appliquer l'algorithme 1 sur G autant que possible, pour chaque sous-système G_1 contenant des sommets internes. Dans G, remplacer, G_1 par son bord.
- Appliquer l'algorithme 2 sur G jusqu'à trouver un sous-système G_1 contenant des sommets internes. Remplacer G_1 par son bord.

fin Tant que

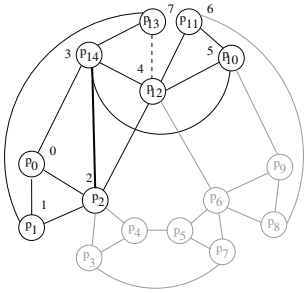


Figure 8: *Première étape de l'algorithme 2*

Sur l'exemple de la figure 1, une propagation arrière aurait conduit d'emblée à la suppression d'une contrainte, le graphe initial ne possédant que des sommets de degré au moins 3. L'algorithme 3, en revanche, utilise LIM en propagation avant autant que possible avant de recourir à une reparamétrisation. Après avoir fixé le repère $p_1 l_1$, par exemple, la construction se poursuit avec la droite l_2, les points p_2, p_5 et s'arrête faute de primitives directement constructibles. Les sommets internes p_1, l_1, l_2 sont supprimés et la distance $p_2 p_5$ ajoutée. Le graphe restant peut ensuite être parcouru sans reparamétrisation. Notons que d'autres décompositions sont possibles avec, par exemple, d'abord le triangle $p_2 p_3 p_4$ et finalement le triangle $p_2 p_4 p_5$.

Considérons maintenant l'exemple de la figure 6. L'algorithme 1 seul ne parvient pas à produire de sous-graphes avec des sommets internes. L'algorithme 2 est alors utilisé. La figure 8 illustre ce comportement après avoir fixé le repère p_0, p_1. Les sommets entourés en gras sont construits dans une première passe. Le nombre à côté de chaque sommet indique l'ordre de construction. L'arête $p_{14}\ p_2$ indique une distance ajoutée. L'ajout du point p_{13} rend le sous-graphe bien contraint. Ce point est lié au reste du graphe par 3 contraintes de distance. Deux d'entre elles serviront à la construction effective de ce point. La troisième sera rattrapée, lors de la phase numérique, en faisant varier la distance ajoutée $p_{14} p_2$.

Les sommets internes sont ensuite supprimés et le bord complété (Cf. figure 9). Ici, le bord contient les points p_2, p_{10}, p_{11}, p_{12} ainsi que 5 contraintes de distance. Parmi celles-ci, 4 figurent déjà dans le système initial, seule la

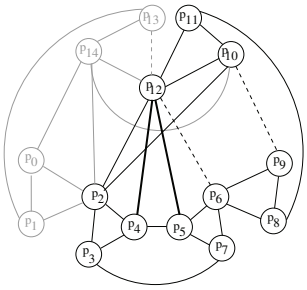


Figure 9: *Seconde étape de l'algorithme 2*

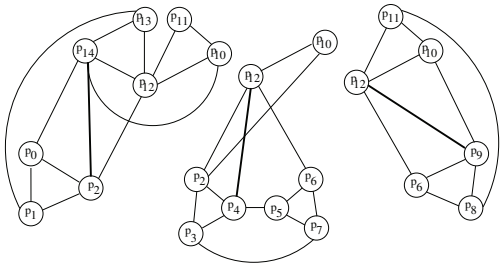


Figure 10: *Résultat d'une décomposition avec reparamétrisation*

distance p_2p_{10} est ajoutée. En choisissant p_2 et p_{12} pour nouveau repère, l'ensemble des sommets restants peut être construit en ajoutant deux contraintes qui seront alors à rattraper.

Il est toutefois possible de n'introduire que deux paramètres guides pour rendre ce problème soluble par LIM. A ce jour, la méthode naïve consistant à considérer toutes les possibilités est la seule méthode connue pour minimiser le nombre de paramètres guides. Le coût en temps de cette méthode est exponentielle avec le nombre de sommets mais la recherche de ce nombre minimum n'est pas forcément une quête judicieuse. En effet, l'introduction de 2 paramètres guides conduisent, lors de la phase numérique, à rechercher les zéros d'une fonction à deux variables. Il est préférable dans cette phase de chercher à réduire le nombre de variables des fonctions. Ainsi, on préféra 3 paramètres guides mais chacun dans un sous-graphe, ce qui amène à chercher les zéros de 3 fonctions à une seule variable. Ainsi pour une décomposition $G_1, \dots G_n$ où G_i peut contenir des contraintes ajoutées. Soit m le nombre maximum de contraintes ajoutées dans les sous-graphes G_i, l'objectif est de minimiser m. C'est précisément ce que s'attache à réaliser l'heuristique suivante.

4.2. Répartition des paramètres guides

Pour répartir au mieux des paramètres guides dans les sous-systèmes, il faudrait connaître tous les sous-graphes impliquant $2n-3$ contraintes. Cette recherche est malheureusement d'un coût exponentiel comme l'a montré [HLS97]. C'est pourquoi nous avons recours ici à une heuristique.

Supposons que G soit un système bien contraint avec une décomposition maximum en deux sous-graphes G_1 et G_2, chacun contenant des sommets internes et chacun nécessitant une reparamétrisation avec une seule contrainte. Le raisonnement suivant se généralise à davantage de sous-graphes. L'objectif est ici de trouver la répartition des deux paramètres dans les deux sous-systèmes.

Nous savons que G_1 est bien-contraint. Par ailleurs, G_2 est bien contraint si $G_{1\cap 2}$ est bien contraint ; il est sous-contraint sinon. Bien entendu, l'algorithme 3 démarre sans connaître cette décomposition. Idéalement, si un repère est fixé dans G_1, on peut espérer isoler G_1. Mais supposons qu'un repère soit fixé dans G_2 et soit Gf le premier sous-graphe déterminé par 3, nous avons alors deux cas :

- soit $G_{1\cap 2}$ est bien contraint, dans ce cas $Gf = G_2$,
- soit au contraire $G_{1\cap 2}$ n'est pas bien-contraint, ce qui est le cas le plus courant en CAO, et alors $Gf = G$

En effet, dans ce dernier cas, comme G_2 est sous-contraint, toute propagation démarrant à l'intérieur G_2 ne peut s'arrêter qu'après avoir marqué tous les sommets de G_1 pour produire le sous-graphe bien contraint Gf. Ainsi, si des contraintes on été ajoutées dans G_2, alors la dernière contrainte retirée appartenait à G_1. Cette remarque conduit à l'heuristique suivante : si deux ou plus paramètres guides sont ajoutés dans un sous-graphe Gf, l'algorithme 3 est relancé sur Gf mais en partant d'un nouveau repère impliquant la dernière contrainte retirée. Alors, le marquage des sommets a de meilleures chances d'isoler G_1 comme composante rigide modifiée.

Revenons à l'exemple de la figure 9. La méthode heuristique amène à reconsidérer le second sous-graphe car celui-ci contient deux paramètres guides. Une construction s'appuyant sur le repère p_8p_9, la distance p_8p_9 ayant été supprimée dans une première passe, conduit à une meilleure décomposition. Les arêtes en gras sur la figure 10 correspondent aux 3 paramètres guides, chacun dans un sous-graphe.

4.3. Terminaison et complexité

À chaque itération, l'algorithme extrait un sous-graphe et le remplace par son bord. Soient n et e le nombre de sommets et d'arètes du graphe de contraintes. L'algorithme 1 met en œuvre $O(n^2+e)$ opérations s'il résout le problème. Dans chaque boucle, les sommets restant sont parcourus, un sommet v est ôté et ses voisins sont mis à jour. La boucle est itérée n fois et la mise à jour des voisins nécessite de

visiter toutes les arêtes adjacentes. L'algorithme 1 échoue quand tous les repères possibles ont été essayés. Dans le plan, le nombre de repères est en $O(e)$ et pour chacun d'eux tous les sommets sont parcourus. Dans ce cas, le nombre d'opérations effectuées est en $O(ne)$. Or dans un graphe de contraintes e est proportionnel à n, on a $e = O(n)$ et l'ensemble du processus donne tous les sous-graphes possibles avec un coût de $O(n^2)$. Dans l'espace, un repère est formé par une combinaison de trois contraintes donc le nombre de repères est au plus en $O(e^3)$, donc l'algorithme a un coût en $O(n^3)$. Remarquons qu'en 3D, un repère peut contenir une contrainte ajoutée quand aucun repère ne peut être extrait du graphe de contrainte original (par exemple dans le cas d'un dodécaèdre).

Le bord est calculé en ajoutant des contraintes entre les primitives du bord. En 2D, un bord structurellement bien contraint peut être produit par des constructions de Henneberg (voir [GS09]). Dans l'espace, [TSM*11] présente un algorithme glouton pour déterminer ce bord.

L'algorithme 2 diffère d'un algorithme de propagation avant des degrés de liberté uniquement par le fait que si aucun sommet ne peut être construit, des nouvelles contraintes sont ajoutées, et les coûts de ces deux algorithmes sont les mêmes. En appliquant l'heuristique de 4.2, l'algorithme 2 est appelé $k < n$ fois où k est le nombre de contraintes supprimées (2 ou 3 en pratique). Donc, dans le pire des cas, l'algorithme général a un coût en $O(n^3)$. Dans la plupart des exemples de CAO, son coût est en $O(n^2)$.

5. Résolution numérique

La phase symbolique décrite précédemment fournit pour un GCS G plusieurs GCS solubles indépendamment par LIM pouvant contenir chacun des paramètres guides. La phase numérique a pour objectif de trouver pour chacun de ces sous-systèmes les valeurs des paramètres guides pour lesquelles les solutions de ces sous-systèmes, une fois assemblées, sont une solution de G.

Pour simplifier le discours et les notations, on considère ici que G n'est pas décomposable, et que l'on a obtenu par reparamétrisation un GCS G' et son plan de construction Cp en ajoutant à (respectivement supprimant de) G d contraintes $c_0^+,...,c_{d-1}^+$ (resp. $c_0^-,...,c_{d-1}^-$) de paramètres $k_0^+,...,k_{d-1}^+$ (resp. $k_0^-,...,k_{d-1}^-$). Les paramètres guides dont dépend Cp sont donc $k^+ = (k_0^+,...,k_{d-1}^+)$. On construit alors la fonction $F : \mathbb{R}^d \to \mathbb{R}^d$ comme indiqué plus haut.

Nous utilisons une méthode par continuation pour trouver les zéros du système $\mathcal{F}_1$ de d équations à d inconnues définit par $F(h) = 0$, dont le principe est le suivant. Connaissant une ou plusieurs des solutions d'un système $\mathcal{F}_0 : F_0(h) = 0$ en les mêmes inconnues que $\mathcal{F}_1$, appelé *système de départ*, $\mathcal{F}_0$ est déformé continûment en $\mathcal{F}_1$, appelé *système cible*, en construisant un nouveau système dépendant d'un paramètre t, $\mathcal{H} : H(t,h) = 0$ où la fonction H satisfait $H(0,h) = F_0(h)$ et $H(1,h) = F_1(h)$. Si H est $\mathcal{C}^m$, le théorème des fonctions implicites affirme qu'il existe localement, autour de chaque solution de $\mathcal{H}$ une fonction $\phi : \mathbb{R} \to \mathbb{R}^d$ telle que $H(t,\phi(t)) = 0$ où ϕ est $\mathcal{C}^{m-1}$.

Pour une solution de $\mathcal{F}_0$, l'image de la fonction ϕ fournie par le théorème des fonctions implicites est appelée un *chemin d'homotopie*. Si H est analytique, il est prouvé dans [GZ79] que ces chemins sont des variétés de dimension 1, plongées dans $\mathbb{R}^{d+1}$, difféomorphes à des intervalles de $\mathbb{R}$ ou à des cercles.

Le chemin passant par la solution donnée par l'esquisse est alors suivi jusqu'à trouver des solutions du système cible (voir la figure 11). On trouvera une description détaillée des méthodes par continuation dans [AG93].

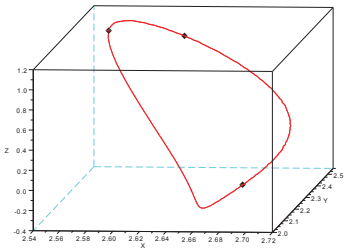


Figure 11: *Un chemin d'homotopie pour un problème nécessitant le rattrapage de 2 contraintes. Le point en $Z = 0$ est la solution initiale, les deux points en $Z = 1$ sont deux solutions trouvées lors du suivi du chemin d'homotopie.*

5.1. Méthode par continuation des paramètres

Dans le cas de la résolution de problèmes (re)paramétrés, la continuation peut être réalisée sur les paramètres du système. [MS89] justifie cette méthode dans le cas de systèmes polynômiaux ; montrons maintenant comment elle peut être adaptée à notre problème.

Une idée empruntée à [LM95] consiste à utiliser l'esquisse fournie par l'utilisateur pour construire le système de départ. Sur cette esquisse, que l'on note sk, toutes les contraintes de G sont satisfaites, mais avec d'autres paramètres. Notons k_{sk} le vecteur des paramètres des contraintes de G satisfaites par l'esquisse en remarquant que certaines de ses composantes correspondent aux paramètres des contraintes supprimées. Comme le plan de construction Cp du problème reparamétré G' dépend aussi des paramètres de toutes les contraintes de G (à l'exception de k^-), il en va de même pour la fonction F. On considère alors la fonction $F_k(h)$ qui dépend de k, le vecteur des paramètres de toutes les contraintes. On peut alors lire sur sk les paramètres des contraintes ajoutées h_{sk} et on a $F_{k_{sk}}(h_{sk}) = 0$.

Le système de départ est alors définit par $\mathcal{F}_0 : F_{k_{sk}}(h) = 0$, et le système cible par $\mathcal{F}_1 : F_{k_*}(h) = 0$, où k_* est le vecteur des paramètres des contraintes à résoudre. La fonction d'homotopie $H : \mathbb{R}^{d+1} \to \mathbb{R}^d$ est alors obtenue en déformant les paramètres k_{sk} en k_* par $H(t,h) = F_{k(t)}(h)$ où $k(t) = (1-t)k_{sk} + tk_*$ pour $t \in \mathbb{R}$.

Ici, aucune expression analytique de H n'a été calculée, puisque la fonction F est évaluée en chaque vecteur en interprétant le plan de construction *Cp* (voir 3.1). Cependant, comme chacune des instructions de *Cp* correspond à une intersection de lieux géométriques,on peut écrire ce plan de construction comme un système d'équations triangulaire par blocs, où chaque bloc correspond à une intersection associée à une primitive géométrique. Comme chaque équation de ce système est analytique et que les fonctions f_i^- évaluant les contraintes supprimées le sont aussi, H est analytique, et notamment $\mathcal{C}^\infty$ sur son domaine de définition.

Les chemins d'homotopie de H sont donc des variétés de dimension 1, et peuvent être suivies depuis les solutions connues de $\mathcal{F}_0$ jusqu'aux solutions de $\mathcal{F}_1$. Ici, une seule solution du système de départ, h_{sk}, est connue, et 5.3 montre comment suivre son chemin d'homotopie.

5.2. Les branches du plan de construction

Rappelons que $F_k(h)$ et donc H, sont des multi-fonctions (voir 3.3), et leur domaine de définition dépend de la branche considérée. Nous cherchons une caractérisation géométrique du bord de ce domaine, sur une certaine branche.

Tout d'abord, le domaine de définition de $F_k(h)$ est une intersection finie d'ensembles de définition. En effet, considérons la i-ème étape d'un plan de construction dépendant de n paramètres. Pour un ensemble de primitives construites avec les paramètres $k_0, ..., k_{m_{i-1}}$, des lieux, et l' intersection correspondante, sont calculés grâce aux paramètres $k_{m_{i-1}}, ..., k_{m_i}$. Toutefois, cette intersection n'existe que pour certaines valeurs des paramètres, disons $(k_{m_{i-1}}, ..., k_{m_i}) \in \mathcal{D}_i \subset \mathbb{R}^{m_i - m_{i-1}}$. On dira que la i-ème instruction du plan de construction est définie si les lieux construits se coupent et qu'elle ne l'est pas s'ils sont disjoints ou confondus.De la même façon, les instructions précédentes sont définies pour $(k_0, ..., k_{m_{i-1}}) \in \mathcal{D}_{i-1} \subset \mathbb{R}^{m_{i-1}}$. La figure produite à la i-ème étape existe donc pour $(k_0, ..., k_{m_{i-1}}, ..., k_{m_i}) \in (\mathcal{D}_{i-1} \times \mathbb{R}^{m_i - m_{i-1}}) \cap (\mathcal{D}_i \times \mathbb{R}^{m_{i-1}})$.

Prenons un point $x = (k,h)$ du bord du domaine de définition $\mathcal{D}$ de $F_k(h)$. Comme $\mathcal{D}$ peut s'écrire $\mathcal{D} = \cap_{0 \leq i \leq l}(\mathcal{D}_i \times \mathbb{R}^{n-(m_i - m_{i-1})})$ où l est le nombre d'instructions du plan de construction, il existe un i tel que x appartienne au bord de $\mathcal{D}_i$. Il reste alors à caractériser les bords des domaines de définition $\mathcal{D}_i$, c'est-à-dire les points $x = (k,h)$ tels que toute boule centrée en x contienne à la fois des points où la i-ème instruction est définie et des points où elle ne l'est pas.

Considérons à présent le cas d'un univers géométrique 2D contenant des points, des droites, des distances et des angles (ce qui suit s'étend en 3D). On distingue deux types de points x du bord $\mathcal{D}$ d'une instruction du plan de construction. Ceux vérifiant $x \in \mathcal{D}$ et ceux tels que $x \notin \mathcal{D}$. On vérifie facilement que le premier type de points du bord correspond à des cas de tangence cercle-cercle ou cercle-droite, et le second à des situations où deux droites sont confondues ou parallèles, ou deux cercles sont concentriques et de même rayon.

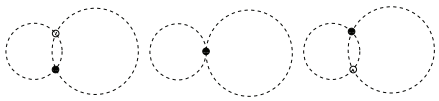


Figure 12: *La i-ème instruction du plan de construction qui consiste en l'intersection de deux cercles impose un choix entre deux solutions. Les deux figures alors construites sont les mêmes si ces cercles sont tangents.*

Si $x = (k,h)$ est sur le bord de $\mathcal{D}$ et $x \in \mathcal{D}$ alors il existe une autre branche du plan de construction sur laquelle la fonction $F_k(h)$ prend la même valeur que sur la branche actuelle. En effet, d'après ce qui a été dit plus haut, dans la figure produite par le plan de construction, deux cercles, ou un cercle et une droite, sont tangents. Si ces deux lieux sont produits par la i-ème instruction, les deux branches induites par cette instruction mènent à la construction de la même figure et $F_k(h)$ prend la même valeur sur ces deux branches. Si un chemin d'homotopie passe par un tel point x et sort du domaine de définition, on peut donc trouver une branche sur laquelle un autre chemin passe par le point x telle que ces deux chemins soient raccordables par continuité.

Les figures 12 et 13 illustrent le cas d'un suivi de chemin passant par un point du bord du domaine de définition où deux cercles deviennent parallèles. Avant de croiser ce bord (à gauche sur les figures), le chemin était suivi sur la branche induite par l'intersection marquée d'un point noir. Sur le bord (au centre sur la figure 12), les deux intersections sont confondues. En choisissant la branche induite par l'autre intersection, on peut continuer le suivi du chemin (à droite sur les figures), le mouvement des primitives des solutions ainsi que le chemin sont continus.

5.3. Suivi de courbe

Il y a principalement deux méthodes pour suivre une courbe définie par d équations indépendantes en $d+1$ variables. Ici, la courbe est définie par le système $H(x) = 0$, avec $x = (t,h)$. Comme on connaît un point x_0 vérifiant $H(x_0) = 0$, x_0 appartient à la courbe suivie.

La première méthode consiste à calculer le vecteur tangent t à la courbe en x_0 puis à calculer une solution prédite $x_0' = x_0 + \delta t$ où δ est un scalaire appelé pas de prédiction.

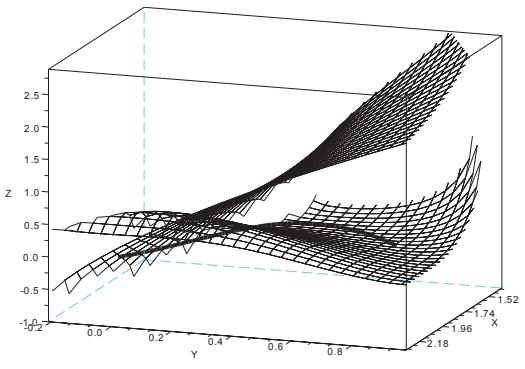


Figure 13: *Le chemin suivi, en trait plein, atteint le bord du domaine de définition et passe sur une nouvelle branche de la fonction.*

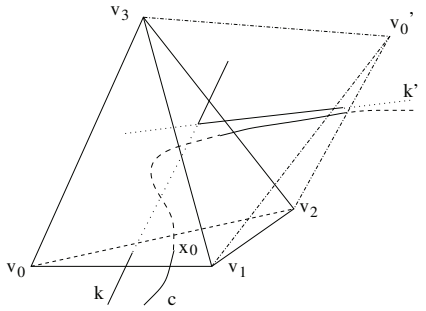


Figure 14: *Deux itérations de la méthode PLI pour suivre la courbe c. Dans $\langle v_0, v_1, v_2, v_3\rangle$ (resp. $\langle v'_0, v_1, v_2, v_3\rangle$), H^A a pour noyau k (resp. k').*

x'_0 est alors corrigé en x_1 satisfaisant $H(x_1) = 0$ en ajoutant au système une contrainte assurant l'avancement sur la courbe (par exemple x_1 appartient à l'hyperplan passant par x'_0 de vecteur normal t). La correction est effectuée par une méthode numérique itérative comme Newton-Raphson. Ces deux étapes sont répétées pour obtenir d'autres points de la courbe. L'enjeu de cette méthode est de choisir un pas de prédiction adapté à chaque courbe suivie, ce qui est abordé en utilisant l'arithmétique par intervalles dans [FM07] et [KX94].

La seconde méthode, utilisée ici, calcule un simplexe de dimension $d+1$ tel que x_0 appartienne à une de ses faces. Sur ce simplexe, H est approximée par une application affine permettant de trouver l'unique autre face du simplexe par laquelle passe la courbe. L'opération est répétée en définissant un nouveau simplexe partageant cette face avec le premier. Cette méthode, appelée interpolation linéaire par morceaux (PLI) est détaillée dans [AG97]. Nous présentons brièvement la version proposée par [DWLT90].

5.3.1. Interpolation linéaire par morceaux

Un simplexe de $\mathbb{R}^{d+1}$ est l'enveloppe convexe de $d+2$ points $v_0, ..., v_{d+1}$ de $\mathbb{R}^{d+1}$ tels que pour tout $0 \leq i \leq d+1$, v_i n'appartienne pas à l'hyperplan $\mathcal{H}_i$ passant par les points $v_0, ..., v_{i-1}, v_{i+1}, ..., v_{d+1}$. La i-ème face d'un simplexe est son intersection avec $\mathcal{H}_i$. $\langle v_0, ..., v_{d+1}\rangle$ forme un repère affine de $\mathbb{R}^{d+1}$, (i.e. la famille composée des vecteurs $v_0v_1, v_0v_2, ..., v_0v_{d+1}$ forme une base de $\mathbb{R}^{d+1}$) et l'évaluation de H en chacun des points de ce repère, si H est définie en ces points, détermine une unique application affine H^A. On notera $\langle v_0, ..., v_{d+1}\rangle$ le simplexe déterminé par les points $(v_0, ..., v_{d+1})$.

Supposons à présent que x_0 vérifiant $H(x) = 0$ appartienne à la face i de $\langle v_0, ..., v_{d+1}\rangle$. Sur $\langle v_0, ..., v_{d+1}\rangle$, la courbe suivie est assimilée au noyau de H^A qui est au moins de dimension 1 d'après le théorème du rang. Si le noyau de H^A est de dimension 1 c'est une droite, et on considère le segment $[a,b]$ définit comme l'intersection de $Ker(H^A)$ avec $\langle v_0, ..., v_{d+1}\rangle$. Si $a \in \mathcal{H}_i$ alors $\exists j \neq i \mid b \in \mathcal{H}_j$ et j est la face du simplexe par laquelle sort la courbe. On peut alors calculer un nouveau simplexe ayant sa j-ème face égale à cette dernière face, puis recommencer avec ce nouveau simplexe.

La figure 14 présente deux itérations de la méthode pour suivre une courbe de $\mathbb{R}^3$. La courbe entre dans $\langle v_0, v_1, v_2, v_3\rangle$ par la face 3 et le noyau k de l'application affine approximant H est calculé. Ce noyau intersecte la face 0 du simplexe. Dans le nouveau simplexe $\langle v'_0, v_1, v_2, v_3\rangle$ partageant sa face 0 avec s_0, un nouveau noyau k' est calculé et ainsi de suite.

Le cas où la matrice de H^A n'est pas de rang maximal, donc où son noyau est de dimension supérieure à 1, est traité dans [AG97]. En pratique, on génère les simplexes au coup par coup par réflexion : si $Ker(H^A)$ sort de $\langle v_0, ...v_{d+1}\rangle$ par la j-ème face, on prend comme nouveau simplexe $\langle v_0, ..., v_{j-1}, v'_j, v_{j+1}, ..., v_{d+1}\rangle$, où v'_j est le symétrique de v_j par rapport au centre de gravité de la face j. Notons que les calculs sont effectués en coordonnées barycentriques dans le repère formé par les sommets du simplexes courant.

5.3.2. Gestion du changement de branche

La courbe suivie peut sortir du domaine de définition et l'analyse de la figure produite par le plan de construction permet de détecter à l'avance une telle situation, avec l'instruction critique et, si elle existe, la nouvelle branche sur laquelle continue le chemin. Nous présentons maintenant une heuristique pour gérer ce changement de branche, dans le cas où l'instruction mise en défaut consiste en une intersection faisant intervenir un ou plusieurs cercles (resp. sphères) pour un GCS dans le plan (resp. l'espace).

Cette instruction amène donc à choisir entre deux intersections (voir figure 12 pour le cas de l'intersection de deux

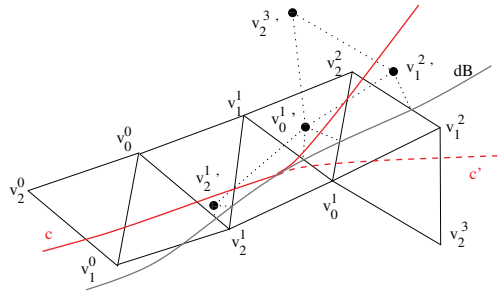


Figure 15: *Le suivi par PLI d'une courbe c passant d'une branche du plan de construction à une autre.*

cercles), que l'on nommera s_1 et s_2. On notera H_1 (resp. H_2) la fonction H sur la branche induite par le choix de s_1 (resp. s_2), B_1 (resp. B_2) le domaine de définition de H_1 (resp. H_2) et on suppose qu'avant de rencontrer dB_1 (le bord de B_1, la courbe était suivie sur la branche induite par s_1. Soit $x_* \in dB_1$ vérifiant $H_1(x_*) = 0$.

Le caractère symétrique de s_1 et s_2 nous amène à considérer que dans un voisinage de x_*, $B_1 = B_2$ et $dB_1 = dB_2$, où dB_2 est le bord de B_2, et nous noterons $B = B_i$ et $dB = dB_i$ pour $i = 1$ et 2. Dans ce même voisinage de x_*, on approxime dB par un hyperplan $\mathcal{P}$ et on considère que $x \in dB \Rightarrow H_1(x) = H_2(x)$. On construit alors dans ce voisinage une nouvelle fonction $H' : \mathbb{R}^{d+1} \to \mathbb{R}^d$ définie par :

$$H'(x) = \begin{cases} H_1(x) & \text{si } x \in B, \\ H_2(\Delta_{\mathcal{P}}(x)) & \text{si } x \notin B, \end{cases}$$

où $\Delta_{\mathcal{P}}(x)$ est le symétrique de x par rapport à $\mathcal{P}$.

Lors du suivi de la courbe c par PLI, on détecte que c s'approche du bord du domaine de définition quand H n'est pas définie en un ou plusieurs des sommets $(v_0, ..., v_{d+1})$ du simplexe courant. Après avoir détecté l'instruction de plan de construction en cause, les deux intersections s_1 et s_2 possibles, les deux fonctions H_1 et H_2 décrites précédemment et l'hyperplan $\mathcal{P}$ on applique la méthode PLI à la fonction H' construite à partir de H_1 et H_2. On suit alors une courbe c' qui est la symétrique de c par rapport à $\mathcal{P}$. Lorsque tous les sommets sont en dehors de B, on calcule les symétriques de ces sommets par rapport au bord pour construire un nouveau simplexe en lequel H_2 est complètement définie, pour continuer à suivre la courbe définie par $H_2 = 0$.

À chaque étape, l'hyperplan $\mathcal{P}$ approximant dB est calculé en considérant les arêtes ab du simplexe telles que $a \in B$ et $b \notin B$. Il y en a au moins d si au moins un des sommets est en dehors de B. On recherche alors par dichotomie pour d de ces arêtes ab un point $a' \in B$ tel que la distance de a' à dB sur l'arête ab soit plus petite qu'un $\lambda \in \mathbb{R}$ donné. Ces d points définissent l'hyperplan $\mathcal{P}$ approximant dB.

La figure 15 illustre cette procédure dans le cas d'une courbe c de $\mathbb{R}^2$. c sort de $\langle v_0^0, v_1^0, v_2^0 \rangle$ par la face 2, et le nouveau sommet v_2^1 calculé est hors du domaine de définition B de H. L'hyperplan $\mathcal{P}$ approximant dB est alors calculé grâce aux arêtes $v_0^0 v_2^1$ et $v_1^0 v_2^1$, et H est approximé par l'application affine valant $H_1(v_1^0)$ en v_1^0, $H_1(v_0^0)$ en v_0^0 et $H_2(v_2^{1,})$ en v_2^1, où $v_2^{1,}$ est le symétrique de v_2^1 par rapport à $\mathcal{P}$. Le noyau de cette application sort de $\langle v_0^0, v_1^0, v_2^1 \rangle$ par la face 1. À l'itération suivante, c sort du simplexe $\langle v_0^0, v_1^1, v_2^1 \rangle$ par la face 0, et le nouveau sommet v_0^1 n'est pas dans B. On continue ainsi jusqu'à ce qu'un simplexe ait tous ses sommets en dehors de B. Alors le nouveau simplexe est $\langle v_0^{1,}, v_1^{2,}, v_2^{3,} \rangle$ et la fonction considérée est H_2.

5.4. Pistes de travail

La méthode PLI et son adaptation aux changements de branche décrite ci-dessus est pour le moment limitée aux intersections entre un cercle (ou une sphère pour les GCS dans l'espace) et d'autres lieux géométriques. Son extension à des lieux géométriques quelconques est un de nos axes de travail. D'autre part, cette méthode suppose que le domaine de définition de la fonction définissant la courbe soit convexe. Hors il est facile de construire un exemple où ça n'est pas le cas. Il faudrait alors, pour un point de la courbe donnée, pouvoir déterminer un voisinage de ce point dans lequel le domaine de définition est convexe. Enfin, le problème de la taille des simplexes lors du suivi de la courbe reste posé. En effet, si le simplexe est trop grand, certaines variations de la courbe ne sont pas détectées, notamment quand la courbe entre et sort par une même face. Il apparaît pertinent de pouvoir adapter la taille du simplexe à la courbe suivie.

6. Conclusion

Dans le cadre de la résolution de systèmes de contraintes géométriques, les méthodes constructives jouissent de propriétés intéressantes, mais elles ne sont cependant pas assez puissantes pour résoudre de nombreux problèmes de type CAO. Ce phénomène est particulièrement marqué dans le cas de la 3D. Du point de vue inverse, les méthodes numériques efficaces ne fournissent qu'une solution du problème, qui peut ne pas satisfaire l'utilisateur. Les méthodes à base de reparamétrisation permettent de contourner ce problème en remplaçant certaines contraintes du système donné pour le rendre soluble constructivement. Il faut alors faire face à la difficulté du rattrapage numérique de ces contraintes.

Dans cet article, nous avons décrit un algorithme choisissant ces contraintes de façon à rendre le système décomposable pour obtenir des sous-problèmes avec peu de contraintes remplacées, grâce à une stratégie de répartition des contraintes ajoutées entre les sous-problèmes. Les paramètres des contraintes ajoutées menant aux solutions du problème à résoudre sont obtenues en déformant continûment l'esquisse fournie par l'utilisateur en une solution du problème grâce à un algorithme de suivi de chemin, adapté au caractère géométrique du contexte.

Nous envisageons de poursuivre ces travaux en étudiant d'autres heuristiques menant à une minimisation des contraintes remplacées dans chacune des composantes rigides, et en adaptant la reparamétrisation à d'autres stratégies de décomposition comme par exemple, la W-décomposition de [TSM*11]. Du point de vue de la résolution numérique, seules les solutions se trouvant sur le même chemin d'homotopie que la solution fournie par l'esquisse sont trouvées. Nous cherchons un moyen de trouver les autres solutions.

Références

[AG93] ALLGOWER E., GEORG K. : Continuation and path following. *Acta Numerica. Vol. 2*, Num. -1 (1993), 1–64.

[AG97] ALLGOWER E., GEORG K. : Numerical path following. *Handbook of Numerical Analysis. Vol. 5*, Num. 3 (1997), 207.

[Brü93] BRÜDERLIN B. : Using geometric rewrite rules for solving geometric problems symbolically. *Theoritical Computer Science* (1993), 291–303.

[DMS97] DUFOURD J.-F., MATHIS P., SCHRECK P. : Formal resolution of geometrical constraint systems by assembling. *Proceedings of the 4th ACM Solid Modeling conf.* (1997), 271–284.

[DWLT90] DOBKIN D., WILKS A., LEVY S., THURSTON W. : Contour tracing by piecewise linear approximations. *ACM Transactions on Graphics (TOG). Vol. 9*, Num. 4 (1990), 389–423.

[FM07] FAUDOT D., MICHELUCCI D. : A new robust algorithm to trace curves. *Reliable computing. Vol. 13*, Num. 4 (2007), 309–324.

[FS08] FABRE A., SCHRECK P. : Combining symbolic and numerical solvers to simplify indecomposable systems solving. In *ACM Symposium on Applied Computing SAC 2008* (Mar 2008), Roger L. Wainwright H. H., (Ed.), ACM, ACM Press, pp. 1838–1842. ISBN 978-1-59593-753-7.

[GHY02] GAO X.-S., HOFFMANN C. M., YANG W.-Q. : Solving spatial basic geometric constraint configurations with locus intersection. In *Proceedings of the seventh ACM symposium on Solid modeling and applications* (New York, NY, USA, 2002), SMA '02, ACM, pp. 95–104.

[GS09] GAO H., SITHARAM M. : Characterizing 1-dof henneberg-i graphs with efficient configuration spaces. In *Proceedings of the 2009 ACM symposium on Applied Computing* (New York, NY, USA, 2009), SAC '09, ACM, pp. 1122–1126.

[GZ79] GARCIA C., ZANGWILL W. : Finding all solutions to polynomial systems and other systems of equations. *Mathematical Programming. Vol. 16*, Num. 1 (1979), 159–176.

[HLS97] HOFFMANN C. M., LOMONOSOV A., SITHARAM M. : Finding solvable subsets of constraint graphs. In *CP* (1997), pp. 463–477.

[IMS11] IMBACH R., MATHIS P., SCHRECK P. : Tracking method for reparametrized geometrical constraint systems. In *13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (2011).

[JAS97] JOAN-ARINYO R., SOTO A. : A correct rule-based geometric constraint solver. *Computer and Graphics. Vol. 5*, Num. 21 (1997), 599–609.

[KX94] KEARFOTT R., XING Z. : An interval step control for continuation methods. *SIAM Journal on Numerical Analysis. Vol. 31*, Num. 3 (1994), 892–914.

[LGL81] LIN V. C., GOSSARD D. C., LIGHT R. A. : Variational geometry in computer-aided design. *SIGGRAPH Comput. Graph.. Vol. 15* (August 1981), 171–177.

[LM95] LAMURE H., MICHELUCCI D. : Solving constraints by homotopy. In *Proceedings of the ACM-Sigraph Solid Modeling Conference* (1995), ACM Press, pp. 134–145.

[LM96] LATHAM R. S., MIDDLEDITCH A. E. : Connectivity analysis : a tool for processing geometric constraints. *Computer-Aided Design. Vol. 28*, Num. 11 (1996), 917–928.

[Mic03] MICHELUCCI D. : Using cayley menger determinants. In *Proceedings of the Workshop on Geometric Constraint Solving)* (2003). Beijing, China (available at the URL http ://www.mmrc.iss.ac.cn/ ascm/ascm03/).

[MS89] MORGAN A., SOMMESE A. : Coefficient-parameter polynomial continuation. *Applied Mathematics and Computation. Vol. 29*, Num. 2 (1989), 123–160.

[MT10] MATHIS P., THIERRY S. E. : A formalization of geometric constraint systems and their decomposition. *Formal Aspects of Computing. Vol. 22*, Num. 2 (2010), 129–151.

[TSM*11] THIERRY S. E., SCHRECK P., MICHELUCCI D., FÜNFZIG C., GÉNEVAUX J.-D. : Extensions of the witness method to characterize under-, over- and well-constrained geometric constraint systems. *Computer-Aided Design. Vol. 43*, Num. 10 (2011), 1234 – 1249.

[Zha11] ZHANG G.-F. : Well-constrained completion for under-constrained geometric constraint problem based on connectivity analysis of graph. In *SAC* (2011), pp. 1094–1099.

Un nouveau système pour créer des mouvements de caméra pour la stop motion

Laura Saini [1] Gudrun Albrecht [1] Nicolas Lissarrague [2] Lucia Romani [3]

[1]UNIV Lille Nord de France, UVHC, LAMAV-CGAO, FR no.2956, F-59313, Valenciennes, France
[2]UNIV Lille Nord de France, UVHC, CALHISTE, EA 4343, F-59313, Valenciennes, France
[3]Dip. di Matematica e Applicazioni, Università di Milano-Bicocca, Via Cozzi 53, 20125 Milano, Italia

Résumé

L'article présente un nouveau système permettant de produire des mouvements de caméra réalistes pour l'animation stop motion. Le système permettra d'enrichir les logiciels d'animation 3D classiques (comme par exemple Maya et 3D Studio Max) afin de leur faire contrôler des mouvements de caméra pour la stop motion, grâce à l'utilisation d'une interface haptique. Nous décrivons le fonctionnement global du système. La première étape consiste à récupérer et enregistrer les données envoyées par le périphérique haptique de motion capture. Dans la seconde étape, nous réélaborons ces données par un procédé mathématique, puis les exportons vers un logiciel de 3D pour prévisualiser les mouvements de la caméra. Finalement la séquence est exécutée avec un robot de contrôle de mouvement et un appareil photo. Le système est évalué par un groupe d'étudiants du Master "Art plastiques et Création numérique" de l'Université de Valenciennes.

Abstract

The article presents a new system that allows to create realistic camera movements for a stop motion animation. The system improves traditional 3D software animation programs (for example Maya and 3D Studio Max) for creating stop motion camera movements by using an haptic interface. After describing the whole system, we explain in detail the mathematical processing to obtain different camera movements by using an haptic interface for motion capture. The recorded haptic positions, once elaborated, are exported, frame by frame, to the motion control software, which allows to calibrate the motion control robot, to control the camera settings and, finally, to execute the sequences. A class of students of the "Art plastiques et Création numérique" Master of the University of Valenciennes evaluated the system.

Keywords : Simulation réaliste, stop motion, motion control, motion capture, 3D.

1. Introduction : camera movements in a stop motion animation

Stop-motion is an animation technique that brings objects, such as puppets or clay models, alive by photographing a series of positions and then playing these as a continuous sequence. Originally, only the objects are moved in a stop-motion animation, because animating the camera is very complicated. In fact, the stop motion process is (very) long and (very) tedious. If an animator makes a mistake on the stage, it is not possible to go back and repeat parts of the movement as it can never be recaptured exactly in the same way. Moreover, it is an impossible job to move the camera frame by frame along a continuous curve to produce a *smooth* movement. Because of these constraints, the camera was strongly fixed on the real stage for a long time. But on the other hand, camera movements subject to the influence of floor irregularities, human manipulations and mechanical imperfections are mainly recognized as part of the aesthetic cinematographic specificity, and therefore desirable to a certain extent. Thus moving the camera in stop motion is referred to as stop motion camera animation. The camera shots are made frame by frame and the camera is slightly moved between frames. Once assembled, the frames produce an illusion of movement.

So far, even the traditional animation methods in 3D soft-

	Keyframing Animation	Path Constraint Animation
separation of position and speed	not possible	possible
global/local control of space trajectory	only local control	only global control
addition of constraints	possible	partially possible
space trajectory's x,y,z-coordinates accessible	yes	no

Table 1: *Advantages and disadvantages of the two main animation tools.*

ware animation programs suffer from limitations in producing realistic camera moves. The most important 3D animation software programs that can control camera moves are : Maya (see e.g. [Der09]), 3D Studio Max (see e.g. [Mur01]), Ligthwave, Blender, Cinema4D, Softimage, Houdini. These programs have two main tools to animate an object : "Keyframing Animation" and "Path Constraint Animation".

- "Keyframing Animation" is based on the traditional animation technique, where the user only sets the important frames, called keyframes. Using interpolation techniques, the software program generates the intermediate frames, called in-betweens. The object's trajectory is internally represented as a parametric space curve where the animator interacts with its three coordinate curves $x(t)$, $y(t)$ and $z(t)$ in order to change position as well as speed. Note that position and speed can not be modified separately.
- "Path Constraint Animation" separately constructs the 3D space trajectory and the so called Ease Curve that controls the object's speed.

The main advantages and disadvantages of these tools are summarized in Table 1. As far as the mathematical background is concerned there exist several interpolation techniques to fit a piecewise curve to a sequence of given points (keyframes), depending on the final motion desired. The most used techniques in animation may, e.g., be found in [VB04], [GP04] and [KB84]. In order to overcome the major disadvantage (dependence of position and speed) of the most popular animation technique, the "Keyframing Animation", several approaches aim at reparameterising the curve by arc length and thus controlling the movement along the curve by an Ease Curve, see, e.g., [ST82], [GP90], [WKA02], [Par04], [Ebe08].

Our objective is to create, by a stop motion animation technique, a 3D animation that looks as realistic as possible. We use these partial solutions proposed in the literature with the aim to overcome the existing drawbacks of the 3D animation software. We present a new motion control system specifically designed for stop motion that is able to simulate a realistic camera animation. We want to elaborate a system that gives stop motion animators total freedom of camera movement and that maintains the handwork visual aesthetics of stop motion. In particular we are aiming at simulating a 3D camera movement that can integrate constraints and imperfections (noise) of real camera devices by using an haptic interface.

The paper is organized as follows. Section 2 describes the whole system, section 3 explains in detail the mathematical processing to obtain different camera movements by using an haptic interface for motion capture. In particular, in subsection 3.1, we describe the robot's trajectory and in subsection 3.2 the camera's rotations. In section 4, we present an assessment of our system carried out with a class of students of the "Art plastiques et Création numérique" Master of the University of Valenciennes. Finally, in section 5 we present the future improvements and developments of our system.

2. A new system for camera movement in a stop motion

We describe, in this section, a new motion control system able to add constraints, by using an haptic interface, that greatly contributes to producing imperfections and the behavior of a real camera device. We use a device from Novint technologies : Novint Falcon (Figure 1), because it is relatively cheap and versatile. The whole system can be summarized in the following steps, as shown in Figure 2.

Figure 1: *Novint Falcon.*

1. *Motion capture device*
   We use the haptic interface to obtain the movement of the camera. Haptic devices are spatial input devices, which can themselves generate a force on the input point (force feedback). In our system we can control a rational parametric cubic Bézier curve, by using the [FTA10] graphical interface. The user can be guided along the curve, as shown in Figure 3. Firstly, the interactor point x is moved to the nearest point $p(t_x)$ on the curve. As common in haptics, a spring force in the direction $p(t_x)-x$ is used to achieve this. Secondly, the user can move forward in time along the curve (Figure 4). See [SAL*11] for more details. We add the possibility to determine the interactor position for every Δt of time. In order to respect the standard frequency, at which an imaging device produces

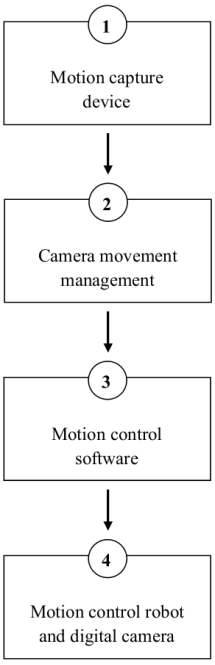


Figure 2: *Motion control system diagram.*

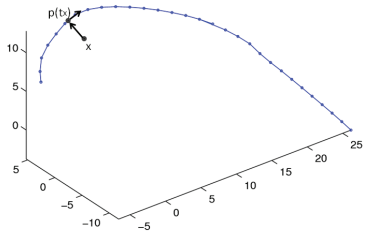


Figure 3: *Haptic input (grey) following a space curve.*

unique consecutive images, we set $\Delta t = 0.04$ seconds, to obtain 25 "frames per second". By using the button in the back of the knob, we start recording the interactor positions along the curve and then we move along the curve with the interactor. When we release the button, we stop the recording and we obtain a sequence of n positions $\mathbf{p}_i^H$, $i = 0,\ldots,n-1$, where H indicates the haptic system's positions. The sequence $\mathbf{p}_i^H$ represents the curve's parametrisation (Figure 5).

2. *Camera movement management*
The motion capture sequence $\mathbf{p}_i^H$ is not exactly on the parametric cubic curve, as we can see in Figure 6. Moreover the corresponding speed is too noisy to obtain a realistic camera movement, as shown in Figure 13. We want that the camera exactly follows the rational parametric cubic Bézier curve $\mathbf{P}(t)$, because our robot is constrained to the given trajectory. For these reasons, to obtain a sequence of points on the curve and to simulate different behaviours of a real camera device, we have to elaborate the interactor positions $\mathbf{p}_i^H$. By a mathematical processing, described in the next section, and a 3D animation software visualization, we obtain the new positions of the virtual camera that we can export frame by frame to the motion control system.
3. *Motion control software*
Using the imported data the motion control software allows to calibrate the motion control robot, to control the camera settings and, finally, to execute the sequence, as shown in Figure 7. It is based on an Arduino system (Figure 8), an open-source platform that allows, by its microcontroller, to send electronic signals to the robot and move it.
4. *Motion control robot and digital camera*
A motion control robot, on which is positioned a camera

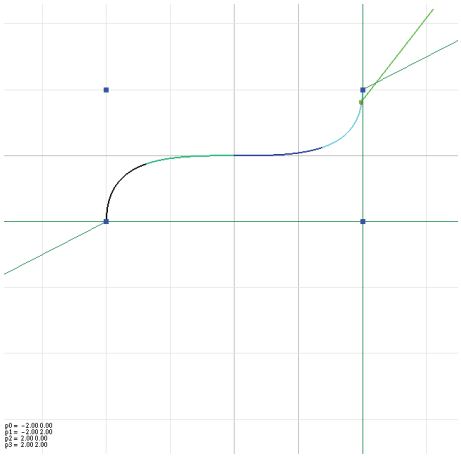


Figure 4: *Graphical interface with the curve space.*

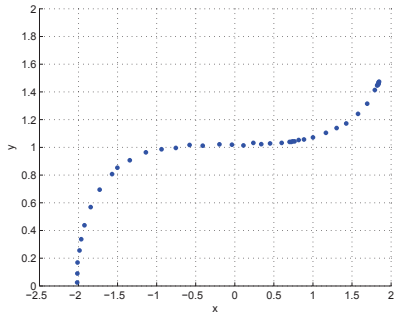


Figure 5: *Matlab's visualisation of* $\boldsymbol{p}_i^H$.

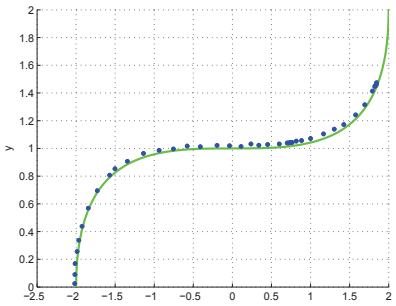


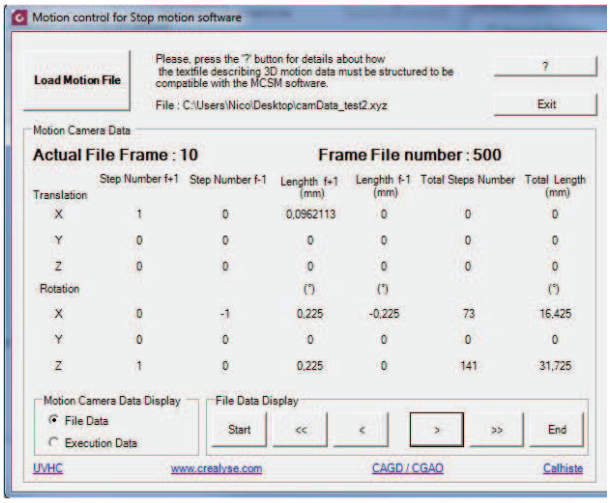
Figure 6: *A rational parametric cubic Bézier curve parameterized by using the haptic system. In blue the interactor positions* $\boldsymbol{p}_i^H$.

and which is able to move on one translation axis and two rotation axes, is used to record the video. It is affordable for any-sized budget production, handles a 2Kg camera, and has at least 1/10th of a millimeter precision for positioning and 1/10th of a degree precision for rotating. See Figure 9 for an illustration.

3. Mathematical processing

In this section we describe in detail the mathematical processing presented at the second step in section 2. In particular, to manage the speed of our robot, we use the concept of an Ease Curve, which represents arc length over time-frame. In the first step (*Projection : "haptic" curve parametrisation*) we project the motion capture sequence $\mathbf{p}_i^H$ on the rational parametric cubic Bézier curve. In the second step (*Fitting : "ideal" Ease Curve*) we determine an "ideal" speed that maintains the haptic speed, thus we can calculate the ideal curve parametrisation (*"Ideal" curve parametrisation*). In the third step (*Blending : "intermediate" Ease Curves*) we calculate different Ease Curves between the "haptic" and the "ideal" one. Finally, we determine the corresponding curve parametrisations (*"Intermediate" curve parametrisations*). Now, we describe in detail every step.

1. *Projection : "haptic" curve parametrisation*
   The sequence of points $\mathbf{p}_i^H(x_i, y_i)$ that we obtain from the haptic system is not exactly on the rational parametric cubic Bézier curve $\mathbf{P}(t)$, as we can see in Figure 10. To allow the robot to move along the given trajectory, we have to project the sequence on the curve $\mathbf{P}(t)$, as shown in Figure 11. We know the length l of $\mathbf{P}(t)$ where $t \in [0,1]$, by approximating the integral using a Gaussian quadrature or a Gaussian adaptive method, as described in [SALR10]. Thus, we can calculate, for $i = 1, \ldots, n-1$, $sp_i^H = ||\mathbf{p}_i^H - \mathbf{p}_{i-1}^H||$ and



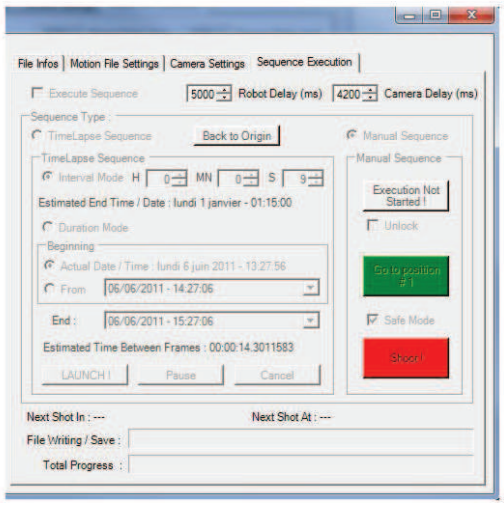


Figure 7: *Robot and camera software interface.*

$$s_i^H = \begin{cases} s_{i-1}^H + sp_i^H, & \text{if } x_i > x_{i-1} \\ s_{i-1}^H - sp_i^H, & \text{if } x_i < x_{i-1} \end{cases}$$

with $s_0^H = 0$ and x_i the abscissa of $\mathbf{p}_i^H$. We need curve parameters $\widetilde{t}_i \in [0,1]$ with $i = 0, \ldots, n-1$ to calculate the coordinates of the corresponding curve points $\widetilde{\mathbf{p}}_i^H$. A reasonable choice for the initial parameter is $\widetilde{t}_0 = \frac{||\mathbf{p}_0^H - \mathbf{P}(0)||}{l}$, because it represents the fraction of arc length at which the first point should be located. We thus want to solve the following equation :

$$F(\widetilde{t}_i) = \int_{\widetilde{t}_{i-1}}^{\widetilde{t}_i} \left|\left| \frac{d\mathbf{P}}{dt} \right|\right| dt - sp_i^H = 0,$$

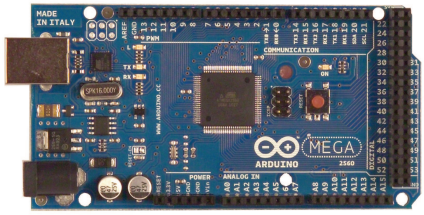


Figure 8: *Arduino Mega 2560.*

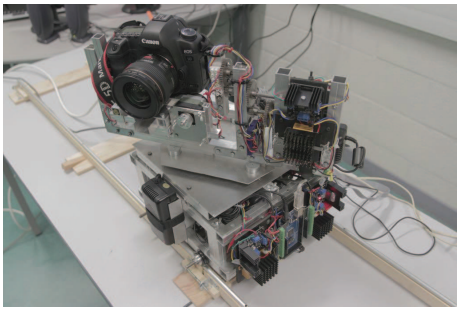


Figure 9: *Robot with 3 axes of freedom and its camera (Canon EOS 5D mark ii).*

with $i = 1,\dots,n-1$. We can solve it by using Newton's method :

$$\widetilde{t}_i^k = \widetilde{t}_i^{k-1} - \frac{F(\widetilde{t}_i^{k-1})}{F'(\widetilde{t}_i^{k-1})}, \quad k = 1,2,\dots$$

where $F(\widetilde{t}_i^{k-1})$ is approximated using standard numerical integrators and $F'(\widetilde{t}_i^{k-1})$ is straightforward because we have a formula for the rational parametric cubic Bézier curve $\mathbf{P}(t)$ and we can compute $d\mathbf{P}/dt$ from it. When F

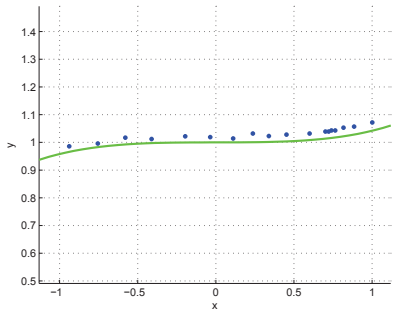


Figure 10: *Visualization of the points $\boldsymbol{p}_i^H$ (blue) and the rational parametric cubic Bézier curve $\boldsymbol{P}(t)$ (green).*

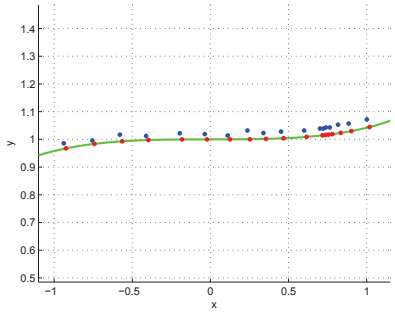


Figure 11: *Visualization of the points $\widetilde{\boldsymbol{p}}_i^H$ (red), the points $\boldsymbol{p}_i^H$ (blue) and the rational parametric cubic Bézier curve $\boldsymbol{P}(t)$ (green).*

is sufficiently near to zero or a maximum number k of iterates has been computed, $\widetilde{t}_i^k$ is accepted as $\widetilde{t}_i$ and the iteration is repeated for $i = 1,\dots,n-1$. If the calculated $\widetilde{t}_i > 1$ we set $\widetilde{t}_i = 1$. Finally, the parametric curve $\mathbf{P}(\widetilde{t}_i)$ is recalculated with the new parameter values $\widetilde{t}_i$, as represented in Figure 12. This procedure is described in detail in [SALR10].

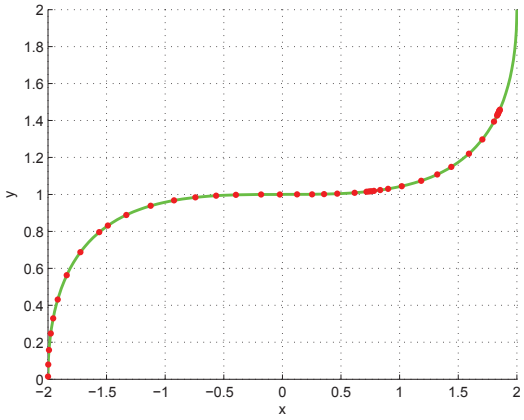


Figure 12: *Projection of the haptic positions $\boldsymbol{p}_i^H$ on the curve $\boldsymbol{P}(t)$ resulting in the red points $\widetilde{\boldsymbol{p}}_i^H$.*

2. *Fitting : "ideal" Ease Curve*
In Figure 13 we visualize the Ease Curve $S^H(f)$, which represents arc length s over time-frame f. On the $y-$axis we have the $s_i^H = S^H(f_i)$ and on the $x-$axis equally spaced frame parameter values f_i with $f_i \in [0,24]$. By analyzing the Ease Curve we observe in Figure 13, that the speed at which the curve is traversed and thus the resulting movement from the haptic system is too noisy for

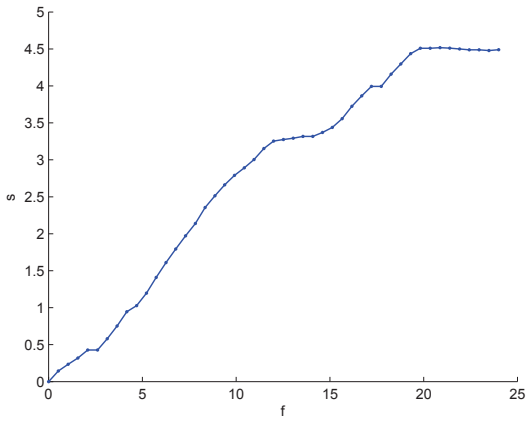


Figure 13: $S^H(f)$ *Ease Curve of* $\boldsymbol{p}_i^H$.

our purpose. So, we want to find a new parametrisation for our rational parametric cubic Bézier curve that is less noisy but still respects the haptic movement. By using the least squares method we fit a quartic polynomial to the discrete data of the Ease Curve $S^H(f)$, as represented in Figure 14. Thus, we obtain an "ideal" speed $S^I(f_i) = s_i^I$.

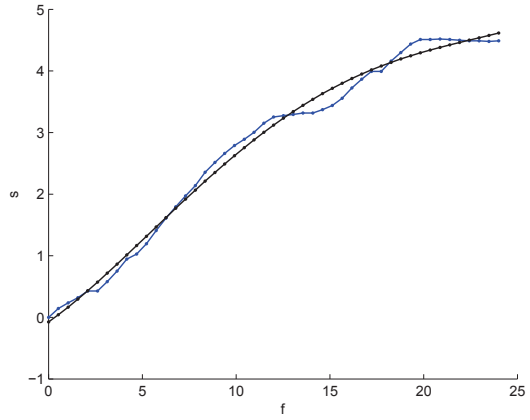


Figure 14: *Least squares Ease Curve* $S^I(f)$ *(black) of the discrete data of the Ease Curve* $S^H(f)$ *(blue) of Figure 13.*

3. *"Ideal" curve parametrisation*
   By applying Newton's method as described above and by replacing sp_i^H with $sp_i^I = |s_i^I - s_{i-1}^I|$ for $i = 1,\ldots,n-1$, we obtain an "ideal" speed, that is used to determine the new sequence of parameter values $\widetilde{t}_i^I$. Thus, we can calculate the corresponding points along the curve, to obtain the "ideal" curve parametrisation $\mathbf{P}(\widetilde{t}_i^I)$ (Figure 15).

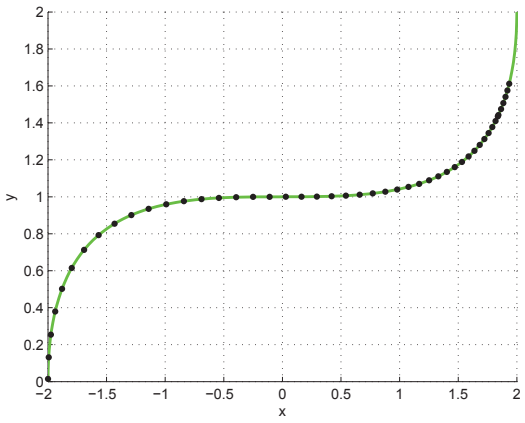


Figure 15: *"Ideal" curve parametrisation corresponding to the Ease Curve* $S^I(f)$ *of Figure 14.*

4. *Blending : "intermediate" Ease Curve*
   We want to propose several parametrisations such that an animator can choose the one that best fits his/her design intention. We thus determine an intermediate Ease Curve between the "ideal" $S^I(f)$ and the haptic one $S^H(f)$. We define

$$\lambda_i := \alpha_i \cdot \left(\frac{d_i}{||\mathbf{d}||} \right)$$

   where $d_i = |s_i^H - s_i^I|$, $\mathbf{d} = (d_0, d_1, d_2, \ldots, d_{n-1})$ and with $\alpha_i \in [0, \frac{||\mathbf{d}||}{d_i}]$. We then can calculate an "intermediate" Ease Curve $S^B(f)$ such that $s_i^B = S^B(f_i)$ by blending :

$$s_i^B = (1-\lambda_i)s_i^H + \lambda_i s_i^I,$$

   with $\alpha_i = 1\ \forall i$. If $\alpha_i = 0\ \forall i$ we have the "haptic" Ease Curve $S^H(f)$ and if $\alpha_i = \frac{||\mathbf{d}||}{d_i}\ \forall i$ we obtain the "ideal" one $S^I(f)$. This choice of λ_i assures that we respect the haptic speed $S^H(f)$, because if s_i^H is close to s_i^I, the influence of s_i^I on s_i^B is neglectable. The curve $S^I(f)$ modifies $S^H(f)$ only if s_i^H is far from s_i^I. In figure 16 we can see the "intermediate" Ease Curve $S^B(f)$.

5. *"Intermediate" curve parametrisations*
   By applying Newton's method to the "blending" Ease Curve $S^B(f)$ and by using $sp_i^B = |s_i^B - s_{i-1}^B|$ for $i = 1,\ldots,n-1$, we find the new sequence of parameter values $\widetilde{t}_i^B$. We can now calculate the corresponding points along the curve, to obtain the "intermediate" curve parametrisation $\mathbf{P}(\widetilde{t}_i^B)$, shown in Figure 17. Moreover, the use of the α_i parameter also allows to modify only a zone of the speed.

This processing, implemented in *Matlab*®, allows a great flexibility in controlling the speed along the curve, without

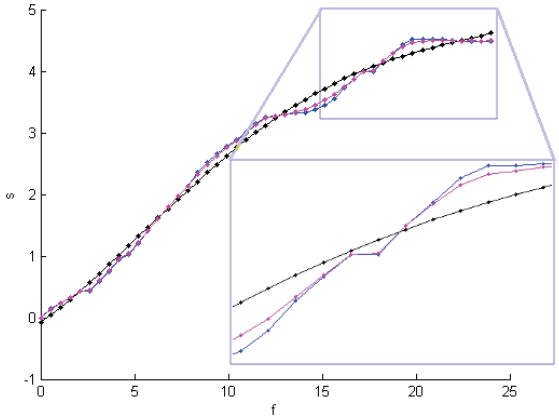


Figure 16: *In magenta the "intermediate" Ease Curve $S^B(f)$ between the "haptic" Ease Curve $S^H(f)$ (blue) and the "ideal" Ease Curve $S^I(f)$ (black).*

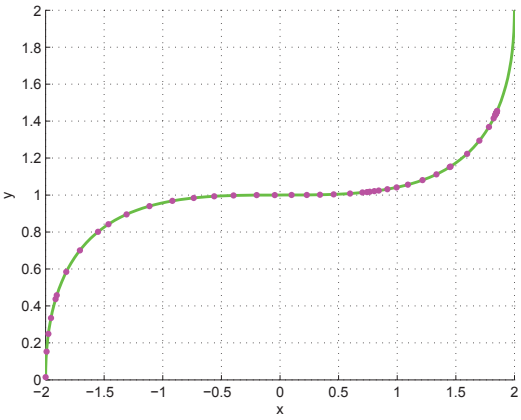


Figure 17: *"Intermediate" curve parametrisation corresponding to the Ease Curve $S^B(f)$ in Figure 16.*

modifying the space trajectory. Since our system is able to move on one translation axis and two rotation axes, we need to control two particular curves : a straight line and a semi-circle and the corresponding procedure is described in the following sections 3.1 and 3.2.

3.1. Translation

We want to control the linear translational robot movement. As shown in Figure 18, a straight line represents our robot trajectory in the graphical interface. In particular, to have an intuitive correspondence with the real robot trajectory, we set the Bézier cubic curve control points as :

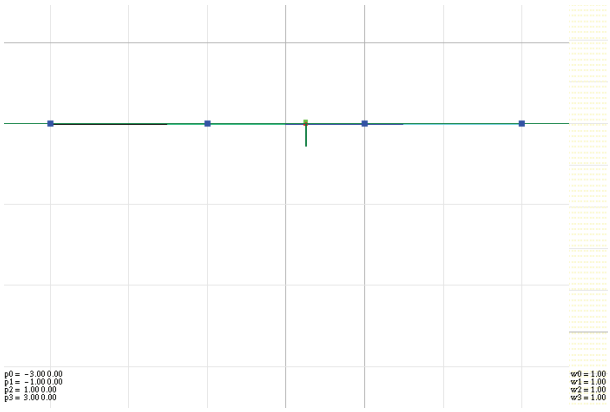


Figure 18: *Robot trajectory in the graphical interface.*

$$cp_1 = (-3,0), \quad cp_2 = (-1,0), \quad cp_3 = (1,0), \quad cp_4 = (3,0)$$

and all the weights equal to 1, as shown in Figure 18. We record the haptic positions $\mathbf{p}_i^{H_i}$ concerning the translation and we export them in *Matlab*®. Now, we follow the procedure described in section 3 to obtain different speed choices for our robot movement. We import these new curve parametrisations (haptic, ideal, intermediate) in an external script in 3D Studio Max. We visualize the trajectory and we simulate the camera's movement. An example of haptic curve parametrisation is visualized in Figure 19 and its intermediate parametrisation with $\alpha_i = \frac{||\mathbf{d}||}{d_i} \cdot 0.9 \; \forall i$ is shown in Figure 20.

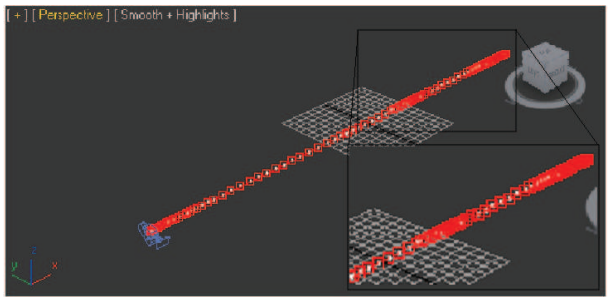


Figure 19: *Projected haptic trajectory parametrisation $\tilde{\mathbf{p}}_i^{H_i}$ visualized in 3D Studio Max.*

3.2. Rotation

The camera on our robot can move on two rotation axes. Thus, we want to control the rotation in the $yz-$plane (Figure 21) and in the $xy-$plane (Figure 22). For both planes, the

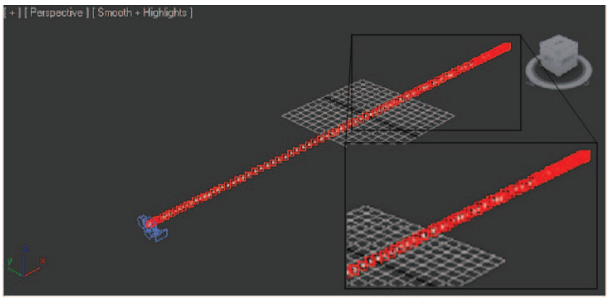


Figure 20: *Intermediate trajectory parametrisation* $\widetilde{p}_i^{B_i}$ *with* $\alpha_i = \frac{||d||}{d_i} \cdot 0.9 \; \forall i$ *visualized in 3D Studio Max.*

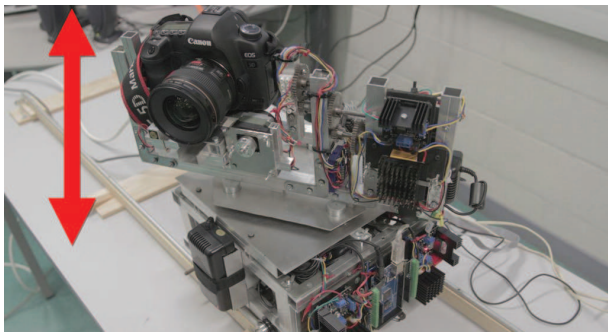


Figure 21: *Visualization of the camera rotation in the* $yz-$*plane.*

camera can rotate by an angle $\omega \in [0,\pi]$. We represent and visualize this angle by a semicircle (Figure 23). The conditions needed to reproduce a circle with a rational parametric cubic Bézier curve are given in [PT95]. We set the Bézier cubic curve control points as :

$$cp_1 = (1,0), \quad cp_2 = (1,2), \quad cp_3 = (-1,2), \quad cp_4 = (-1,0)$$

and the weights as :

$$w_1 = 1, \quad w_2 = \frac{1}{3}, \quad w_3 = \frac{1}{3}, \quad w_4 = 1.$$

The same procedure is applied to the two rotations. We determine two haptic sequences of points $\widetilde{\mathbf{p}}_i^{H_{r1}}$ and $\widetilde{\mathbf{p}}_i^{H_{r2}}$. We export them into Matlab to obtain our speed elaboration. We visualize an example of camera's rotation in the $xy-$plane in Figure 24 ("haptic" parametrisation), in Figure 25 ("ideal" parametrisation) and 26 ("intermediate" parametrisation).

4. System assessment

The evaluation of the system is very important to see if it does what it is supposed to do and if it produces visually convincing camera movements, as well as to identify

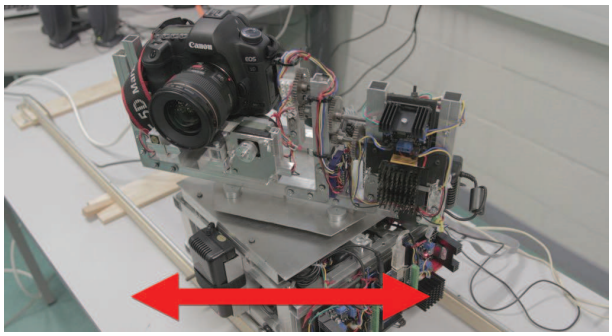


Figure 22: *Visualization of the camera rotation in the* $xy-$*plane.*

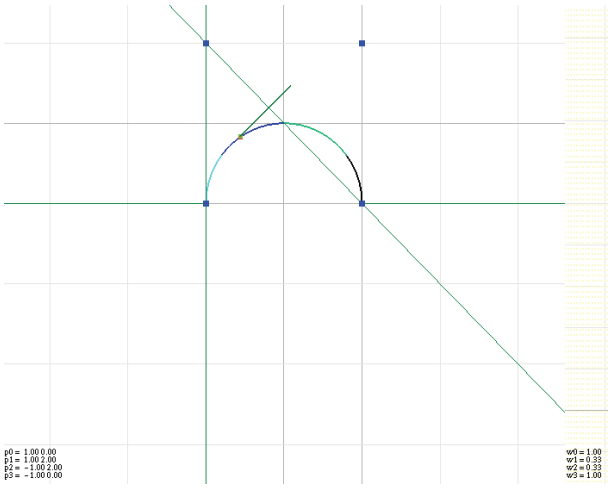


Figure 23: *Semicircle in the graphical interface.*

problems and possible improvements. So, it is essential to get feedback from users of the system.
As part of a project with the Art department of the University of Valenciennes, we asked a class of 14 students of the "Art plastiques et Création numérique" Master to test our system. In particular they tested the different steps, from motion capture through the haptic device to importing the data in the motion control software. They didn't directly move the robot and the camera, but they visualized in 3D Studio Max the different movements ("haptic", "ideal" and "intermediate"). The recorded movements are actually used in the realisation of a stop motion video clip. Before the system analysis, we gave a general presentation on the whole system, to explain why we decided to create it and how it works. After that, every student had the possibility to use the haptic interface to determine both the robot's translation and rotations and visualize his/her results in 3D

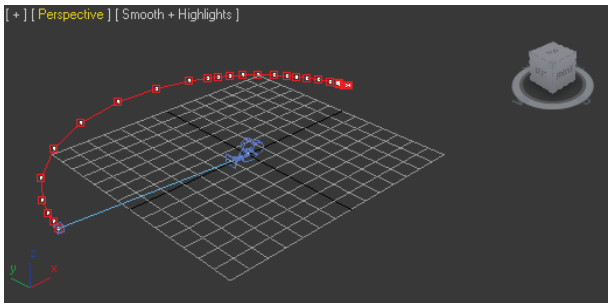


Figure 24: *Haptic $xy-$rotation parametrisation $\widetilde{\boldsymbol{p}}_i^{H_{r2}}$ visualized in 3D Studio Max.*

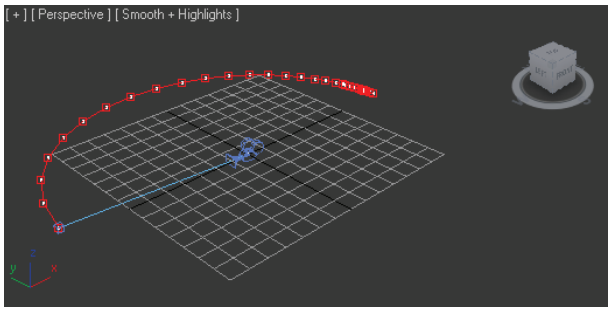


Figure 25: *Ideal $xy-$rotation parametrisation $\widetilde{\boldsymbol{p}}_i^{I_{r2}}$ visualized in 3D Studio Max.*

Studio Max.
We gathered as much information as possible about the system by having them fill out a questionnaire. We collected information about efficiency, ease of use, appropriateness of the whole system for translation and rotation movements.
In particular, four principal parts composed the questionnaire : "system's presentation", "motion capture movement (for translation and rotation)", "whole system" and "general questions". In the first part we asked if the system presentation was clear and if they needed more details. For all students the presentation was clear (for 7 students it was very clear and for the others clear), even if mathematics is not their domain. They only remarked that it is necessary to say that translation and rotation are separated.

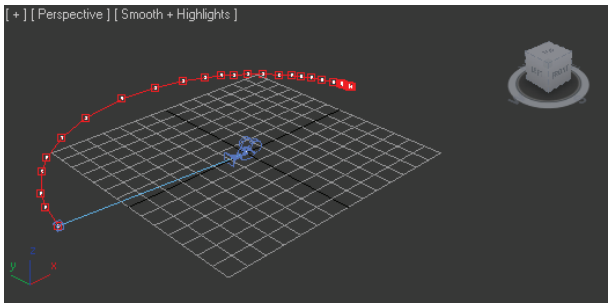


Figure 26: *Intermediate $xy-$rotation parametrisation $\widetilde{\boldsymbol{p}}_i^{B_{r2}}$ visualized in 3D Studio Max.*

In the second part we asked if both motion capture trajectory and rotation are intuitive (very intuitive, intuitive, moderately intuitive, not much intuitive, not intuitive). For all students the robot's translation is easy to do (for 3 students it is very intuitive, for 8 students intuitive and for 3 students moderately intuitive). They only suggested to improve the graphical interface visualization by adding a camera icon or a hand drawing, to be more immersed in the system. On the contrary, the camera rotations are a bit less intuitive (for 8 students it is moderately intuitive, for 4 students it is intuitive and for 2 students very intuitive). The students' feedback is that rotating camera movements are not so simple to execute correctly. Anyway, the rotation in the $yz-$plane (up/down camera rotation) is better than the rotation in the $xy-$plane (left/right camera rotation), because they have the sensation of holding in their hands a camera tripod rather than a camera. As a consequence, the resulting movement is inverted, because if you lift up the tripod, the "head" on which the camera is fixed goes down and vice versa. Indeed, the mathematical processing, in both cases, is very appropriate (for 7 students the process is very efficient and for the others it is efficient). They noticed that the resulting movements match the realistic haptic ones.
The third part is about the evaluation of the whole system. We asked if the system is efficient and what are the weaknesses of the system and what they would like to improve. The outcome of the evaluation shows that the whole system is efficient. For 10 students it is efficient, for 2 moderately efficient and for the remaining 2 very efficient. It works well and allows to easily reproduce realistic camera movements. The result doesn't look like a computer generated move, because the haptic device allows to add noise in a natural manner. On the other hand, half of the students performed the desired movement with some difficulties. The main reason for that is that they lack practical experience with the haptic system. They said that with more practice and several tests on the use of the haptic device, they would be able to improve their results. The main technical and ergonomic problem is the scale representation of the trajectory (the real is 6 meters and the virtual 6 centimeters). Thus, controlling and managing the speed along the curve is complicated. They suggested to amplify the translation length and the rotation radius. They also noted that there are too many steps before visualizing the movement in 3D Studio Max.
Another question was about the resulting movements from the mathematical processing. They found this process efficient (for 7 students it is very efficient and for the remaining 7 efficient). They remarked that the haptic movement needs to be modified, but that the elaborated movements

respect what they wanted to represent. In particular, they said that the most correct movement is the one created with $\alpha_i = \frac{||\mathbf{d}||}{d_i} \cdot 0.9 \ \forall i$, that is a movement near to the ideal one, as shown in Figure 20. Finally, we asked what they liked or not of the whole system and to add some remarks. Overall, all students found the new motion control system intuitive and innovative. It allows to simulate a realistic camera movement as in the world of cinematography and it overcomes the disadvantages of the 3D software animation. Moreover, it suits to a wide range of users, because it is simple to use and has an affordable price.

5. Conclusions

The analysis in section 4 brings us to the conclusion that our system can be proposed as a new method to help animators realize a realistic camera movement for stop motion animation. For our future work we would like to integrate all the steps of the system into a plug-in for 3D Studio Max and we would like to improve the motion capture rotation. We would like to create an open system in which a central part (Motion control software) synchronizes the other three (Camera movement management, Motion capture device and Motion control robot and digital camera), as summarized in Figure 27. We also want to experience the system within a professional production project.

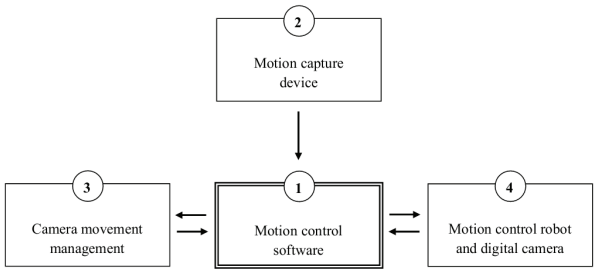


Figure 27: *Future Motion control system diagram.*

Références

[Der09] DERAKHSHANI D. : *Introducing Maya 2009*. SYBEX Inc., Alameda, CA, USA, 2009.

[Ebe08] EBERLY D. : Moving along a curve with specified speed. *Preprint* ((2008)). See http ://www.geometrictools.com.

[FTA10] FÜNFZIG C., THOMIN P., ALBRECHT G. : Haptic manipulation of rational parametric planar cubic using shape constraints. *SAC '10 Proceedings of the 2010 ACM Symposium on Applied Computing* (2010), 1253–1257.

[GP90] GUENTER B., PARENT R. : Computing the arc length of parametric curves. *IEEE Computer Graphichs and Applications. Vol. 10*, Num. 3 (mai (1990)), 72–78.

[GP04] GOVIL-PAI S. : *Principles of computer graphics : theory and practice using OpenGL and Maya*. Springer, (2004).

[KB84] KOCHANEK D. H. U., BARTELS R. H. : Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH Comput. Graph.. Vol. 18*, Num. 3 ((1984)), 33–41.

[Mur01] MURDOCK K. L. : *3D Studio Max 4 Bible*. Wiley Publishing, 2001.

[Par04] PARENT R. : *Computer Animation : Algorithms and Techniques*. Morgan Kaufmann Publishers Inc., (San Francisco, CA, USA, 2004).

[PT95] PIEGL L., TILLER W. : *The NURBS book*. Springer-Verlag, London, UK, 1995.

[SAL*11] SAINI L., ALBRECHT G., LISSARRAGUE N., ROMANI L., FUNFZIG C., BÉCAR J. : Animation 3d : mouvements de caméra réalistes pour la stop motion. *Actes de H2PTM, Metz* (Octobre 2011), 137–148.

[SALR10] SAINI L., ALBRECHT G., LISSARRAGUE N., ROMANI L. : Animation 3d : le problème de mouvement de caméra. *Actes de GTMG, Dijon* (2010), 69–76.

[ST82] SHARPE R., THORNE R. : Numerical method for extracting an arc length parameterization from parametric curves. *Computer Aided Design. Vol. 14*, Num. 2 (mars (1982)), 79–81.

[VB04] VERTH J. V., BISHOP L. : *Essential Mathematics for Games and Interactive Applications : A Programmer's Guide*. Morgan Kaufmann Publishers Inc., (San Francisco, CA, USA, 2004).

[WKA02] WANG H., KEARNEY J., ATKINSON K. : Arc-length parameterized spline curves for real-time simulation. In *in Proc. 5th International Conference on Curves and Surfaces* ((2002)), pp. 387–396.

Réalité virtuelle et manipulation de surfaces B-splines

M. Daniel, C. Guyot, S. Mavromatis, A. Polette

LSIS, Aix-Marseille Université, campus de Luminy, case postale 925, 13288 Marseille cedex 9

Résumé

La manipulation d'objets 3D est actuellement contrainte par les outils d'entrée-sortie (écran et souris ou tablette), par essence 2D. Il y a donc un besoin réel de pouvoir immerger le concepteur dans son modèle pour lui permettre une véritable interaction 3D. Nous proposons dans cet article un exemple de manipulation de surfaces B-splines en immersion 3D à l'aide d'outils légers de réalité virtuelle.

Manipulating 3D objects is constrained by input and output devices (display and mouse or graphics tablet), which are 2D devices. There is a real need for a designer to be embedded into his model, providing him a 3D interaction. We propose in this paper an example of B-splines manipulation with a 3D immersion obtained with simple RV tools.

Mots-clés : Modélisation géométrique, surfaces, B-splines, interaction, réalité virtuelle…

1. Introduction

La modélisation de surfaces B-splines ou NURBS est arrivée à un degré important de maturité du point de vue des outils mathématiques même s'il reste des difficultés comme par exemple la modélisation en respectant des courbures, des problèmes de sur-contraintes de raccordement ou des problèmes d'intersection. Par contre, les outils mis à la disposition des utilisateurs restent largement en deçà des capacités de modélisation. On peut analyser ce décalage avec deux points de vue.

Le premier est l'absence d'outils de haut niveau permettant à l'utilisateur de s'affranchir des contraintes mathématiques en lui permettant de travailler dans son langage métier ou avec ses contraintes métiers. Il y a évidemment des travaux réalisés dans ce sens [Gre05, VC07] ou plus récemment [Per11].

L'autre point de vue est intrinsèque aux postes de travail utilisés. Les outils de visualisation (écran) et les organes d'entrée (souris ou tablette graphique) sont 2D pour manipuler des objets 3D. Donc la visualisation se fait avec une inévitable projection réductrice et la saisie ne peut se faire qu'à travers des artifices de manipulation dans des plans.

Il apparaît donc clairement qu'immerger l'utilisateur dans son modèle 3D pour lui permettre de le manipuler directement en 3 dimensions est une approche prometteuse pour ne pas dire inéluctable. Plusieurs expériences ont déjà été tentées. On peut citer [AL01] qui utilise un système immersif lourd (CAVE : Cave automatic virtual environment) [CAV], [FB02] qui utilise un outil de désignation 3D, sorte de stylet monté sur un ensemble d'articulations, ou [PV02] qui utilise en même temps une télécommande dans la main pour désigner les fonctions à entreprendre. Curieusement, peu de travaux ont été, à notre connaissance, proposés récemmment.

Nous pensons que pour être efficace, la manipulation doit se faire de façon la plus naturelle possible et avec des moyens légers en terme d'équipement et donc de coût. C'est pour cela que nous avons développé une première expérimentation où l'utilisateur muni d'un casque stéréo et d'un gant va manipuler la surface en attrapant les points de contrôle avec sa main et en les déplaçant directement. Le déplacement global de la surface est envisagé de la même façon. L'utilisateur peut aussi tourner autour de son modèle et s'en rapprocher ou s'en éloigner. Nous décrirons dans le paragraphe 2 le matériel utilisé, puis dans le paragraphe 3 le réalisation technique du dialogue. Des résultats seront proposés dans le paragraphe 4 et nous tirerons les enseignements de cette première expérience dans le paragraphe 5. Ce travail a été réalisé avec des étudiants du master 2 professionnel SIS.

2. Les moyens matériels

L'atelier de réalité virtuelle qui a été utlisé est composé (figure 1) :

- d'une station graphique équipée d'un processeur Intel Xeon X5650, 8 Go RAM et d'une carte graphique NVIDIA Quadro 5000 2.5GB
- d'un casque Trivisio | ARVision-3D
- de deux capteurs magnétiques 6DOF PATRIOT (6 degrés de liberté par capteur)
- d'un gant de données 5DT à 14 degrés de liberté

Figure 1: *A gauche, le gant de données, à droite le casque. Les carrés gris (le gros et les deux petits) composent le système de capteurs 6 DOF.*

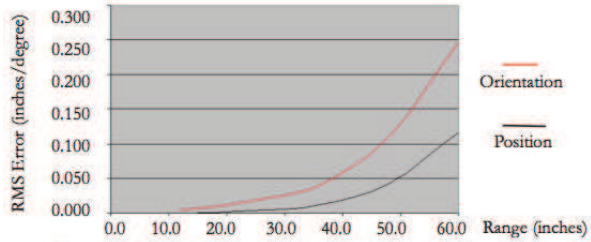
Le casque Trivisio | ARVision-3D est muni de deux caméras et de deux écrans permettant d'imaginer des applications de réalité augmentée. Dans notre expérimentation, nous avons utilisé uniquement les écrans, sans utiliser les deux caméras. La résolution de chaque écran est de 800x600 pixels. La longueur du cable est de 1,20 mètres. Le système de tracking magnétique "Polhemus Patriot" filaire est composé d'une base (le gros parallélépipède gris de la figure 1) permettant le calibrage des capteurs et de deux capteurs magnétiques à 6 degrés de liberté. La portée du système est de 1,5 mètres. La précision des capteurs varie en fonction de l'éloignement à la base comme le montre la figure 2. Le coût de cet atelier est d'environ 20 000 euro.

Le développement a été réalisé en C++ avec la bibliothèque graphique OpenGL et les SDK fournis avec chaque matériel. La programmation nécessite une réelle attention mais ne présente pas de difficulté particulière.

3. Modèle d'interaction

3.1. B-splines

Le modèle mathématique sous-jacent n'est pas le plus important dans cette démarche. Les deux modèles principalement utilisés pour la modélisation de surfaces à pôles sont actuellement les B-splines et les NURBS. Le lecteur souhaitant quelques précisions sur ces modèles peut se référer par



Range (inches)	Position Resolution (inches)	Orientation Resolution (degrees)
12.0	0.00046	0.0038
24.0	0.0035	0.0168
36.0	0.0113	0.0407
48.0	0.0428	0.1108
60.0	0.1175	0.2470

Figure 2: *Eloignement vs. résolution du Polhemus Patriot*

exemple à [Dan07]. Dans les deux cas, il y a évidemment des paramètres importants à fixer : les ordres des fonctions de base, les vecteurs de noeuds et pour les NURBS les poids. Si en règle générale les ordres et la paramétrisation sont fixés en préalable, les poids, quand ils sont modifiés en cours de modélisation, sont des paramètres d'ajustement (peu intuitifs d'ailleurs). Il s'agit cette fois de valeurs numériques à fixer et nous avons dans un premier temps exclu cette possibilité. Nous manipulons donc des B-splines d'ordres et paramétrisation fixés au départ.

3.2. L'environnement virtuel

Nous visualisations la surface et son réseau de points de contrôle. Les objets avec lesquels l'utilisateur peut interagir sont représentés par des boules. La figure 3 représente en vue plane ce que voit l'utilisateur. Il s'agit soit des points de contrôle qui peuvent être déplacés (boules blanches), soit de "poignées" qui permettent de réaliser des rotations (les 3 boules de couleur périphériques) et translations de la surface (boule jaune centrale).

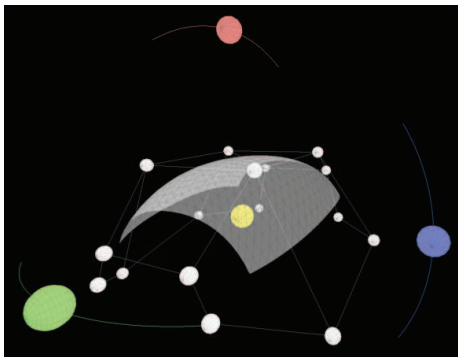


Figure 3: *L'environnement virtuel*

3.3. Vision

Afin de favoriser l'immersion de l'utilisateur dans cet environnement, nous proposons de visualiser la surface en vision stéréoscopique. L'affichage passant par la carte vidéo de l'ordinateur, il n'y a pas de logiciel fourni avec le casque, donc indépendant de l'ordinateur, pour réaliser directement cet affichage. De plus, il existe différentes solutions chacune ayant des avantages et des inconvénients. Nous avons choisi de développer une solution, indépendante de la carte graphique, décrite ci-dessous. Techniquement il faut produire deux images sensiblement différentes (une par oeil). Ces deux images représentent deux perspectives que nos yeux "voient" naturellement en vision binoculaire. Pour cela, nous avons choisi de dessiner la scène deux fois, en effectuant une translation de notre point d'observation entre les deux affichages. Nous allons donc utiliser un point d'observation temporaire, qui correspond au point d'observation "non-stéréoscopique" de notre scène, dans le monde réel (positionné entre nos deux yeux). Le décalage est effectué après le positionnement et l'orientation du point d'observation temporaire, nous effectuons une translation de ce point, sur un axe perpendiculaire à la droite passant par le point d'observation temporaire et par le point observé. Nous effectuons alors une translation de part et d'autre de ce point d'une valeur constante égale à la moitié de la distance entre nos deux yeux, afin de créer deux nouveaux points d'observation. Nous utiliserons ces deux nouveaux points d'observation pour réaliser nos deux affichages correspondant aux deux yeux de l'utilisateur.

Afin de réaliser le rendu 3D en fonction du point de vue de l'utilisateur, nous avons fixé un capteur magnétique sur le casque. L'ajout de ce capteur permet avec ses 6 degrés de liberté de tourner autour du modèle et d'éloigner ou rapprocher le point de vue. Ce type de visualisation donne une bonne appréhension des profondeurs et permet d'envisager une utilisation naturelle du couple "gant/capteur" pour la partie interaction. Bien évidemment, le casque est tout de même contraignant et le fait qu'il ne soit pas wifi limite fortement les déplacements autour du modèle. Nous nous sommes essentiellement attachés à la sélection et manipulation des points de contrôle et de la surface. L'utilisation du casque pose naturellement le problème de la mise en place d'une interface utilisateur totalement immersive pour permettre d'interagir avec l'application et dépasse le cadre de cette étude.

Nous avons également remplacé le dispositif casque/capteur par de simples lunettes cartons à filtres de couleur (lunettes pour anaglyphes). Le principe est de placer deux filtres de couleurs différentes devant chaque oeil (dans notre cas un rouge et un cyan), et d'afficher deux images simultanément en filtrant au préalable ces images afin de choisir ce que chaque oeil doit voir, pour recréer la perception du relief. La production des deux images est basée sur le même principe que pour la vision stéréoscopique décrite plus haut. Avant d'être affichées, ces images sont en plus filtrées : nous supprimons une partie des composantes des couleurs de chaque image de façon à ne conserver que les composantes correspondantes aux filtres des lunettes (cyan et rouge), ce qui aura pour effet de masquer des éléments différents à travers chacun des filtres. L'objectif était de tester un dispositif très léger même si la "qualité" du rendu stéréoscopique était moins bonne (luminosité plus faible, images fantômes, …). Nous notons que les moyens mis en place pour l'interaction avec l'environnement virtuel (voir le paragraphe 3.4) restent valides.

3.4. Sélection et manipulation

Pour interagir avec l'environnement, nous avons choisi d'utiliser un gant de données associé à un capteur 6DOF accroché à "l'intérieur" de la main. Le gant va nous permettre de reconnaître les mouvements des doigts (indépendamment de l'orientation), tels que fermer ou ouvrir la main, la capteur additionnel donne le positionnement de la main dans l'espace, donc dans la scène. La position du capteur dans le gant est très importante pour une saisie naturelle. Positionner celui-ci dans le creux de la main évite la sensation désagréable de désignation "à côté" obtenue si le capteur est positionné au niveau du poignet. La position de la main est indiquée dans la scène par une sphère faiblement lumineuse (cerclée d'un pointillé rouge sur la figure 4 pour la rendre visible dans cet article).

La tâche de sélection est décomposée en deux sous-tâches : la désignation d'un objet et la validation de cette sélection. La désignation consiste à déplacer la main jusqu'à "croiser" un point de contrôle ou une "poignée. La validation consiste à fermer la main et à la garder fermée. L'objet désigné change alors de couleur et devient immédiatement caractéristique dans la scène. Le fait d'ouvrir la main invalide la sélection. La tâche de manipulation est implémenté en agissant directement sur l'objet. Ainsi après avoir sélectionné un objet, l'utilisateur peut le déplacer en déplacement sa main. Le fait d'ouvrir la main invalidant la sélection arrête la manipulation. La surface visualisée suit immédiatement les déformations de son polyèdre de contrôle.
Le déplacement de la scène complète (translations et rotations) s'effectue de la même façon avec les quatre poignées définies précédemment, en complément de la capacité qu'a l'utilisateur de réaliser les mêmes actions par les mouvements de sa tête grâce au capteur installé sur le casque stéréo. Il nous faut noter que seuls les trois degrés de liberté associés à la translation sont utilisés pour le capteur placé dans la main. Il serait donc possible d'utiliser les trois angles pour gérer les rotations de la scène en tournant la main. Cette extension n'a pas encore été testée.

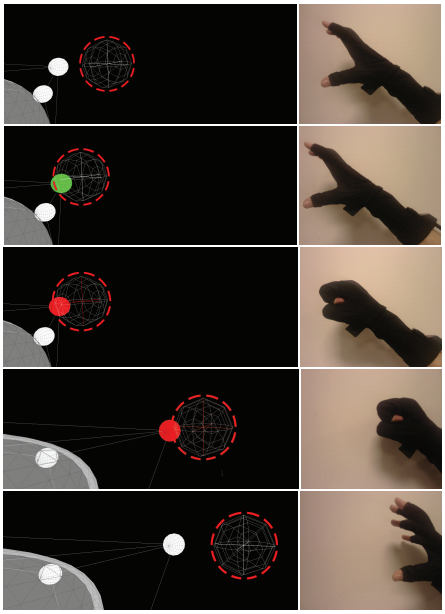


Figure 4: *Sélection et déplacement d'un point de contrôle*

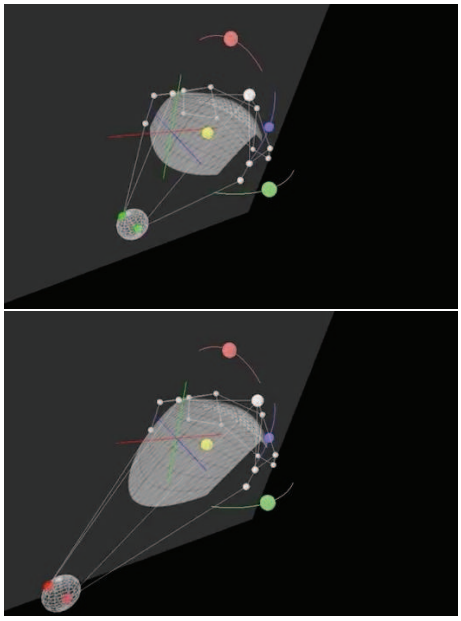


Figure 5: *La surface est déformée en "tirant" deux points de contrôle*

4. Résultats

Dans l'état actuel de nos développements, la "sensation" d'interaction est tout à fait bien reproduite, l'utilisateur s'appropriant immédiatement le dispositif. Un exemple complet de manipulation d'un point de contrôle est proposé sur la figure 4, en montrant simultanément à gauche une vue plane de la scène et à droite les actions de la main de l'utilisateur. De haut en bas, l'utilisateur s'approche d'un point (image 1), puis le point est détecté dans la main (image 2). Le fait de fermer la main sélectionne le point (image 3), le point est tiré vers sa nouvelle position (image 4) et est finalement désélectionné (image 4), validant ainsi la nouvelle position. On peut constater sur la figure 5 que la sélection de plusieurs points simultanément est possible, bien que cette possibilité de permette pas de définir réellement des opérateurs de haut-niveau. La figure 6 montre l'utilisateur dans son environnement de travail.
Nous sommes conscients qu'il existe différentes techniques de sélection et de manipulation et que nous avons simplement choisi ce qui nous semblait le plus naturel pour cette première expérience. Evidemment, les tests ont été réalisés sur des surfaces ayant un nombre réduit de points de contrôle et le passage à des surfaces "industrielles" possédant beaucoup plus de points de contrôle donc a priori de plus grandes dimensions nécessitera quelques aménagements. En particulier, avec cette première approche, il est compliqué d'attraper des objets "lointains". Une technique Go-Go ([IP96]) ou une variante pourrait être intéressante. Avec ce type de technique, la main réelle est représentée par une main virtuelle dans l'environnement. La position de la main virtuelle est estimée par une fonction non-linéaire, de façon à ce que la main virtuelle aille plus loin que la main réelle après une certaine distance. L'utilisateur dispose donc d'un bras virtuel plus long que son bras réel, ce qui lui permet d'atteindre des objets "lointains".

Comme dans toute application de réalité virtuelle, tout périphérique "lourd" est contraignant. Ainsi dans notre approche, le casque est la plus grosse contrainte, accentuée par le fait que celui-ci n'est pas wifi. Nous avons donc fait un essai avec des lunettes bicolores en carton qui donne sur l'écran le rendu 3D de facon plus légère sur le même principe que la télévision 3D mais ne permet plus de s'éloigner ou se rapprocher et de tourner autour du modèle.

5. Conclusion

L'expérience décrite ici permet de valider l'approche proposée pour manipuler une surface B-splines ou NURBS. Le matériel utilisé est de faible coût et il est donc facile de reproduire cette expérience. La phase de test doit être prolongée pour mieux analyser les limites du concept qui parait pourtant naturel. Il semble difficile de manipuler simplement dans ce cadre une surface définie par un grand nombre de points de contrôle (plusieurs centaines), ni de modifier de façon très fine la surface. L'outil semble donc plus adapté à l'obtention efficace d'ébauche de modèles (dans le cas de la CAO). Le matériel lui même reste contraignant pour l'utili-

Figure 6: *Manipulation en cours*

sateur, en particulier le casque stéréo alors que le gant se fait oublier. L'utilisation d'un deuxième gant rendrait peut-être l'approche plus naturelle ou du moins plus agréable pour un gaucher ! Les 14 degrés de liberté disponible par gant, permettent d'envisager des combinaisons complexes et donc différents opérateurs de manipulation. Il reste aussi à étudier comment l'utilisateur peut interagir avec l'application, comme par exemple changer le degré des B-splines, les vecteurs de noeuds, le poids des NURBS, ... sans passer par la souris ou le clavier, passage rendu d'ailleurs problématique par le port du casque stéréo. Pour la modification des poids, on pourrait envisager d'utiliser les nombreux degrés de liberté disponibles sur le gant, et donc la prise en compte d'une gestuelle particulière sur un point désigné qui ferait graduellement varier son poids. Pour les vecteurs de noeuds ou les ordres des splines, l'interaction semble moins naturelle.
il est facile de prédire que ces matériels de réalité virtuelle vont être de plus en plus perfectionnés et permettront donc le développement d'applications toujours plus naturelles pour un coût de plus en plus faible.

Références

[AL01] A. LIVERANI G. P. : Full-scale surface modeling in virtual reality. *12e Conférence Internationale on Design, Tools and Methods in Industrial Engineering* (septembre 2001), E1–17–E1–25.

[CAV] http ://en.wikipedia.org/wiki/cave_automatic_virtual_environment.

[Dan07] DANIEL M. : Courbes et surfaces paramétriques. In *Informatique Graphique, modélisation géométrique et animation, Dominique Bechmann et Bernard Péroche (eds), Lavoisier-Hermès* (2007), pp. 95–134. ISBN : 978-2-7462-1514-6.

[FB02] F. BRUNO M.L. LUCHI M. M. S. R. : A virtual reality desktop configuration for free-form surface sketching. *14e congrès Ingénierie Graphique* (juin 2002), 1–10.

[Gre05] GRECA R. L. : *Approche déclarative de la modélisation de surfaces*. PhD thesis, Université de la Méditerranée, octobre 2005.

[IP96] I. POUPYREV S. WEGHORST M. B. T. I. : The go-go interaction technique non-linear mapping for direct manipulation in vr. *ACM Symposium on User Interface Software and Technology (UIST Õ96)* (1996), 79–80.

[Per11] PERNOT J. : *Vers de nouveaux paradigmes et outils de génération et traitement de maquettes numériques tournés vers la sémantique*. PhD thesis, Habilitation à Diriger des Recherches, Grenoble INP, décembre 2011.

[PV02] PERLES B. P., VANCE J. M. : Interactive virtual tools for manipulating nurbs surfaces in a virtual environment. *Journal of Mechanical Design. Vol. 124* (juin 2002), 158–163.

[VC07] V. CHEUTET M. DANIEL S. H. R. L. G. J.-C. L. R. M. D. M. B. S. : Constraint modeling for curves and surfaces in cagd : A survey. *International Journal of Shape Modeling. Vol. 13*, Num. 2 (décembre 2007), 159–199.

Cartes Combinatoires *d*D dans CGAL

Guillaume Damiand

Université de Lyon, CNRS, LIRIS, UMR5205, F-69622 France

Résumé

CGAL est une importante bibliothèque C++ libre fournissant de nombreux outils et algorithmes de géométrie algorithmique. Elle propose entre autre des structures de données permettant de représenter des objets triangulés ou non. Pour les objets triangulés, deux structures de données spécifiques existent pour représenter des surfaces composées de triangles et pour représenter des volumes composés de tétraèdres. De plus une généralisation dD est en cours de développement. Pour les objets quelconques, CGAL proposait jusqu'alors uniquement une implantation de la structure de demi-arêtes qui est seulement 2D (`Halfedge Data Stuctures`) et une sur-couche géométrique permettant de décrire des surfaces 2D plongées dans $\mathbb{R}^3$ (`Polyhedron_3`). Afin de pouvoir représenter des objets dD quelconques, nous avons développé un module de cartes combinatoires permettant de représenter des quasi-variétés en dimension quelconque (`Combinatorial Maps`), ainsi qu'une sur-couche géométrique permettant de plonger ces quasi-variétés dans $\mathbb{R}^{d2}$ (`Linear Cell Complex`). Dans cet article, après une rapide présentation des cartes combinatoires, nous présentons ces deux modules en détaillant les classes et les principales opérations existantes.

Mots-clés : Modèles combinatoires et topologiques, Bibliothèque C++, Modélisation géométrique

1. Introduction

En modélisation géométrique, les deux grandes classes de méthodes de représentation des objets volumiques sont la CSG (*constructive solid geometry* ou modélisation volumique) et la B-Rep (*Boundary representation* ou modèle de représentation par bord) [BP07]. En CSG, tout objet est construit à partir d'un ensemble d'objets primitifs par application d'opérations booléennes. De ce fait, les modèles B-Rep sont beaucoup plus flexibles car ils autorisent plus d'opérations. Pour cette raison, ces modèles sont utilisés dans plusieurs modeleurs géométriques professionnel (nous pouvons citer par exemple Acis [Aci] ou Parasolid [Par]).

Parmi les modèles B-rep, une des structures de données les plus utilisées est la structure des demi-arêtes (en anglais *half-edges* data structure ou HDS) [Wei88] qui offre un excellent compromis en terme de taille de la structure et de temps d'accès aux information. Pour cette raison, plusieurs implantation des HDS existent (par exemple OpenMesh [Ope], ou dans CGAL [Ket12b, Ket12a]). Mais les HDS sont définies uniquement en 2D et permettent donc uniquement de représenter des objets subdivisés en sommets, arêtes et faces.

Pour lever cette restriction, les HDS ont été étendues en dimension 3 au travers différentes solutions (par exemple les pavages [AK89] ou les facet-edge data structure [DL87]) puis généralisées en dimension quelconque dans la notion de carte combinatoires [Lie91] étendue par la suite afin de pouvoir représenter des objets à bords [PABL07, Dam10].

L'avantage principal des cartes combinatoire est leur généricité en dimension quelconque qui autorise leur utilisation au sein de différentes applications de traitement d'images par exemple en 2D et 3D ou de modélisation géométrique 3D ou 4D pour de l'animation. Nous pouvons citer comme illustration les travaux de thèse de Sébastien Horna qui utilise des cartes en modélisation géométrique afin de proposer une méthode de reconstruction semi-automatique de complexes architecturaux à partir de plans numériques pour définir un modeleur de bâtiment [Hor08], ou les travaux de thèse d'Alexandre Dupas en traitement d'images qui utilise des cartes pour proposer différents algorithmes de segmentation topologique d'images 3D [Dup09] Un autre avantage des cartes combinatoires est que de nombreuses opérations existent et permettent la manipulation et la modification des objets représentés à l'aide de cartes.

Pour tous ces avantages, plusieurs implantations des cartes ont été réalisées. Nous pouvons citer la librairie `Cgogn` de l'équipe IGG du LSIIT à Strasbourg [Cgo] qui propose différentes variantes autour des cartes combinatoires 2D et 3D et le modeleur géométrique 3D `Moka` de l'équipe IG du XLIM-SIC à Poitiers et M2DisCo du LIRIS à Lyon [Mok] qui utilise un noyau de carte généralisée 3D. De plus, les cartes combinatoires ont été également utilisées en traitement d'images 2D et 3D et différentes librairies ont été développées dans ce cadre spécialisé. La librairie `Girl` de l'équipe Image & Son du Labri à Bordeaux [Gir] utilise un noyau de carte combinatoire 2D, et la librairie [Cpy] de l'équipe Image du Greyc propose un noyau de pyramide de carte combinatoire 2D.

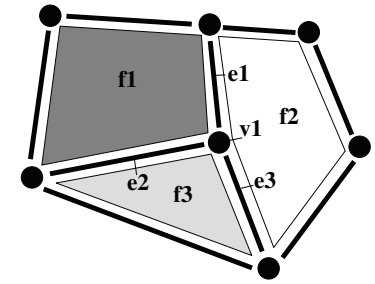
Tous ces développements ont différents avantages et inconvénients, mais à notre connaissance il n'existait pas jusqu'alors de librairie générique implantant les cartes combinatoires en dimension quelconque. C'est pour lever cette limitation que nous avons développé le module `Combinatorial map` [Dam12a] dans CGAL [Cga]. L'objectif de ce module est de pouvoir factoriser tout les développements existant autour des cartes combinatoires en proposant une solution pérenne garantie par l'intégration dans CGAL qui est une librairie importante et reconnue de géométrie algorithmique. Dans le cadre de la modélisation géométrique, il est souvent nécessaire de pouvoir représenter également la forme des objets représentés et nous avons pour cela proposé une implantation des complexes cellulaires linéaires dans lesquels un point de l'espace R^{d2} est associé à chaque sommet de la carte (module `Linear cell complex` [Dam12b]). Cette sur-couche propose quelques opérations géométrique et pourra être enrichie par de très nombreuses opérations de modélisation géométrique.

Le plan de cet article est le suivant. Nous commençons Sect. 2 par faire une rapide présentation des cartes combinatoires et des complexes cellulaires linéaires. Puis Sect. 3, nous présentons comment ces notions ont été implantées dans CGAL, et quelles opérations ont été définies. Enfin la Sect. 4 conclut cet article et donne des perspectives.

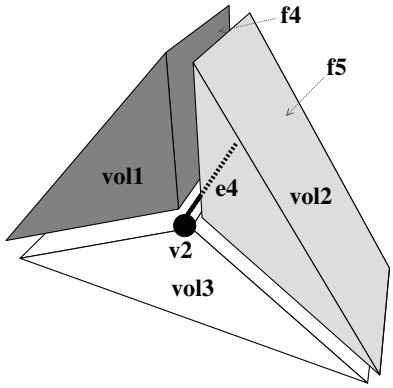
2. Rappels Autour des Cartes Combinatoires

L'objectif des modèles cellulaires est de représenter des *objets d*D orientables subdivisés en *cellules*. Sur l'exemple de la Fig. 1(a), l'objet 2D est subdivisé en sommets (cellules de dimension 0), arêtes (cellules de dimension 1) et faces (cellules de dimension 2), tandis que l'objet 3D de la Fig. 1(b) est également subdivisé en volumes (cellules de dimension 3).

Représenter ces cellules est important car elles sont le support naturel permettant d'associer des informations à certaines parties des objets. Par exemple étant donné un objet 3D subdivisé, nous pouvons associer des coordonnées 3D à chaque sommet, des couleurs (un triplet de nombres entre 0 et 255) à chaque volume…



(a)



(b)

Figure 1: *Exemples d'objets subdivisés qui peuvent être représentés par des cartes combinatoires. (a) Un objet 2D composé de 3 faces (2-cellules)* f1*,* f2 *et* f3*, neuf arêtes (1-cellules) et sept sommets (0-cellules). (b) Un objet 3D (représenté partiellement pour les sommets et les arêtes) composé de trois volumes (3-cellules)* vol1*,* vol2 *et* vol3*, douze faces (2-cellules) (une seule face,* f4*, sépare les volumes* vol1 *et* vol2*, et de manière similaire une seule face sépare* vol1 *et* vol3*, et* vol2 *et* vol3*), seize arêtes (1-cellules), et huit sommets (0-cellules).*

En plus des cellules, il est nécessaire de représenter les différentes relations entre ces cellules. Ces relations d'incidence et d'adjacence représentent la structure de la subdivision. La relation d'incidence concerne deux cellules de dimension différentes. Un sommet est incident à une arête s'il est un de ses deux bords ; une arête est incidente à une face si elle décrit une partie du bord de la face ; une face est incidente à un volume si elle décrit une partie du bord du volume… La relation d'adjacence concerne deux cellules de même dimension. Deux arêtes sont adjacentes si elles partagent un même sommet ; deux faces sont adjacentes si elles partagent une même arête ; deux volumes sont adjacents s'ils partagent une même face…

De manière générale, nous parlons de *i*-cellule pour une cellule de dimension *i*, et un objet *d*D est subdivisé en cellules de dimension 0 jusqu'à des cellules de dimension *d*.

Deux cellules sont incidente si l'une est dans le bord de l'autre, et deux i-cellules sont adjacentes si elles partagent une même $(i-1)$-cellule incidente.

Sur l'exemple de la Fig. 1(a), l'objet 2D est composé de 3 faces (2-cellules), neuf arêtes (1-cellules) et sept sommets (0-cellules). *e1* est incidente à *f1* et à *f2*, et donc *f1* et *f2* sont adjacentes le long de l'arête *e1*. Le sommet *v1* est incident à l'arête *e1*, donc *v1* est incident à *f1* et à *f2* par transitivité. Dans la Fig. 1(b), l'objet 3D est composé de trois volumes (3-cellules), douze faces (2-cellules), seize arêtes (1-cellules), et huit sommets (0-cellules). *vol1* et *vol2* sont adjacent le long de la face *f4*, donc *f4* est incidente à *vol1* et à *vol2*. l'arête *e4* est incidente aux trois faces entre *vol1* et *vol2*, *vol1* et *vol3*, et *vol2* et *vol3*. *e4* est donc incidente aux trois volumes par transitivité.

Il se pose alors la question de la définition d'une structure de données permettant de décrire la subdivision des objets en cellules, ainsi que les relations d'adjacence et d'incidence. Plusieurs structures de données spécifiques ont été proposées pour répondre à ce besoin dans les cas 2D et 3D, souvent inspirées par les graphes d'incidence [Ede87], mais il existe très peu de solutions génériques, c'est à dire valable en d-dimension, et ordonnées, c'est à dire représentant l'ordre des relations d'adjacence entre les cellules . À notre connaissance, les seules structures existantes sont les cartes combinatoires [Lie91] (dont différentes variantes plus ou moins équivalentes, comme par exemple les cell tuple [Bri93]).

Dans les cartes combinatoires, les objets sont représentés par leurs bords au moyen de *brins* et de relations reliant ces brins. Un brin peut-être vu comme un morceau d'arête orientée, ainsi qu'un morceau de chaque i-cellule incidente pour tout $i \in \{0,2,\ldots,d\}$. Nous pouvons voir dans la Fig. 2 les deux cartes combinatoires 2D et 3D représentant les deux objets subdivisés de la Fig. 1.

La carte combinatoire 2D de la Fig. 2(a) est composée de 12 brins représentés par des segments orientés, et de deux relations : β_1 donnant pour un brin le brin suivant de la même face (par exemple $\beta_1(1) = 2$), et β_2 donnant pour un brin l'autre brin de la même arête mais de la face opposée (par exemple $\beta_2(3) = 4$). La carte combinatoire 3D de la Fig. 2(b) est composée de 54 brins et de trois relations : β_1 et β_2 qui agissent comme précédemment, et β_3 donnant pour un brin l'autre brin de la même arête, de la même face mais du volume opposé.

Lorsqu'il n'existe pas de brin en relation avec un brin b pour un β_i donné, nous définissons $\beta_i(b) = \varnothing$ et disons que b est i-libre. Ces cas arrivent lorsque les objets représentés sont des objets à bords. C'est par exemple le cas dans la carte 2D de la Fig. 2(a) pour le β_2 des brins 1, 5, 6, 10, 11 et 12 (et nous avons donc par exemple $\beta_2(1) = \varnothing$).

Ces premiers exemples de cartes combinatoires illustrent leur principal intérêt pour la définition en dimension quelconque. En effet, augmenter la dimension se fait simplement

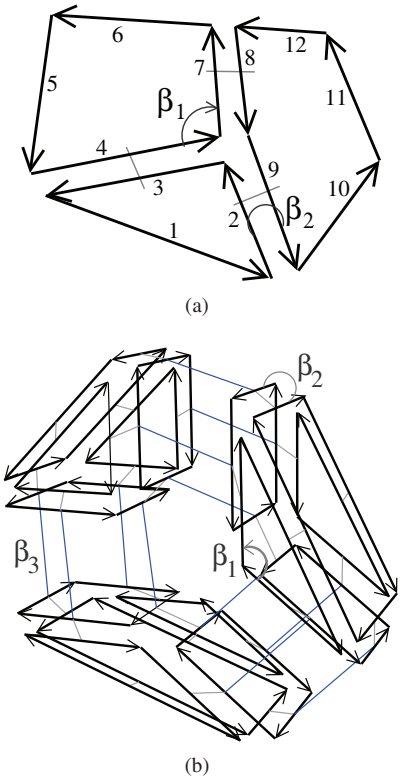


Figure 2: *Cartes combinatoires décrivant les deux objets de la Fig. 1. (a) La carte combinatoire 2D contenant 12 brins. (b) La carte combinatoire 3D contenant 54 brins.*

en ajoutant une nouvelle relation β_d. Cela nous amène à la définition 1 des cartes combinatoires dD.

Définition 1 (Carte combinatoire) Soit $d \geq 0$. Une d-carte combinatoire, (ou d-carte) est une algèbre $C = (B, \beta_1, \ldots, \beta_d)$ où :

1. B est un ensemble fini de brins ;
2. β_1 est une *permutation partielle* sur B ;
3. $\forall i : 2 \leq i \leq d : \beta_i$ est une *involution partielle* sur B ;
4. $\forall i, j : 1 \leq i < i+2 \leq j \leq d : \beta_i \circ \beta_j$ est une *involution partielle*.

β_1 est une *permutation partielle*, c'est à dire une bijection de $B \cup \{\varnothing\}$ dans $B \cup \{\varnothing\}$. Les autres β_i sont des *involutions partielles*, c'est à dire une bijection de $B \cup \{\varnothing\}$ dans $B \cup \{\varnothing\}$ tel que si $\beta_i(b) \neq \varnothing$, alors $\beta_i(\beta_i(b)) = b$. Cela s'explique car pour tout brin b n'appartenant pas à un bord, $\beta_i(b)$ donne l'autre brin appartenant aux mêmes j-cellules pour tout $j \in \{1,\ldots,i-1,i+1,\ldots,d\}$ et à la i-cellule opposée. Il existe exactement un seul brin b' satisfaisant cette propriété, et pour b', le brin satisfaisant cette propriété est b : nous avons donc $\beta_i(b) = b'$ et $\beta_i(b') = b$.

La dernière ligne de la définition 1 fixe les contraintes de

cohérence que doivent vérifier les différentes applications afin que les objets représentés soient “valides”. Intuitivement, cette condition garantit que les objets soient des quasi-variétés, qui peuvent être vues comme l'analogue combinatoire des variétés topologiques.

Nous avons déjà indiqué qu'un brin représente un morceau d'arête orientée, ainsi qu'un morceau de chaque i-cellule incidente. Pour cette raison, chaque cellule est représentée de manière implicite dans une carte combinatoire par un ensemble de brins. Ces ensembles de brins sont obtenus par la notion d'orbite. Intuitivement, l'orbite d'un brin b pour les permutations f_1, ..., f_k notée $\langle f_1, \ldots, f_k \rangle (b)$ est l'ensemble des brins qu'il est possible d'atteindre à partir de b en utilisant n'importe quelle suite de f_i et de f_i^{-1}.

Pour la 2-carte de la Fig. 2(a), le sommet v_1 est décrit par les brins $\{3,7,9\}$, l'arête e_1 par les brins $\{7,8\}$ et la face f_1 par les brins $\{4,5,6,7\}$.

L'intérêt de cette représentation implicite est qu'elle évite d'avoir plusieurs structures de données à maintenir et à mettre à jour en parallèle lors des opérations. Toute l'information structurelle est définie dans les brins et les applications. L'autre intérêt de cette représentation est que la définition d'adjacence et d'incidence entre cellules se transpose très simplement sur les brins. Deux cellules sont incidentes si les deux ensembles de brins correspondants ont une intersection non vide. Deux i-cellules sont adjacentes (pour $i > 0$) s'il existe un brin de la première b_1 et un brin de la seconde b_2 vérifiant $b_1 = \beta_i(b_2)$.

Une d-carte décrit donc la subdivision d'un objet dD orientable en cellules ainsi que toute les relations d'incidence et d'adjacence entre ces cellules. Mais dans la plupart des applications ces informations ne sont pas suffisante. En effet, il est souvent nécessaire d'associer des informations spécifiques à certaines cellules : c'est la notion *d'attributs*. Comme cette association est possible pour les cellules de toute dimension, nous parlons de i-attributs pour les attributs associés aux i-cellules.

Comme les cellules sont représentées de manière implicite dans les cartes combinatoires par des ensembles de brins, l'association d'un i-attribut a à une i-cellule c se fait au moyen des brins : chaque brin de c pointe vers l'attribut a. Dans le but d'être le plus générique possible, une i-cellule n'est pas obligatoirement associée à un i-attribut. Les brins des i-cellules sans i-attribut pointerons alors vers `NULL`.

Nous présentons Fig. 3 deux exemples de cartes combinatoires pour lesquelles nous avons ajouté des attributs. Pour la 2-carte de la Fig. 3(a), des 2-attributs associent des couleurs aux faces (2-cellules), et des 1-attributs associent une longueur aux arêtes (1-cellules). Nous pouvons observer que seules 3 arêtes sur les 7 possèdent des 1-attributs. Pour la 3-carte de la Fig. 3(b), trois 2-attributs sont associés à 3 faces (2-cellules) et contiennent une couleur.

Ces attributs permettent d'associer n'importe quelle information aux cellules d'une carte, et permettent ainsi d'associer de la géométrie en dimension $d2$ à toute carte combinatoire de dimension d : nous disons que la d-carte est plongée dans $\mathbb{R}^{d2}$ (en toute généralité d et $d2$ ne sont pas nécessairement liés). Différent types de plongement existent, et nous avons considéré pour le moment uniquement les plongements linéaires. Pour cela, un point de dimension $d2$ est associé à chaque sommet de la carte. Le plongement de chaque arête est le segment composé des deux points des extrémités de l'arête. Le plongement de chaque face est le polygone bordé par tout les segments des arêtes incidente à la face... Nous parlons de complexe cellulaire linéaire pour désigner une carte combinatoire plongée linéairement. La Fig. 4 montre un exemple de 2-carte et de 3-carte plongées qui sont donc des complexes cellulaires linéaires 2D et 3D.

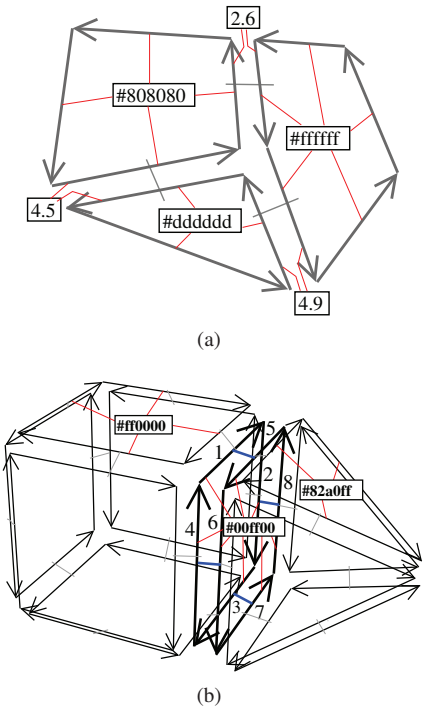


Figure 3: *Exemple d'association d'attributs dans une carte combinatoire. (a) Une 2-carte avec une couleur (un triplet hexadécimal) associée aux faces (2-attributs) et une longueur (un float) associée aux arêtes (1-attributs). (b) Une 3-carte avec une couleur (un triplet hexadécimal) associée aux faces (2-attribut).*

3. Présentation des Modules Développés

Nous avons implanté dans CGAL les cartes combinatoire dD intégrant la possibilité d'associer des attributs aux cellules, des itérateurs permettant de parcourir les cartes, et une sur-couche géométrique permettant de représenter les cartes

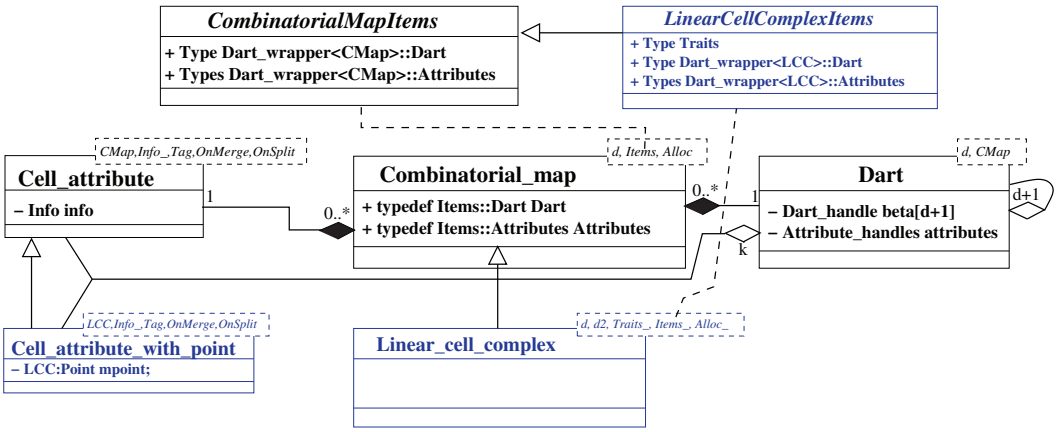


Figure 5: *Diagramme UML des principales classe des modules* `Combinatorial map` *en noir et* `Linear cell complex` *en bleu. k est le nombre d'attributs* non void.

combinatoires plongées dans un espace Euclidien. Comme c'est le cas pour la majorité des structures de données de CGAL, le niveau combinatoire est séparé du niveau géométrique dans deux modules distincts : `Combinatorial map` et `Linear cell complex`. Pour simplifier cet article, nous présentons principalement les fonctionnalités du module `Linear cell complex` qui englobe la majorité des fonctionnalités de la couche de base. Pour cette même raison, seules les principales caractéristiques des modules sont présentées. Le lecteur intéressé est invité à consulter les manuel utilisateur et développeur se trouvant sur le site de CGAL [Dam12a, Dam12b].

3.1. Structure des Modules

La Fig. 5 présente le diagramme UML des principales classes des modules `Combinatorial map` et `Linear cell complex`. Comme pour la plupart des structures de CGAL, les classes définies dans nos deux modules sont toutes paramétrables afin de laisser le choix aux utilisateurs de les spécialiser selon leurs besoins. Le principal mécanisme utilisé pour fixer ces paramètres est l'utilisation de classes d'items dans lesquelles sont définis les types de base utilisés.

Le type de classe d'items utilisé pour paramétrer la classe Linear_cell_complex doit respecter le schéma du modèle LinearCellComplexItems. Ce modèle définit tout d'abord le type de noyau CGAL utilisé au moyen du type Traits, le type de brin utilisé au moyen du type Dart, et tout les attributs actifs au moyen du k-uplet Attributes. Ce k-uplet contient au maximum $d+1$ type, un par i-attribut possible. Le premier élément du k-uplet donne le type des 0-attributs, le second élément donne le type des 1-attributs... Pour chaque type, il est possible d'utiliser **void** pour ne pas activer les i-attributs correspondant. Enfin, tout les i-attributs pour $i \geq k$ sont désactivés. Chaque type utilisé dans le k-uplet doit être **void** ou un modèle du concept CellAttribute (en utilisant par exemple la classe Cell_attribute), tandis que le type associé aux 0-attributs doit obligatoirement être un modèle du concept CellAttributeWithPoint (en utilisant par exemple la classe Cell_attribute_with_point) pour garantir que chaque sommet de la carte soit au moins associé avec un point géométrique. Il existe une classe d'items par défaut permettant d'utiliser directement un complexe cellulaire linéaire dD plongé dans $\mathbb{R}^{d2}$ et ayant tous les attributs inactifs, à l'exception des 0-attributs.

La classe Dart<d,CMap> propose une implantation standard des brins. Elle contient un tableau de pointeurs pour chaque β_i, pour $i \in \{0,\ldots,d\}$, ainsi que des pointeurs vers chaque attribut non void. Cette classe est paramétrée par d la dimension de la carte, et par CMap la carte combinatoire qui utilise ce type de brins.

La classe principale du module est Linear_cell_complex<d,d2, Traits_ ,Items_,Alloc_> qui fournit une implantation d'un complexe cellulaire linéaire dD, plongé linéairement dans $\mathbb{R}^{d2}$. Traits_ est le noyau géométrique utilisé et peut être construit en utilisant la classe Linear_cell_complex_traits paramétrée par n'importe quel noyau de CGAL. Cela permet de choisir en fonction de ses besoin un noyau simple cartésien, un noyau possédant des prédicats exacts, ou un noyau utilisant des constructions exactes. Items_ est une classe d'items qui définit le type de brins et le type d'attributs utilisés, et Alloc_ est une classe gérant les allocations mémoires.

Il existe des paramètres par défaut pour tout les arguments à l'exception du premier. Par défaut, Linear_cell_complex<d> va définir un complexe cellulaire linéaire dD plongé dans $\mathbb{R}^d$, utilisant un noyau cartésien avec des prédicats exacts, et une classe d'items utilisant la classe Dart<d> et tout les attributs à **void** à part les 0-attributs qui sont de type

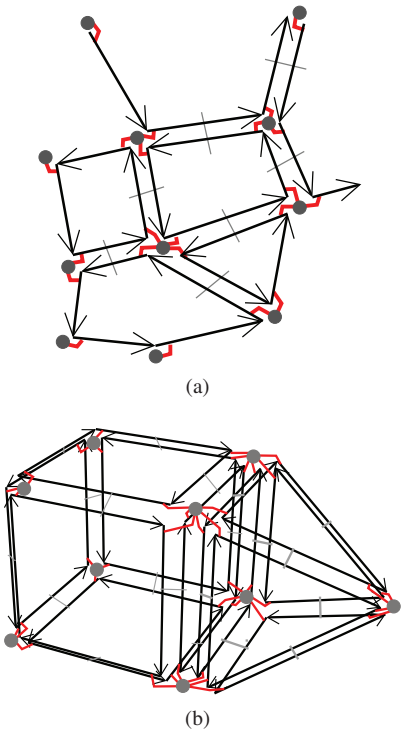


Figure 4: *Exemple de plongement d'une carte combinatoire dans un espace Euclidien. (a) Une carte combinatoire 2D avec un point 2D associé à chacun de ses sommets. (b) Une carte combinatoire 3D avec un point 3D associé à chacun de ses sommets.*

Cell_attribute_with_point , et l'allocateur qui est l'allocateur STL standard.

Nous présentons dans le Listing 1 trois exemples de définition de complexes cellulaires linéaires utilisant la classe d'items par défaut. Le premier exemple est une carte combinatoire 2D plongée dans $\mathbb{R}^2$ et correspond donc à la notion de graphe plan, le second exemple est une carte combinatoire 2D plongée dans $\mathbb{R}^3$ et correspond donc à la notion de polyèdre (ou maillage), et le troisième exemple est une carte combinatoire 3D plongée dans $\mathbb{R}^3$.

Listing 1: *Exemples de définition de complexes cellulaires linéaires standards*

```
typedef CGAL::Linear_cell_complex<2>
  Plane_graph;
typedef CGAL::Linear_cell_complex<2,3>
  Polyhedron_3;
typedef CGAL::Linear_cell_complex<3>
  LCC_3;
```

L'exemple du Listing 2 montre comment il est possible de spécialiser sa propre classe Linear_cell_complex afin de définir ses propres attributs. Dans cet exemple, un **int** est associé aux sommets (en plus d'un point 3D), et un **double** est associé aux faces. Il est bien évidemment possible d'associer n'importe quelle information au sein de n'importe quel attribut en utilisant des classes complexes au lieu de type de base. Dans toute la suite, nous notons lcc une instance de CGAL::Linear_cell_complex.

3.2. Opérations de Base

Les opérations de base sont de deux types. Tout d'abord la création d'objets de base, puis plusieurs itérateurs permettant de parcourir certaines parties des objets existants.

Pour la création des objets de base, outre la création d'un brin isolé, il existe pour le moment cinq méthodes créant des configurations spécifiques (la création d'objets plus complexes pourra s'appuyer sur ces objets de base et sur les opérations de modification) à partir de points donnant la géométrie. Toutes ces méthodes retournent un pointeur vers un nouveau brin créé par l'opération. Ces méthodes sont lcc.make_segment, lcc.make_triangle, lcc.make_quadrangle, lcc.make_tetrahedron et lcc.make_hexahedron.

Enfin il existe deux fonctions permettant de convertir d'autres structures de données CGAL vers des complexe cellulaire linéaire : import_from_triangulation_3(lcc, atr) pour ajouter dans lcc tous les tétraèdres présents dans atr, une instance de CGAL::Triangulation_3; et import_from_polyhedron_3(lcc,ap) pour ajouter dans lcc toutes les cellules présentes dans ap, une instance de CGAL::Polyhedron_3.

Afin de parcourir certaines parties d'objets existant, six itérateurs différents sont définis. Au lieu d'utiliser des couples d'itérateurs begin/end à la manière de la STL [Stl], la classe Combinatorial_map définit des classes intervalles, chacune ayant des méthodes begin() et end() retournant un itérateur sur le début et sur la fin de l'intervalle.

Il existe trois classes intervalles permettant de parcourir tout les brins d'une partie donnée :

1. Dart_range est l'intervalle de tous les brins de la carte ;
2. Dart_of_orbit_range<Beta ...> est l'intervalle de tous les brins de l'orbite <Beta ...>(d0) d'un brin d0 donné. Beta ... est une séquence ordonnée de nombre entiers tous distincts et compris entre 0 et *d* (la dimension de la carte) ;
3. Dart_of_cell_range<i> est l'intervalle de tous les brins de la *i*-cellule contenant un brin donné.

Il existe ensuite deux classes intervalles permettant de parcourir un brin de chaque *i*-cellule. Dans ces deux cas, le brin de chaque *i*-cellule peut-être quelconque.

1. One_dart_per_cell_range<i> est l'intervalle d'un brin de chaque *i*-cellule de la carte ;
2. One_dart_per_incident_cell_range<i,j> est l'intervalle d'un brin de chaque *i*-cellule incidente à une *j*-cellule donnée.

Listing 2: *Exemple de définition d'un complexe cellulaire linéaire spécifique*

```
struct Myitem {
  template<class CMap>
  struct Dart_wrapper {
    typedef CGAL::Dart<3, CMap> Dart;
    typedef CGAL::Cell_attribute_with_point< CMap, int> Myattr0;
    typedef CGAL::Cell_attribute< CMap, double> Myattr2;
    typedef CGAL::cpp0x::tuple<Myattr0,void,Myattr2> Attributes;
  };
};
typedef CGAL::Exact_predicates_inexact_constructions_kernel EPIC;
typedef CGAL::Linear_cell_complex_traits<3,EPIC> mytraits;
typedef CGAL::Linear_cell_complex<3,3,Mytraits,Myitem> My_lcc_3;
```

3.3. Opérations de Modification

Les opérations de modification sont de deux types. Un premier type d'opération permet de mettre en relation des objets ou d'enlever des relations (coutures et décousures), tandis qu'un deuxième type d'opérations permet de supprimer ou ajouter des cellules. Durant ces opérations, lorsque des cellules sont fusionnées ou sont découpées, des foncteurs On_split et On_merge sont automatiquement appelés sur les attributs associés afin que l'utilisateur puisse effectuer des traitement particuliers (par défaut ces foncteurs sont vide).

L'opération de base permettant de mettre en relation des objets existant est la méthode de couture (sew). Étant donnés deux brins i-libre dh1 et dh2, l'appel à la fonction sew<i>(dh1,dh2) va relier par β_i deux à deux les brins dh1 et dh2 ainsi que tous les brins des orbites $\langle\beta_1,\ldots,\beta_{i-2},\beta_{i+2},\ldots,\beta_n\rangle(dh1)$ et $\langle\beta_1,\ldots,\beta_{i-2},\beta_{i+2},\ldots,\beta_n\rangle(dh2)$. Intuitivement, cette opération peut être vue comme le collage de deux i-cellules par identification de deux $(i-1)$-cellules. Notons que cette opération n'est pas toujours possible. Il faut en effet qu'il existe une bijection entre les deux orbites respectant les différentes relations β. Ce test est réalisable à l'aide de la fonction **bool** is_sewable<i>(dh1,dh2) qui va retourner vrai lorsque une telle bijection existe et donc si la couture est possible, et faux sinon.

La Fig. 6 montre un exemple d'utilisation de l'opération de couture dans une 3-carte. L'appel sew<3>(1,5) va coudre par β_3 tout les brins des deux faces carrées contenant le brin 1 et le brin 5. De ce fait ces deux faces sont identifiées par l'opération, et donc les arêtes et les sommets de ces deux faces sont également identifiés deux à deux.

L'opération inverse est la méthode de décousure (unsew). Étant donné un brin non i-libre dh, l'appel à la fonction unsew<i>(dh) va découdre le β_i de tout les brins de l'orbite $\langle\beta_1,\ldots,\beta_{i-2},\beta_i,\beta_{i+2},\ldots,\beta_n\rangle(dh)$. Pour chaque brin dh de cette orbite, cela revient à affecter $\beta_i(dh)=\varnothing$. De ce fait, étant donnés deux brins i-libre dh1 et dh2, l'appel

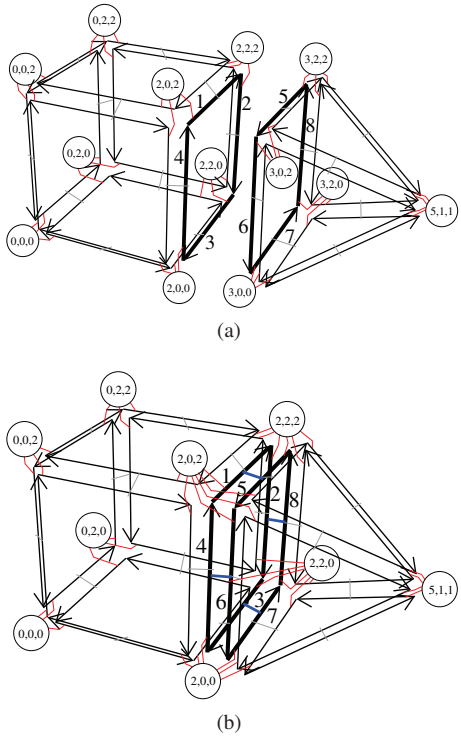


Figure 6: *Exemple d'utilisation de l'opération de 3-couture. (a) Une 3-carte composée de deux volumes déconnectés, et ayant des points 3D associés à chaque sommet. (b) La 3-carte obtenue par l'opération sew<3>(1,5) (ou par sew<3>(2,8), sew<3>(3,7), sew<3>(4,6)). Les huit 0-attributs des deux faces de la carte initiale sont fusionnés deux à deux en quatre 0-attributs dans la carte finale.*

sew<i>(dh1,dh2) suivit de l'appel unsew<i>(dh1) va donner une carte combinatoire isomorphe à la carte de départ.

Concernant les modifications de cellules, la première

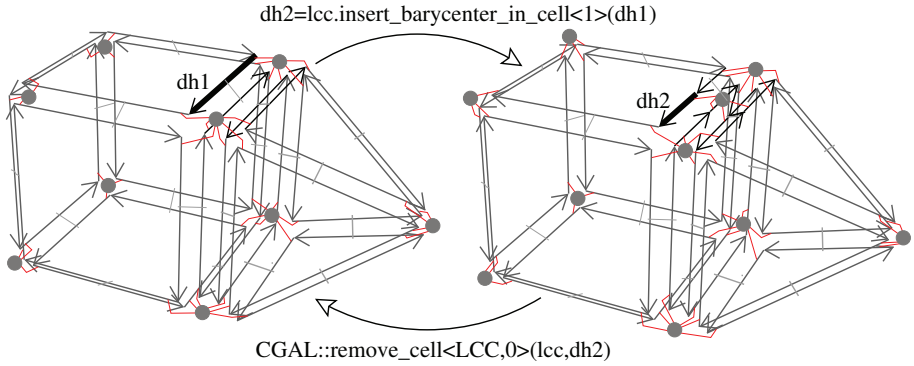


Figure 7: *Exemple d'utilisation des opérations insert_barycenter_in_cell <1> et remove_cell<0>. À gauche, le complexe cellulaire linéaire initial. À droite le complexe obtenu après l'insertion d'un point au barycentre du segment contenant le brin dh1. À partir de cette carte, si nous supprimons la 0-cellule contenant le brin dh2, nous obtenons un complexe cellulaire linéaire isomorphe au complexe initial.*

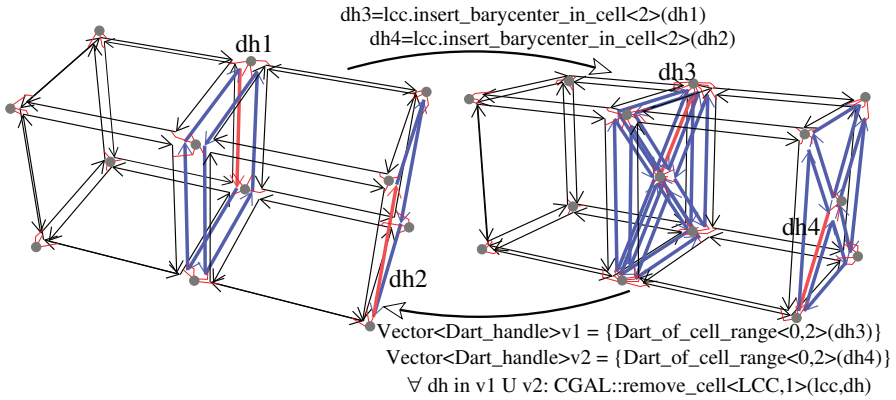


Figure 8: *Exemple d'utilisation des opérations insert_barycenter_in_cell <2> et remove_cell<1>. À gauche, le complexe cellulaire linéaire initial. À droite le complexe obtenu après l'insertion d'un point au barycentre de la face contenant le brin dh1 et un autre point au barycentre de la face contenant le brin dh2. À partir de cette carte, si nous supprimons toutes les 1-cellules incidentes aux deux sommets insérés, nous obtenons un complexe cellulaire linéaire isomorphe au complexe initial.*

opération de ce type est l'opération de i-suppression (remove_cell<i>(dh)) qui va supprimer la i-cellule désignée en fusionnant les deux $(i+1)$-cellules incidentes (lorsque $i < d$). Cette opération est définie en dimension quelconque pour tout $i \in \{0,\ldots,d\}$. Lorsque $i < d-1$, cette opération est possible uniquement s'il existe au plus deux $(i+1)$-cellules incidentes à la cellule à supprimer. Cette précondition nécessaire peut être testée à l'aide de la fonction **bool** is_removable<i>(dh) (cf. les exemples des Figs. 7 à 10).

L'opération inverse de la i-suppression est l'opération de i-insertion qui va insérer une i-cellule dans une $(i+1)$-cellule, en la coupant en deux. Contrairement à la suppression, il existe plusieurs versions différentes de l'opération d'insertion (une par dimension) car les paramètres de l'opération sont différents. Pour le moment les versions suivantes existent :

- insert_barycenter_in_cell <1>(dh) qui va insérer un sommet au milieu du segment contenant le brin dh (cf. exemple Fig. 7). Le segment initial est coupé en deux ;
- insert_barycenter_in_cell <2>(dh) qui va insérer un sommet au milieu du polygone contenant le brin dh (cf. exemple Fig. 8). Le polygone initial est coupé en k triangles, un par arête du polygone initial ;
- insert_cell_1_in_cell_2 (lcc,dh1,dh2) qui va insérer une arête entre les brins dh1 et dh2 (qui doivent appartenir à la même face, cf. exemple Fig. 9). Le polygone initial est coupé en deux ;
- lcc. insert_dangling_cell_1_in_cell_2 (dh,p) qui va insérer une arête pendante dans la face contenant le brin dh, et incidente au sommet contenant ce même brin, l'extrémité pendante de l'arête ayant p comme point (cf. exemple Fig. 9) ;

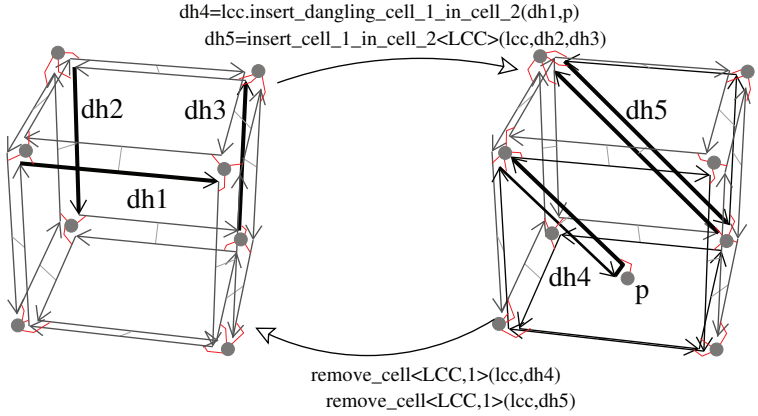


Figure 9: *Exemple d'utilisation des opérations* insert_cell_1_in_cell_2 *et* insert_dangling_cell_1_in_cell_2 *, et* remove_cell<1>*. À gauche, le complexe cellulaire linéaire initial. À droite le complexe obtenu après l'insertion d'une arête entre les brins dh2 et dh3 et l'insertion d'une arête pendante incidente au sommet contenant le brin dh1. À partir de cette carte, si nous supprimons les deux arêtes contenant les brins dh4 et dh5, nous obtenons un complexe cellulaire linéaire isomorphe au complexe initial.*

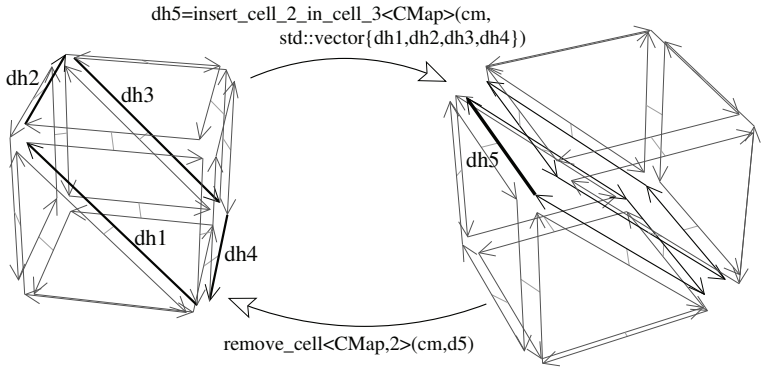


Figure 10: *Exemple d'utilisation des opérations* insert_cell_2_in_cell_3 *et* remove_cell<2>*. À gauche, le complexe cellulaire linéaire initial. À droite le complexe obtenu après l'insertion d'une face le long du chemin d'arêtes contenant les brins dh1, dh2, dh3 et dh4. À partir de cette carte, si nous supprimons la 2-cellule contenant le brin dh5, nous obtenons un complexe cellulaire linéaire isomorphe au complexe initial.*

– insert_cell_2_in_cell_3 (itbegin , itend) qui va insérer une face au milieu du volume contenant les brins se trouvant dans l'intervalle [itbegin , itend) (cf. exemple Fig. 10). Le volume initial est coupé en deux.

Ces opérations de modification peuvent servir pour développer des algorithmes de simplification ou de raffinement en dimension quelconque. Groupées, elles peuvent également servir de briques de base pour le développement d'opérations de plus haut niveau, comme par exemple la création d'une porte dans un mur pour le modeleur de bâtiment, où la fusion de régions dans le cadre de la segmentation d'images.

4. Conclusion

Dans cet article, nous avons présenté les deux nouveaux modules développés dans CGAL : `Combinatorial map` et `Linear cell complex`. Ces deux modules permettent de représenter n'importe quelle subdivision de quasi-variété dD orientable plongée dans $\mathbb{R}^{d2}$, en associant des informations à n'importe quelle cellule. Toutes les classes sont paramétrables et l'utilisateur de ces modules peut choisir le type de brins ainsi que les types des attributs qu'il souhaite utiliser.

Les opérations proposées sont pour le moment limitées à de nombreux itérateurs permettant de parcourir les objets représentés, la création d'objets de base, les opérations de coutures et décousures, et des opérations de suppression et insertion de cellules.

Les travaux futurs sont nombreux. En effet, ces modules sont très récents et de nombreuses améliorations sont possibles. Nous envisageons tout d'abord de développer de nou-

velles opérations pour enrichir ces modules. Nous souhaitons également développer des applications basées sur ces structures afin de mettre en avant leurs intérêts. Nous souhaitons entre autre porter certains de nos logiciels existant (Moka et des logiciels de traitement d'images 2D et 3D basés cartes combinatoires) afin qu'ils utilisent ces modules. Cela permettra de mettre en évidence l'intérêt de cette librairie et de sa généricité. Enfin nous voulons ajouter la possibilité de représenter des objets troués.

5. Remerciements

Merci à Monique Teillaud pour son aide importante lors de la rédaction des manuels de `Combinatorial map` et `Linear cell complex`, à Andreas Fabri, Sébastien Loriot et Laurent Rineau pour leur participations et leur aide dans le développement des modules et de la documentation, à Bernd Gärtner pour ses commentaires sur les manuels et à Sylvain Brandel pour sa relecture de cet article. Ce travail a été partiellement financé par l'agence nationale française (ANR), au travers du projet DIGITALSNOW ANR-11-BS02-009.

Références

[Aci] Acis 3d solid modeler. `http://www.spatial.com/products/3d-acis-modeling`.

[AK89] ARQUES D., KOCH P. : Modélisation de solides par les pavages. In *Proceedings of Pixim 89* (Paris, 1989), pp. 47–61.

[BP07] BECHMANN D., PEROCHE B. : *Informatique Graphique, modélisation géométrique et animation*. Traité IC2 - Information - Commande - Communication. Hermès, Feb 2007.

[Bri93] BRISSON E. : Representing geometric structures in *d* dimensions : topology and order. *Discrete & Computational Geometry. Vol. 9*, Num. 1 (1993), 387–426.

[Cga] CGAL : **C**omputational **G**eometry **A**lgorithms **L**ibrary. `http://www.cgal.org`.

[Cgo] CGoGN : **C**ombinatorial and **G**eometric m**o**deling with **G**eneric **N**-dimensional maps. `http://cgogn.u-strasbg.fr/Wiki/index.php/CGoGN`.

[Cpy] Combinatorial pyramids. `http://www.greyc.ensicaen.fr/~luc/PYRAMIDES`.

[Dam10] DAMIAND G. : *Contributions aux Cartes Combinatoires et Cartes Généralisées : Simplification, Modèles, Invariants Topologiques et Applications*. Habilitation à diriger des recherches, Université Lyon 1, Septembre 2010.

[Dam12a] DAMIAND G. : Combinatorial maps. In *CGAL User and Reference Manual*, 4.0 ed. CGAL Editorial Board, 2012. `http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html#Pkg:CombinatorialMaps`.

[Dam12b] DAMIAND G. : Linear cell complex. In *CGAL User and Reference Manual*, 4.0 ed. CGAL Editorial Board, 2012. `http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html#Pkg:LinearCellComplex`.

[DL87] DOBKIN D. P., LASZLO M. J. : Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of the third annual symposium on Computational geometry* (1987), SCG '87, ACM, pp. 86–99.

[Dup09] DUPAS A. : *Opérations et Algorithmes pour la Segmentation Topologique d'Images 3D*. Thèse de doctorat, Université de Poitiers, Novembre 2009.

[Ede87] EDELSBRUNNER H. : *Algorithms in Computational Geometry*. Springer, New-York, 1987.

[Gir] GIRL : **G**eneral **I**mage **R**epresentation **L**ibrary. `http://girl.labri.fr`.

[Hor08] HORNA S. : *Reconstruction géométrique et topologique de complexes architecturaux 3D à partir de plans numériques 2D*. Thèse de doctorat, Université de Poitiers, Novembre 2008.

[Ket12a] KETTNER L. : 3D polyhedral surfaces. In *CGAL User and Reference Manual*, 4.0 ed. CGAL Editorial Board, 2012. `http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html#Pkg:Polyhedron`.

[Ket12b] KETTNER L. : Halfedge data structures. In *CGAL User and Reference Manual*, 4.0 ed. CGAL Editorial Board, 2012. `http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html#Pkg:HDS`.

[Lie91] LIENHARDT P. : Topological models for boundary representation : a comparison with n-dimensional generalized maps. *Commputer Aided Design. Vol. 23*, Num. 1 (1991), 59–82.

[Mok] MOKA : **Mo**deleur de **ka**rtes. `http://moka-modeller.sourceforge.net`.

[Ope] Open mesh. `http://openmesh.org`.

[PABL07] POUDRET M., ARNOULD A., BERTRAND Y., LIENHARDT P. : *Cartes Combinatoires Ouvertes*. Research Notes 2007-1, Laboratoire SIC E.A. 4103, F-86962 Futuroscope Cedex, France, October 2007.

[Par] Parasolid 3d geometric modeling engine. `http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml`.

[Stl] STL : **S**tandard **T**emplate **L**ibrary. `http://www.sgi.com/tech/stl`.

[Wei88] WEILER K. : The radial edge structure : a topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications*, Wozny M., McLaughlin H., Encarnacao J., (Eds.). Elsevier Science, 1988, pp. 217–254.