Segmentation de nuage de points 3D pour la modélisation automatique d'environnements industriels numérisés

Thomas Chaperon

ECOLE DES MINES
DE PARIS
Collège doctoral

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

T H E S E

pour obtenir le grade de
Docteur de l'Ecole des Mines de Paris
Spécialité «Informatique temps réel, Robotique et Automatique»

présentée et soutenue publiquement par
Thomas Chaperon

le 23 septembre 2002

SEGMENTATION DE NUAGE DE POINTS 3D POUR LA MODELISATION AUTOMATIQUE D'ENVIRONNEMENTS INDUSTRIELS NUMERISES

Jury

M. Jacques Droulez	Rapporteur
M. Francis Schmitt	Rapporteur
M. François Goulette	Directeur de thèse
M. Claude Laurgeau	Examinateur
M. Rachid Deriche	Examinateur
M. Xin Chen	Examinateur
M. Guillaume Thibault	Examinateur

Il faut traiter la nature par le cône, le cylindre et la sphère.
Cézanne.

Summary

The context of this work is the 'As-built' CAD modeling of digitized industrial structures. The existing environment is first digitized using a 3D laser scanner. The data obtained this way consist of an unorganized and inhomogeneous 3D point cloud. The next step is the segmentation of this cloud and the reconstruction of the various surfaces constituting the CAD model of the scene. Industrial environments yield scenes that are complex with respect to the size of data and the number of objects, but that can be described with simple geometric primitives : the plane, the sphere, the cylinder, the cone, the torus. The current tools that process 3D point clouds are not able to perform this segmentation task automatically.

In this project, algorithms have been developed for this purpose. We have particularly focused on the segmentation of pipelines. The software solutions implemented have been validated through tests involving users of the current tools. The main feature of the methods developed is the use, throughout the segmentation process, of constrained geometric primitives. The corresponding constraints are derived from a ground knowledge of the environment (e.g., continuity or tangency relationships). Geometric primitive fitting is a key topic in this work. The algorithms developed, which use a truly geometric definition of the primitives, show good results and performance on real scenes. Moreover, one of the major issue is then the validation of the fitted model. We hence have examined this question and given original methods based on statistics. Another contribution of this work relies in the algorithms for extracting a primitive from a point cloud. These methods have been applied in the context of pipelines, but seem to be relevant in the more general question of totally automatic modeling of digitized environments.

Key words : 3D modeling, CAD, vision, segmentation, surface recognition, geometric primitives, surface fitting / approximation, least squares, model validation, primitive extraction.

Résumé

Le contexte de ce travail est la modélisation CAO «tel que construit» de grandes structures industrielles numérisées (usine, ...). L'environnement existant est tout d'abord numérisé à l'aide d'un scanner laser. Les données ainsi obtenues prennent la forme d'un nuage de points 3D non-structuré et non-homogène. L'étape suivante consiste à segmenter ce nuage de points et reconstruire les différentes surfaces constituant le modèle CAO de la scène. Les environnements industriels génèrent des scènes complexes par le nombre de données et d'éléments présents, mais qui se décrivent par des primitives géométriques simples : plan, sphère, cylindre, cône, tore. Les outils actuels traitant les nuages de points 3D ne permettent pas de réaliser cette segmentation de manière automatique.

Dans cette thèse, des algorithmes ont été développés dans ce but. L'attention a en particulier été portée sur la segmentation des lignes de tuyauterie. Les solutions logicielles implémentées dans ce cadre ont été validées par des tests auprès d'utilisateurs experts des outils actuels. Les méthodes développées se caractérisent par l'utilisation au cours de la segmentation de primitives géométriques contraintes, issues de connaissances «métier» (par exemple relations de continuité ou de tangence). L'ajustement de primitive géométrique est un élément de base au sein de ces travaux. Les procédés mis en oeuvre, qui utilisent une définition véritablement géométrique des primitives, montrent de bonnes performances en pratique. D'autre part, l'un des problèmes majeurs concerne les moyens de valider le modèle ajusté. La question de la validation de modèle géométrique a été examinée. Nous présentons des méthodes originales construites à partir d'outils statistiques. Enfin, une autre contribution de cette thèse se situe au niveau des algorithmes d'extraction de primitives géométriques d'un nuage de points. Les méthodes présentées ont été appliquées dans le contexte des lignes de tuyauterie, mais semblent également pertinentes pour résoudre la question plus générale de la modélisation totalement automatique d'un environnement numérisé.

Mots clés : modélisation 3D, CAO, vision, segmentation, reconnaissance de surfaces, primitives géométriques, ajustement de surfaces / approximation, moindres carrés, validation de modèle, extraction de primitives.

Table des matières

Notations et conventions

Les points de l'espace euclidien de dimension 3 et vecteurs de $\mathbb{R}^3$ sont notés sans distinction. En particulier, on désigne la droite (de $\mathbb{R}^3$) qui passe par le point $\mathbf{p}$ et de vecteur directeur $\mathbf{v}$ par l'expression

$$\mathbf{p}+\mathbb{R}\mathbf{v}$$

De même, on désigne le plan (de $\mathbb{R}^3$) qui passe par le point $\mathbf{p}$ et de vecteur normal $\mathbf{v}$ par l'expression

$$\mathbf{p}+\mathbf{v}^{\perp}$$

Etant donné deux vecteurs $\mathbf{a}$ et $\mathbf{b}$ de $\mathbb{R}^3$, $\mathbf{a}.\mathbf{b}$ désigne le produit scalaire euclidien de $\mathbb{R}^3$, $\|\mathbf{a}\| = \sqrt{\mathbf{a}.\mathbf{a}}$ la norme euclidienne associée, et $\mathbf{a} \times \mathbf{b}$ désigne le produit vectoriel.

La base canonique de $\mathbb{R}^3$ est notée $\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z}$. L'origine du repère est notée comme le vecteur nul : $\mathbf{0}$. Dans ce repère $(\mathbf{0}; \mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z})$, on définit les vecteurs $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{w_1}$ à partir des coordonnées sphériques θ et ϕ (Figure 1) :

$$\mathbf{u} = \begin{pmatrix} \cos\phi\sin\theta \\ \sin\phi\sin\theta \\ \cos\theta \end{pmatrix}, \mathbf{v} = \begin{pmatrix} \cos\phi\cos\theta \\ \sin\phi\cos\theta \\ -\sin\theta \end{pmatrix}, \mathbf{w} = \begin{pmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{pmatrix}, \mathbf{w_1} = \begin{pmatrix} \cos\phi \\ \sin\phi \\ 0 \end{pmatrix} \tag{1}$$

avec $\theta \in [0, \pi]$ et $\phi \in [-\pi, \pi[$. Dans la suite, sauf mention contraire, les termes $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{w_1}$ désignent ces vecteurs et sont ainsi liés aux angles θ et ϕ.

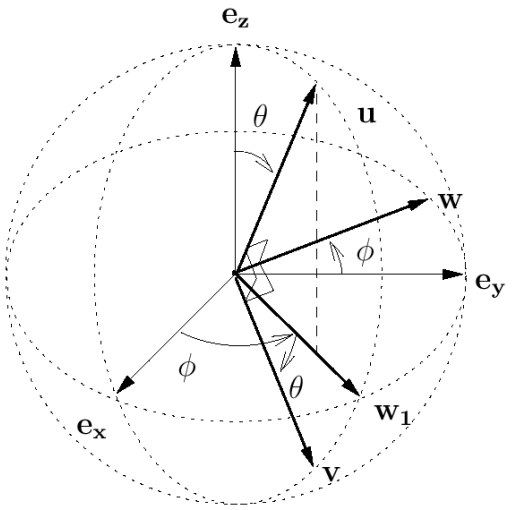


FIG. 1 – Conventions pour les coordonnées sphériques, et définition des vecteurs $\mathbf{u}, \mathbf{v}, \mathbf{w}$ et $\mathbf{w_1}$

Chapitre 1

Introduction

La rétro-ingénierie est connue principalement pour les pièces mécaniques. Au niveau des grandes structures industrielles, il existe également un besoin de modèles CAO fidèles à l'existant (Figure 1.1). En effet, les plans de conception de telles installations ne sont pas toujours disponibles ou à jour. Les

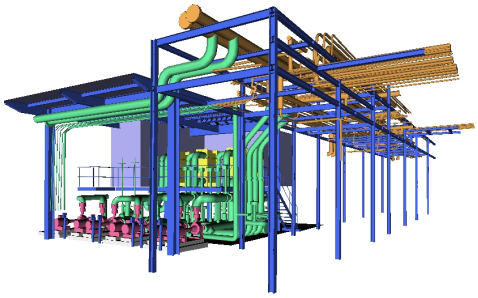


FIG. 1.1 – Modèle CAO d'un environnement industriel

modèles d'installations existantes répondent à des besoins spécifiques : modification d'installation, maintenance, démantèlement, construction de nouvelles installations dans un environnement existant, mise aux normes, gestion de production assistée par ordinateur, réalité virtuelle pour la formation, etc.

La modélisation «tel que construit» a été introduite par la topographie et la photogrammétrie. Cette discipline a connu un nouvel essor avec l'émergence de techniques de numérisation 3D à partir de laser.

La société MENSI, filiale d'EDF, développe et commercialise des scanners 3D : Soisic (S10, S25) et GS100. MENSI édite également des logiciels dédiés : 3Dipsos et RealWorks Survey, qui permettent de traiter les données issues des scanners afin de construire le modèle CAO souhaité.

Les environnements industriels peuvent se caractériser par deux aspects :

– les vues peuvent être denses et très complexes étant donné le nombre d'éléments présents : tuyauterie, bâtiment, conduits d'aération, etc.(Figure 1.1) ;
– on peut en général décrire la plupart des formes présentes dans la scène par des primitives géométriques simples, telles que le plan, la sphère, le cylindre, le tore, ou le cône (Figure 1.2).

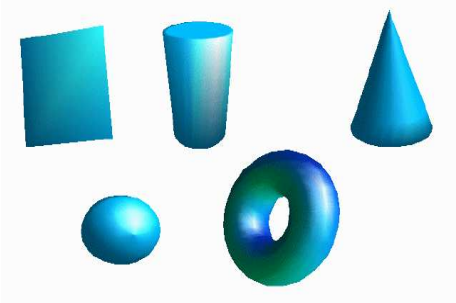


FIG. 1.2 – Primitives géométriques avec lesquelles on peut décrire la plupart des surfaces présentes dans les scènes industrielles

En outre, les scènes industrielles sont constituées en grande partie de lignes de tuyauterie.

1.1 La numérisation 3D

Les deux techniques les plus couramment utilisées par les scanners laser sont la triangulation optique et le temps de vol.

1.1.1 Triangulation optique

Le principe de la triangulation optique est le suivant : l'émetteur envoie un faisceau laser avec un angle connu. Ce faisceau est (en partie) réfléchi par la surface, notamment dans la direction du capteur. Au niveau du scanner, après la lentille du capteur, une matrice CCD (*charged couple device*) permet de mesurer l'endroit éclairé par le faisceau. Cette mesure donne directement l'angle du faisceau au niveau du capteur. La distance entre l'émetteur et le capteur étant connue, la distance du point d'impact peut être déduite par trigonométrie (Figure 1.3). Le principe de la triangulation est utilisé par les géomètres, les marins ou même par les astronomes [1] depuis fort longtemps. Avec le laser, ce principe fournit des

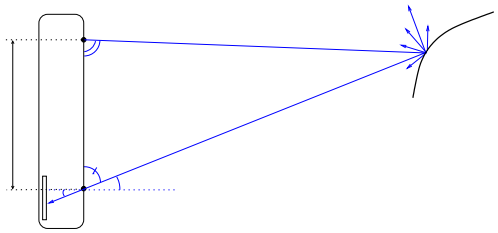


FIG. 1.3 – Principe de la triangulation

données très précises à courtes distances, puisque le triangle n'est pas dégénéré. A titre d'exemple, les scanners Soisic S10 et S25 de MENSI (Figure 1.4), construits sur ce principe, donnent des résultats optimaux dans des gammes respectives de 0,8m à 10m, et de 2m à 25 m. Lorsque la distance augmente, la précision diminue en théorie comme le carré de la distance.

[1] la distance des astres était autrefois estimée par ce biais, en utilisant le décalage entre les positions la Terre à différents instants de sa révolution autour du Soleil.

FIG. 1.4 – Scanner Soisic(MENSI)

Les données sont obtenues par un double balayage angulaire.

1.1.2 Temps de vol

Le principe du temps de vol est de mesurer le temps que met la lumière pour parcourir la distance du scanner à la surface et revenir (Figure 1.5).

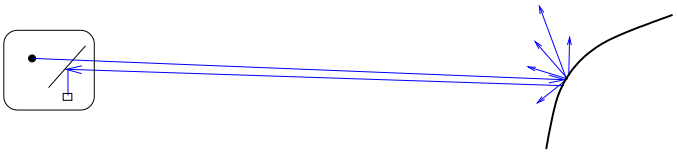


FIG. 1.5 – Principe du temps de vol

Etant donné que le temps mis par la lumière est en général très faible, et qu'il est techniquement difficile de mesurer des temps aussi court, on recourt à la modulation de phase. Cette technique consiste à moduler le signal émis et de mesurer la différence de phase du signal reçu [Gal01, chap.1].

Le scanner GS100 de la société MENSI(Figure 1.6) fonctionne sur ce principe. C'est le cas également de la plupart des autres scanners longue distance disponibles sur le marché. Certains de ces scanners sont brièvement présentés section 3.2.3, page 58. Avec ce principe, la précision ne dépend en théorie plus de la distance au scanner : les points lointains sont aussi précis (voire plus) que des points proches du scanner. Le scanner GS100 est par exemple spécifié pour acquérir des points de 2 à 100 mètres de distance.

FIG. 1.6 – Scanner GS100(MENSI)

Les données sont également obtenues par un double balayage angulaire.

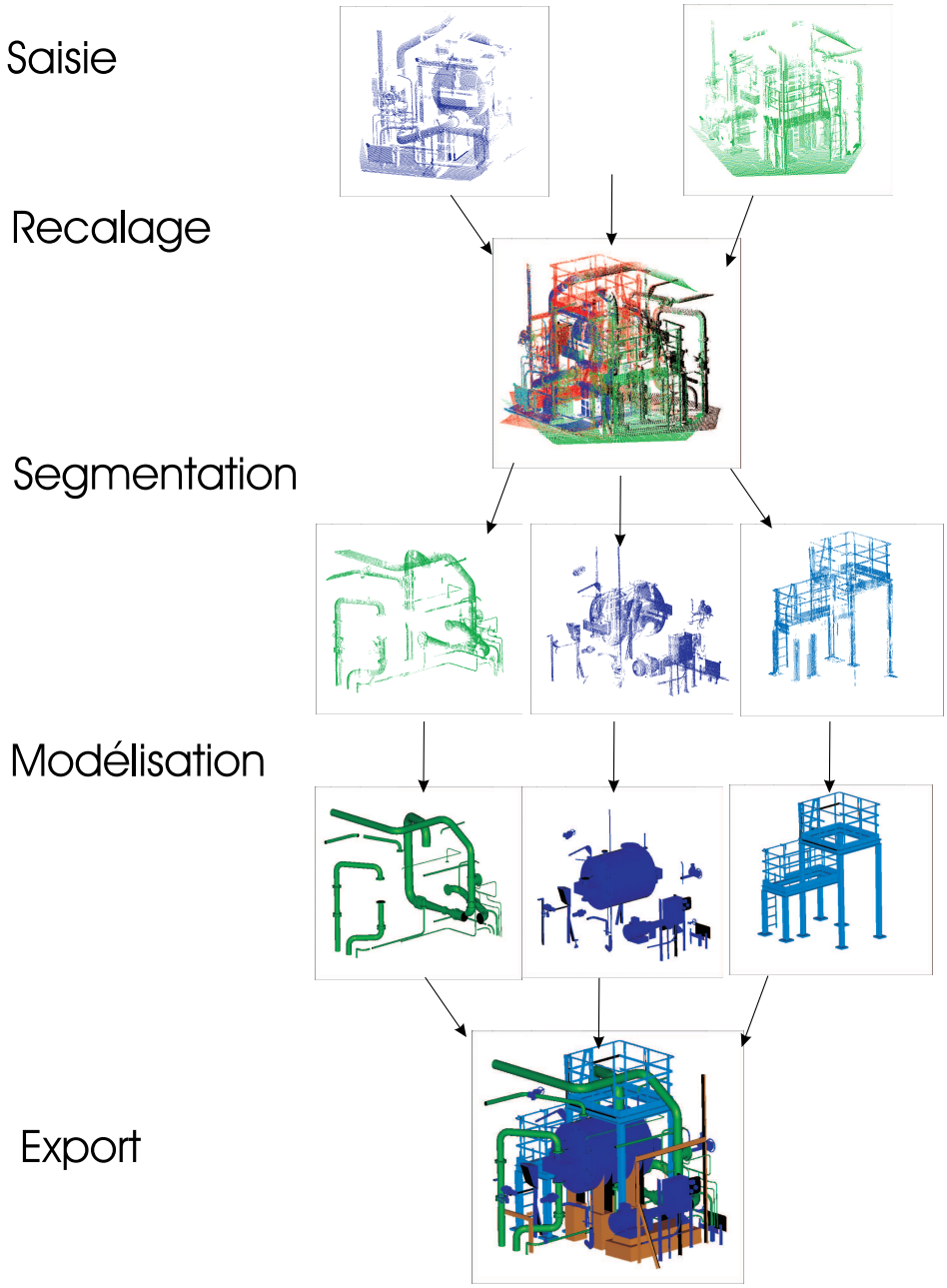


FIG. 1.7 – Etapes de la modélisation «tel que construit»

1.2 Etapes de la modélisation «tel que construit»

La modélisation tel que construit utilisant les produits de MENSI se déroule en plusieurs étapes (Figure 1.7) :

- Saisie des données sur site
- Pré-traitement des données (vérification qualité, recalage des points de vue)
- Segmentation et modélisation

– Export vers des logiciels spécialisés

Il convient ici de décrire ces étapes plus en détails.

1.2.1 Saisie des données

La saisie des données consiste à réaliser des «scans» depuis différents points de vue de la scène à numériser. Pour des environnements complexes ou de grande taille, il peut être nécessaire d'effectuer un grand nombre de points de vue (quelques dizaines). La planification des vues à réaliser est en général une tâche complexe. Si plusieurs scanners sont disponibles, il est possible d'effectuer la saisie de différentes zones en parallèle.

1.2.2 Pré-traitement

Le pré-traitement des données comprend les vérifications éventuelles de la qualité de la numérisation et surtout le recalage des vues dans un même repère, pour obtenir le nuage de points. Ceci est réalisé dans le logiciel 3Dipsos. En général, on place des sphères aimantées dans la scène, qui sont étiquetées lors de l'acquisition (depuis PointScape, le logiciel de pilotage du scanner). Dans ce cas, le recalage se fait de manière automatique dans 3Dipsos. Une vérification du recalage par l'utilisateur s'impose à ce niveau. Pour des projets nécessitant plusieurs jours (ou nuits) d'acquisition, ce pré-traitement est réalisé au fur et à mesure (typiquement une fois par jour).

1.2.3 Segmentation et modélisation

La phase suivante consiste à traiter le nuage de points.

Méthode de segmentation manuelle

La segmentation manuelle est réalisée par des filtrages successifs. Dans une vue donnée, l'utilisateur dessine un polygone incluant l'entité à extraire et met de côté les points qui se trouvent à l'intérieur du volume délimité par ce polygone. Il est nécessaire de répéter cette tâche dans plusieurs vues pour parvenir à dégager complètement l'entité (Figure 1.8).

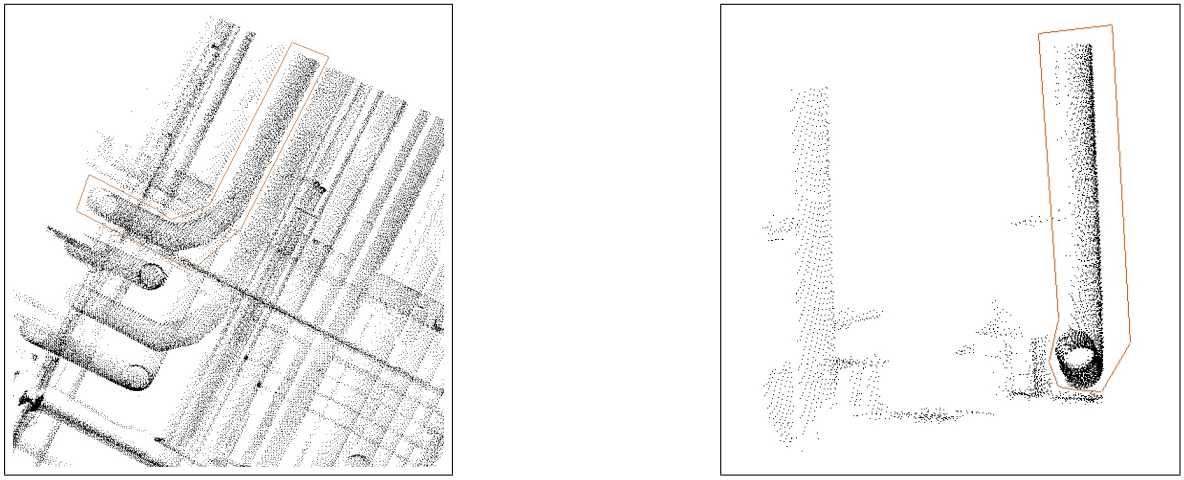


FIG. 1.8 – Principe de la segmentation manuelle dans 3Dipsos.

Segmentation

Pour les gros projets, la segmentation peut commencer par une séparation des différents métiers : tuyauterie, structures (poutres, escaliers, ...), équipements(appareils), etc. (Figure 1.7). Ce premier découpage permet à plusieurs utilisateurs de travailler en même temps sur la même scène (l'un traitant la tuyauterie, un autre traitant les structures, etc.).

> Remarque
> Dans un contexte multi-utilisateurs, il est également possible de découper simplement le nuage en grandes zones, mais se pose alors le problème des raccords aux frontières entre les zones. Ce problème concerne surtout les lignes de tuyauterie, qui doivent être cohérentes dans toute la scène. Or si deux portions de la même ligne de tuyauterie ont été modélisées séparément, la jonction des deux portions ne sera pas bonne (décalage des axes, rayons différents).

La segmentation comprend ensuite la séparation des éléments distincts : poutre, ligne de tuyauterie, mur, escalier, pompe, etc.

Modélisation

La phase de modélisation est celle qui permet de générer les surfaces du modèle CAO final. Pour cela, il est encore nécessaire de segmenter chacun des éléments, cette fois en primitives géométriques simples : plan, cylindre, cône, tore, sphère. A l'issue de cette segmentation, chaque sous-nuage correspond à une seule primitive géométrique (Figure 1.9).

FIG. 1.9 – Principe de la modélisation dans 3Dipsos.

La phase finale de la modélisation consiste, pour chacun de ces sous-nuages, à spécifier un type de surface à ajuster sur les points. Le logiciel ajuste la surface du type spécifié sur le sous-nuage. Une fois cela fait, l'utilisateur peut définir les extrémités de cette surface (Figure 1.9).

Les primitives spécifiées intègrent généralement des contraintes entre surfaces voisines. C'est le cas en particulier pour les lignes de tuyauterie : des relations lient par exemple les cylindres de la figure 1.9 (même rayon, axes concourants). En outre, cet ajustement peut éventuellement se faire avec utilisation de catalogues standard afin de faciliter l'export vers les logiciels de CAO (PDS ou PDMS par exemple).

Spécificité des lignes de tuyauterie

Considérons maintenant le cas des lignes de tuyauterie. Pour des scènes de taille et de complexité importantes, cette étape est fastidieuse, et ce pour deux raisons. Tout d'abord, les lignes de tuyauterie étant en général longues, elles s'étendent sur de grandes parties de la scène complète, ce qui fait que l'on peut difficilement travailler sur des zones plus petites. Ensuite, il peut être relativement difficile de visualiser le cheminement d'une ligne dans une scène comportant de nombreux éléments. Ceci est lié au fait que la perception des formes dans un nuage de points n'est pas toujours aisée : les points ne cachent pas ceux qui sont derrière, il y a des ambiguïtés «avant-arrière», les formes ne se perçoivent en général facilement que depuis certains points de vue, etc. Notons que la visualisation du nuage de points peut tout de même être améliorée : affichage avec simulation de l'éclairage à partir des normales (orientées ou non), brouillard en fonction de la profondeur, luminance (si disponible).

> Remarque
> De récents développements ont accéléré cette étape (fonction «smart line» dans 3Dipsos v2.4) : l'utilisateur dessine une ligne brisée suivant la tuyauterie (chaque segment correspond alors à peu près à une primitive). Toutefois, cette méthode s'avère sensible aux points qui sont cliqués par l'utilisateur.

1.2.4 Exportation du modèle

Enfin, la phase d'exportation permet d'utiliser le modèle «tel que construit» comme un modèle CAO classique. Pour les environnements industriels, l'export se fait en particulier vers les logiciels PDS (InterGraph), PDMS (CadCentre), Microstation (Bentley).

1.2.5 Automatisation de la segmentation-modélisation

Motivation

Pour des environnements de grande taille, cette chaîne de traitement peut nécessiter plusieurs semaines de temps-homme. En particulier, l'étape de segmentation-modélisation est de loin la plus longue. En effet, les ratios observés du temps de modélisation sur le temps d'acquisition des données sont généralement assez élevés. Ce ratio varie bien entendu selon l'ampleur du projet, le type de scène et le modèle souhaité. Pour donner un ordre de grandeur, voici deux exemples d'estimation de ce ratio, pour des applications industrielles utilisant la technologie développée par MENSI (scanner Soisic, logiciel 3Dipsos) [2] :

Exemple 1 (société Technip) : pour un gros projet (8 semaines sur site, 36 nuits-scanner de saisie).
Ratio modélisation/saisie = 4 environ.

Exemple 2 (société Chleq Froté) : pour un projet comportant 50h d'acquisition, en excluant la vérification finale et la modélisation pour le logiciel PDS.
Ratio modélisation/saisie = 2,8 environ.

En outre, ce ratio tend à augmenter étant donné que les scanners 3D sont de plus en plus rapides : pour la même durée d'acquisition, on obtient beaucoup plus de points qu'auparavant (environ 100 points par seconde pour le scanner Soisic, et 1000 points par seconde pour le scanner GS100).

[2] Informations recueillies lors du «club ingénierie» de MENSI le 14/12/2000 réunissant plusieurs cabinets d'ingénierie et bureau d'études utilisant la technologie de MENSI.

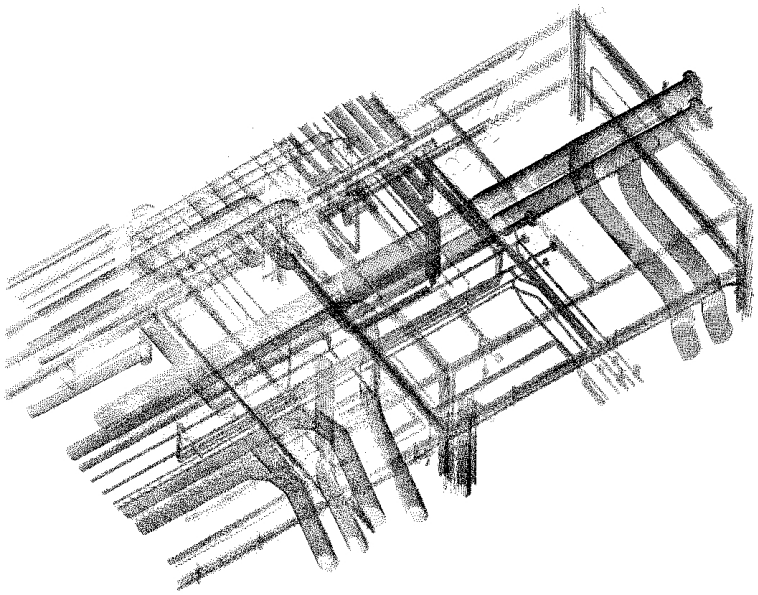


FIG. 1.10 – Partie d'un environnement industriel numérisé

Sujet de la thèse

Le présent projet avait donc pour objectif de rendre les étapes de segmentation et de modélisation les plus automatiques possibles dans le but d'en augmenter la productivité. Comme nous l'avons dit en introduction, les lignes de tuyauterie constituent une grande partie des environnements industriels. L'attention a donc été portée en premier lieu sur ces éléments.

1.3 Caractéristiques des données 3D

Les données 3D disponibles à l'issue de l'étape de recalage se présentent sous la forme d'un nuage de points 3D(Figure 1.10). On entend ici par nuage de points 3D une simple liste de coordonnées :

$$\begin{matrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{matrix}$$

Ce nuage de points est caractérisé par un ensemble important et désorganisé de points, une densité de points non-homogène et un bruit sur les points.

1.3.1 Un ensemble volumineux et non-structuré de points

Un nuage typique comporte quelques millions de points. De plus, avec les progrès des procédés d'acquisition, les tailles de nuages de points à traiter augmentent rapidement.

Dans les fichiers «.soi» provenant du scanner Soisic, il y a encore l'information du point de vue, c'est-à-dire la position du scanner dans la vue correspondante, associée à chaque point. De plus, les points suivent l'ordre du scan : les points immédiatement avant et après dans la liste de coordonnées

sont les points respectivement précédent et suivant dans la ligne de scan, hormis aux extrémités d'une ligne, où le point après se trouve au début de la ligne suivante (de l'autre côté). Dès lors que plusieurs vues sont recalées et fusionnées, il n'y a plus de structure de grille ou de ligne de scan. En particulier, dans les fichiers «.neu» provenant de Soisic, il n'y a plus d'information sur la position du scanner. Cependant, les points sont encore regroupés par points de vue dans la liste. Ceci implique que l'ordre des lignes de scan est quand même préservé : lorsque l'on est dans un même point de vue, les points avant et après sont voisins sur la ligne de scan (toujours hormis les extrémités). On peut donc dire que des points voisins dans la liste sont «la plupart du temps» voisins (au sens des lignes de scan) dans la scène. Ceci étant, on ne sait plus quand ils le sont ou quand ils ne le sont pas.

1.3.2 Une densité de points non-homogène

Cette propriété provient :
- de la distance et de l'orientation de la surface par rapport au scanner (dans chaque point de vue),
- de parties scannées dans plusieurs points de vue et d'autres par un seul,
- des occlusions (Figure 1.11),
- de scans spécifiques pour certains détails («sous-point de vue»).

FIG. 1.11 – Exemple de différence de densité de points

Une conséquence des points 1.3.1 et 1.3.2 est qu'il n'est pas évident de trouver les «vrais» voisins (c'est-à-dire des points voisins se trouvant sur la même surface) : par exemple, dans la même scène, peuvent se trouver des points avec un pas de 10 cm (sur le sol), et deux tuyaux qui sont à moins de 5cm de distance.

1.3.3 Des points bruités

Le bruit sur les points a trois origines :
- dispersion (bruit a priori gaussien étalé dans la direction du scanner, dépend de l'angle à la surface et de sa spécularité),
- distorsion (méconnue : ceci nécessite un étalonnage en 3D du scanner),
- erreurs de recalage (Figure 1.12).

Dans la pratique, on supposera que les erreurs provenant de la distorsion et du recalage ne sont pas trop importantes. En particulier, il ne semble pas réaliste de vouloir traiter automatiquement les situations extrêmes, c'est-à-dire pour lesquelles un utilisateur aurait lui-même des difficultés.

La dépendance du niveau de bruit par rapport à la distance au scanner dépend du type de scanner. En effet, l'erreur en distance d'un scanner à triangulation (ex : Soisic) croît théoriquement comme le carré de la distance au scanner (ordre de grandeur pour Soisic : 1 mm lorsque l'on se trouve à 5m du scanner), alors que celle d'un scanner à temps de vol est théoriquement indépendante de la distance

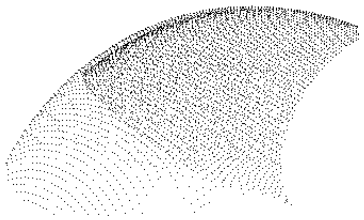


FIG. 1.12 – Exemple de zone où il existe un (léger) problème de recalage.

[LNdC95, p.11] . Notons toutefois que le bruit total dépend aussi d'autres paramètres, comme par exemple de l'orientation de la surface par rapport à la direction du faisceau lumineux, mais aussi de facteurs tels que la spécularité des surfaces, la lumière ambiante ou même la température.

Pour synthétiser, on peut dire que l'on ne dispose (pour l'instant) pas d'information sur le niveau de bruit des points, et qu'il n'est pas évident de déduire cette information du nuage de points seul.

1.4 Contributions de la thèse et structure du document

Ce document est structuré de la façon suivante. Le chapitre 2 présente un état de l'art des domaines connexes au problème posé. Le chapitre 3 décrit les principaux algorithmes qui ont été développés pour résoudre le problème de la segmentation de lignes de tuyauterie. Les chapitres qui suivent concernent essentiellement les outils qui sont utilisés dans les algorithmes du chapitre 3. Le chapitre 4 présente la méthode développée pour réaliser l'ajustement de primitive géométrique sur un ensemble de points. Ensuite, le chapitre 5 décrit comment reconnaître quelle est la meilleure primitive parmi plusieurs ajustées sur un nuage de points et décrit différents outils pour valider ou invalider une primitive. Enfin, le chapitre 6 traite de l'extraction de primitives géométriques d'un nuage de points, et conduit vers la modélisation automatique. Les conclusions et perspectives générales font l'objet du chapitre 7. Les liens entre les thèmes traités dans les différents chapitres sont symbolisés sur le diagramme de la figure 1.13.

L'une des spécificités du sujet est le traitement d'un nuage de points 3D non-structuré, avec des données de densité non-uniforme. En effet, même s'il existe un nombre croissant de scanners 3D pouvant numériser des environnements de grande taille, les données de ce type restent à l'heure actuelle plutôt rares dans les travaux de recherche en vision et en rétro-ingénierie.

L'une des contributions de ce travail réside dans les méthodes développées pour l'extraction et la modélisation de lignes de tuyauterie. Ces méthodes se caractérisent par le concept de propagation directionnelle. Une autre caractéristique importante de ces méthodes est qu'elles s'appuient fortement sur l'ajustement de primitives géométriques spécifiques. En particulier, les algorithmes développés font intervenir des primitives contraintes, ce qui est à ce jour encore rare dans la recherche en vision [3]. Les outils développés dans ce contexte ont également mené à considérer des problématiques plus générales que le strict cadre des lignes de tuyauterie. En particulier, des méthodes originales ont été développées pour effectuer l'extraction de primitives géométriques d'un nuage de points 3D. Enfin,

[3] De telles primitives contraintes existent dans le logiciel 3Dipsos, mais y sont spécifiées manuellement.

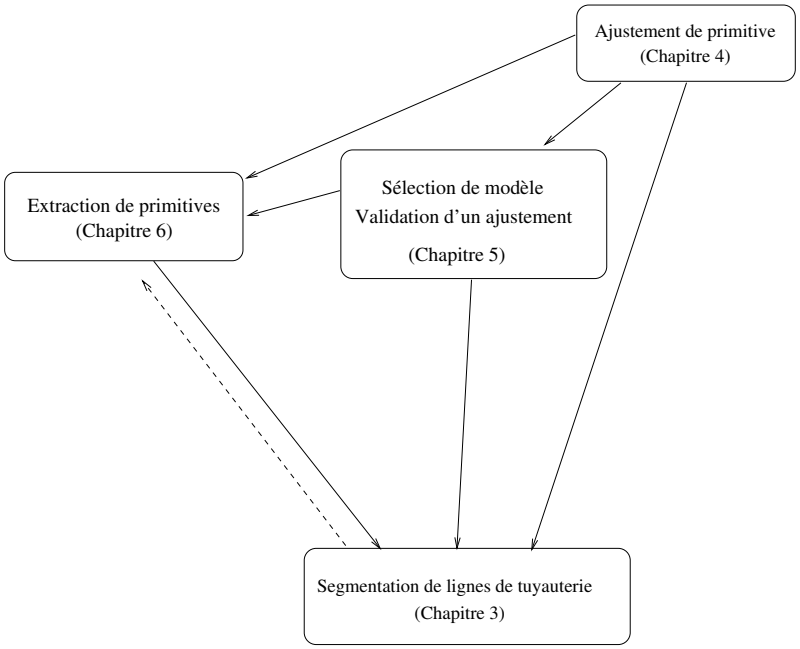


FIG. 1.13 – Interaction entre les thèmes traités dans les différents chapitres. Un lien $i \rightarrow j$ signifie que le thème développé au chapitre i est utilisé dans le chapitre j.

des procédés de sélection et de validation d'ajustement de primitives géométriques ont été construits pour traiter le problème de la reconnaissance de surface.

Chapitre 2

Etude bibliographique, état de l'art

Dans ce chapitre est présentée une étude bibliographique et de l'état de l'art des domaines qui gravitent autour de notre sujet. Plus précisément, les domaines concernés sont la vision par ordinateur, la reconnaissance de formes, la modélisation 3D, la géométrie algorithmique. Un petit nombre de travaux s'intéressent à des données d'environnements industriels, en photogrammétrie [VT00], ou en télémétrie laser [HHJJ95]. Les environnements numérisés de grande taille sont encore à l'heure actuelle relativement rares au sein des laboratoires de recherche.

En particulier, la plupart des travaux à ce jour sont caractérisés par les points suivants :

- un seul objet est présent dans la scène, sur un fond simple (typiquement, un plan),
- la scène est de petite taille (par rapport au scanner), ex : images du NRC [RC88], de Michigan State University (figure 2.1),
- la densité de points est relativement homogène (point lié au précédent).

Ce domaine subit cependant une évolution rapide, étant donné le nombre grandissant de scanners longue distance disponibles sur le marché [Gal01].

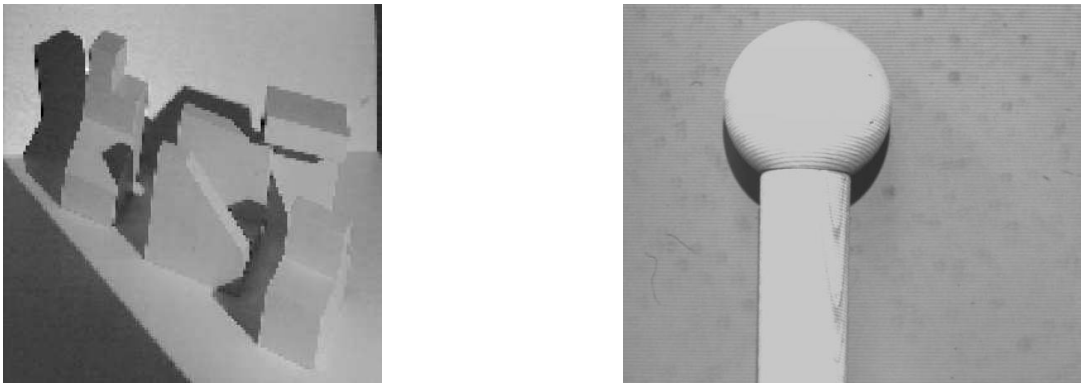


FIG. 2.1 – Scènes 2,5D : «abw» et «balljoint»

D'autre part, si un nombre important — et croissant — de publications traitent de données issues de scanners 3D, peu concernent véritablement le cas «3D» ; en effet, même si «3D» y apparaît, la plupart des articles considèrent le cas d'une image «2,5D».

Définition(Image 2,5D)

Une image 2,5D, ou $2\frac{1}{2}D$, est l'équivalent d'une image vidéo dans laquelle le niveau de gris de chaque pixel (x,y) serait remplacé par une altitude z. La particularité de ce type de données réside donc essentiellement en une structure de grille (x,y) et en la possibilité de décrire la scène comme un graphe de fonction $z = f(x,y)$.

Les expressions image de profondeur(*range image*), carte de disparité (*depth map*), désignent des notions équivalentes.

> Remarque
> Le terme 2.5D provient du fait que cette situation est un intermédiaire entre les cas 2D et 3D. Rien à voir avec les dimensions fractales !

> Remarque
> On traite généralement le cas d'une vue de scanner comme une image 2,5D. Ceci est tout à fait rigoureux lorsque l'on a un scanner qui mesure, pour chaque position (x,y), une valeur de distance suivant une même direction z (double balayage linéaire). Ceci pose par contre des problèmes lorsque le scanner effectue un balayage angulaire et produit ainsi une projection centrale (comme c'est le cas des scanners commercialisés par MENSI). En effet, la grille correspond dans ce cas à un rectangle non pas tracé sur un plan mais tracé sur une sphère, ce qui une fois projeté sur un plan ne produit plus un rectangle. Il est alors nécessaire d'effectuer une transformation afin d'obtenir une image 2,5D régulière approchée à partir du scan initial (voir par exemple le code de UB CurveSegmenter qui produit une image 2,5D par une sorte d'interpolation et [BJ88, p.189]).

Cette étude bibliographique est organisée comme suit : dans un premier temps sont présentés les traitements de données 3D dits «bas niveau», c'est-à-dire qui ne font pas intervenir de modèles de surfaces ; ensuite, sont présentés les traitements de données 3D qui font explicitement intervenir la connaissance des modèles.

En parallèle avec l'étude bibliographique, les systèmes existants suivants ont été testés :

- *Hough / UpWrite* de University of Western Australia,
- *Crust* de University of British Columbia et U. Austin Texas,
- *Segmentor* de Univ. Ljubljana,
- *UB (CurveSegmenter)* de l'Université de Bern ,
- la fonction auto-segmentation des logiciels commerciaux CGP v2.01 et Cyclone (*Cyra*),
- *TensorVoting* de University of Southern California.

Ces systèmes sont mentionnés et classés par types de méthodes dans la suite, et sont également répertoriés dans un tableau récapitulatif à la fin de ce chapitre.

2.1 Traitements «bas niveau» des données 3D (sans modèles)

Les traitements sans modèles de surface ou d'objets sont ici présentés sous trois aspects : la structuration de nuage de points et la reconstruction 3D, l'extraction d'information géométrique sur les points, et la classification - segmentation des données 3D.

2.1.1 Structuration du nuage de points, reconstruction 3D

La structuration permet de rendre les méthodes de segmentation qui suivent plus simples, plus rapides, plus efficaces,... Les structures généralement introduites sont de plusieurs types : partitionnement en grille, maillage et triangulation, croûte, α-forme, arbre, graphe, squelette, etc.

Partitionnement en grille

L'espace et le nuage de points peuvent être divisés en quadrillage régulier, ce qui donne les baquets (*buckets*) [Gou97], ou, lorsque la taille est petite, les voxels [Alg95]. Il est également possible de s'adapter un peu plus aux données par des partitions adaptatives de type *quadtree*(2D), *octree*(3D) [YS99, WSI98, PDH+97], ou *k-d trees* . Le principe de ces structures est de prendre en compte la densité de points pour créer des cellules plus petites dans les régions denses. L'*octree* préserve une régularité dans la structure mais avec des nombres de points différents dans chaque cellules, alors que le k-d tree divise chaque cellule en laissant le même nombre de points de chaque côté.

Ces structures ne donnent pas vraiment d'indication sur les surfaces présentes, mais peuvent toutefois être utiles pour accélérer l'accès aux points d'un nuage. Ceci s'avère en particulier intéressant pour optimiser les algorithmes faisant intervenir la recherche de voisins. Bien entendu, ceci requiert de redéfinir des algorithmes qui exploitent ces structures.

Maillages, triangulations

Tout d'abord, on peut effectuer une triangulation 3D des points (c'est-à-dire produire des tétraèdres). La triangulation de Delaunay (2D, 3D ou nD) constitue désormais un classique en géométrie algorithmique [PS85, BY95, Lem97]. Dans notre cas, les points se trouvant sur des surfaces, nous nous intéressons davantage aux méthodes de maillage surfacique.

Suivant [Bes99], on peut classer les différentes méthodes de construction de maillage surfacique suivant les types de données au départ :

- maillage sur une image 2,5D [GB96] ,
- fusion de maillages de plusieurs vues 2,5D [WSI98, HI97, HSIW96, MY95, SL95, RAS97],
- maillage à partir de points 3D [Alg95, HDD+92, ABK98, ST92, GM97, Guy95, TM98].

En ce qui concerne la construction de maillage à partir de points 3D, la méthode de Hoppe et al. [HDD+92] est une référence fréquemment citée et employée de construction de maillage à partir de points 3D. Notons que cette méthode a été appliquée à des scènes où les points sont sur une seule surface, fermée, et que les points sont uniformément répartis sur cette surface (Figure 2.2). Ce sont des conditions idéales, rarement vérifiées en pratique, pour lesquelles la plupart des algorithmes de construction de maillages produisent un bon résultat [ABK98]. On peut s'attendre à ce que cette méthode ne donne pas les résultats escomptés lorsque la surface est ouverte et que la densité de points n'est plus uniforme.

D'autre part, la triangulation de Delaunay en 2D peut s'étendre simplement au cas du maillage de surfaces en 3D du moment que l'on a une indication de la topologie de la surface. En effet, si l'on sait que la surface peut se projeter sur un plan, et que ce plan est connu, il suffit de réaliser une

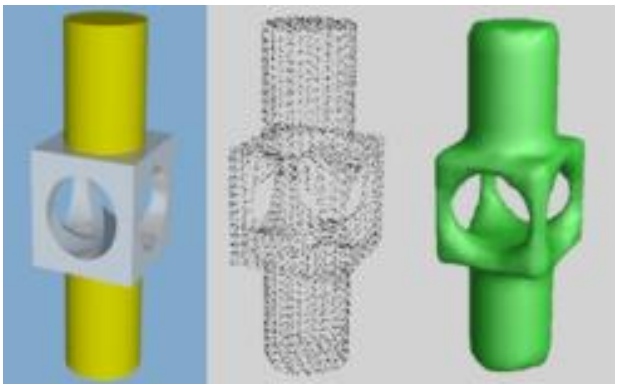


FIG. 2.2 – Triangulation par l'algorithme de Hoppe et al.[HDD+92]

triangulation de Delaunay sur le plan que l'on projette sur la surface. C'est le principe de construction de maillage utilisé dans plusieurs logiciels, dont 3Dipsos et Imageware Surfacer (EDS). Notons que l'on peut faire de même avec un cylindre, une sphère, etc. Ceci suppose que l'on connaisse le type de surface, ce qui n'est bien entendu pas toujours le cas au départ ! Remarquons également que l'on peut construire une triangulation surfacique en prenant comme surface de projection le plan tangent au nuage de points (avec le centre d'inertie). Ici intervient alors la taille du voisinage utilisé (soit une taille fixe, soit un nombre de voisins).

Croûte d'un nuage de points

Dans [ABK98] le diagramme de Voronoï et la triangulation de Delaunay servent à produire une triangulation surfacique, la «croûte» (*crust*). Cette représentation n'est pas nécessairement connexe, et permet donc de réaliser une sorte de segmentation de la scène (en 2D et en 3D). C'est pourquoi il y est également fait référence dans la section 2.1.3, avec plus de détails.

α-formes

Les maillages permettent ensuite d'autres types de représentations, comme les α-*shapes* [LGS99]. L'intérêt des α-*shapes* est de produire un maillage suivant mieux la forme du nuage de points. La faiblesse de ce type de représentation est qu'il faut trouver la bonne taille (α), et que cette taille est constante sur toute la scène.

Un maillage permet également des représentations de plus haut niveau telles que les surfaces de subdivision [HDD+94].

Arbres, graphes, squelettes...

Le squelette est souvent défini sur des images 2D, via des opérations de morphologie mathématique. Mais on peut également définir un squelette d'un ensemble de points à partir de la triangulation de Delaunay (au moins en 2D) [CHW96, BTG95] .

L'arbre d'escarpement minimal («*minimum spanning tree*») est une autre structure de type arbre qu'il peut être intéressant de construire à partir d'un nuage de points [CB00].

2.1.2 Extraction d'information sur les surfaces

Les informations géométriques classiques — le plus souvent basées sur des opérateurs différentiels — que l'on peut extraire des données 3D peuvent servir à une classification ou une segmentation sans modèle de la scène.

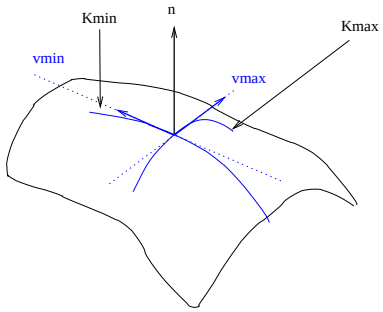


FIG. 2.3 – Normale et courbures locales à une surface en un point

Normales

La normale à une surface en un point est l'information locale la plus simple pour représenter la forme de la surface (Figure 2.3). Une définition plus précise de cette notion est présentée à l'annexe B, page 205.

La normale à la surface en un point est la plupart du temps estimée grâce au plan d'inertie des points dans un voisinage du point considéré (voir annexe B). Lorsque le voisinage n'est pas trop important par rapport aux dimensions de la surface considérée, le plan d'inertie fournit une bonne approximation du plan tangent et donc de la normale. Ceci dépend bien entendu aussi de la densité et du bruit sur les points.

Courbures

Après la normale, les informations locales qui sont naturellement introduites pour décrire la forme de la surface sont les courbures (Figure 2.3). On trouve les courbures moyennes et gaussiennes (notées H et K), ainsi que les courbures principales (notées K_{min} et K_{max}, ou K_1 et K_2). Une définition précise de ces quantités est présentée à l'annexe B, page 205.

De même que les normales, les courbures sont des quantités invariantes du modèle utilisé pour décrire la surface. Elles ne dépendent donc pas, en théorie, de la représentation utilisée pour les estimer. Si les normales sont toujours estimées à peu près de la même façon, il existe plusieurs variantes pour l'estimation des courbures (voir par exemple [MV97, Sav00, SOR99]).

- Pour une image 2,5D, il est simple d'estimer les courbures puisqu'un opérateur de convolution permet d'estimer les dérivées partielles [BJ88, Dav92].
- Sur un maillage 3D, les courbures peuvent être estimées à partir des angles formés par les normales voisines [FFE97, SPPH99]. Garcia [GB96] introduit l'«idée» de courbure par la variance des angles de normales de triangles voisins.
- Pour un nuage de points 3D, quelques unes des méthodes existantes sont mentionnées en annexe B, qui présente en détail celle que nous avons utilisée.

Remarquons que cette estimation est en général fortement bruitée, et nécessite le plus souvent un lissage pour pouvoir être utilisée [TF95]. Un tel lissage peut être effectué en 2,5D par un filtre de convolution [BJ88]. Notons ici la nécessité d'un lissage qui préserve les discontinuités (de profondeur et d'orientation), d'où l'utilité de filtres adaptatifs ou multi-résolution [Bou94a].

De même que pour la normale, il est nécessaire de définir une taille de voisinage (a priori un peu plus grande que pour les normales). Ce choix de taille est critique dans l'estimation des courbures, car les valeurs obtenues peuvent varier beaucoup selon la taille. Il n'existe pas de manière simple de définir une taille appropriée.

> Remarque
> Certains auteurs ont remarqué qu'ils ne souhaitaient estimer que le signe de la courbure (gaussienne), et que cette estimation pouvait être plus simple et meilleure que l'estimation de la valeur de la courbure elle-même [AW98].

Sphère et image de Gauss

La normale à la surface en un point étant choisie unitaire, ce vecteur peut se représenter sur la sphère unité, appelée *sphère de Gauss*. L'*image de Gauss* (*Gaussian image*) d'une surface ou d'un objet est la représentation des normales de cette surface ou de cet objet sur la sphère de Gauss [dC76, chap. 3], [Hor86, chap. 16]. Le terme carte d'aiguilles (*needle map*) désigne la même notion [Bah97].

Nous étendons la notion d'image de Gauss d'une surface au cas d'un nuage de points. On peut de la même façon représenter les normales unitaires estimées sur un nuage de points sur la sphère de Gauss, comme l'illustre la figure 2.4. L'intérêt principal de cette représentation est qu'elle est

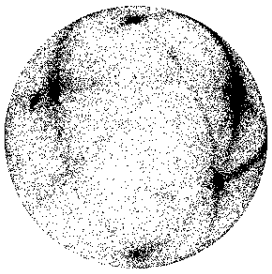


FIG. 2.4 – Image de Gauss d'une scène réelle

invariante par changement d'échelle et translation. De plus, elle est stable par rotation, en ce sens que la représentation subit la même rotation en 3D que la surface elle-même.

Une variante de cette représentation est l' «image de Gauss étendue» ou *EGI (extended Gaussian image)* : l'image de Gauss étendue est une image de Gauss où chaque vecteur est non plus unitaire mais affecté d'une longueur égale à l'inverse de la courbure de Gauss au point considéré.

> Remarque
> La courbure de Gauss en un point est la limite lorsque la taille du voisinage autour de ce point tend vers 0 du rapport de l'aire sur la sphère de Gauss et de l'aire sur l'objet [Hor86].

L'image de Gauss étendue a pour intérêt d'être une représentation bijective (à translation près) pour les objets convexes. Cependant, des méthodes permettant de retrouver l'objet à partir de l'image de Gauss étendue existent seulement dans le cas où cet objet est polyédrique [Hor86, p. 371].

Dans la littérature, l'image de Gauss étendue a été utilisée sous une forme discrétisée, la sphère de Gauss étant approchée par un maillage [Shi87, p. 207], [Hor86].

Enfin, il existe des variantes de ce type de représentation, comme par exemple la *SAI (spherical attribute image)* [HID95, Del94].

Centres de courbure d'une surface

On peut étendre la notion de centre de courbure d'une courbe de $\mathbb{R}^2$ à une surface de $\mathbb{R}^3$ [FLW93].

Ainsi, dans [Gou97], le centre de courbure est défini à l'aide de la normale et de la courbure maximale : c'est le point se trouvant à $K_{\max}^{-1}$ du point courant suivant la direction de la normale au point (orientée par le repère de Darboux)[Gou97, p.59].

Pour des surfaces de type cylindre, cône, ou tore, la notion intuitive de centres de courbure correspond à des points qui se trouvent sur l'axe (pour le cylindre et le cône), ou le cercle directeur(pour le tore). Ce fait est utilisé dans [Gou97], mais aussi dans les procédés de la section 4.4 utilisant les centres de courbure. Notons toutefois que la définition faite précédemment à partir de $K_{\max}^{-1}$ ne correspond pas toujours à cette notion intuitive. En effet, dans le cas du tore de petit rayon r et de grand rayon R où $r \leq R \leq 2r$ (ce qui est un cas tout à fait courant en pratique), les points sur le cercle directeur ne sont pas tous à $K_{\max}^{-1}$ du point de la surface : certains sont à $K_{\min}^{-1}$. Le passage entre les deux zones se fait lorsque $K_{\min} + K_{\max}$ change de signe. Ceci est donc lié au signe de la courbure moyenne H en ce point. Le tableau présenté à l'annexe B.2, page 206 indique que le changement de signe de H a lieu lorsque

$$\cos v = \frac{R}{2r}$$

La figure 2.5 illustre les zones sur lesquelles la courbure moyenne H a un signe distinct. Ceci a pour effet de créer, dans la partie intérieure du virage, des centres de courbures qui se trouvent au centre du tore et non sur le cercle directeur (Figure 2.5).

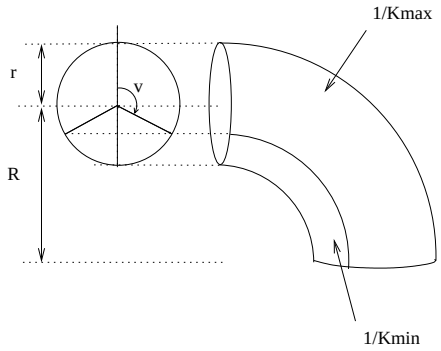


FIG. 2.5 – Extension au cas 3D de la notion de centre de courbure, cas particulier du tore.

Quoiqu'il en soit, estimer le centre de courbure local revient à estimer la normale et la courbure locales. Le caractère bruité, en particulier de l'estimation de la courbure, se reporte spatialement sur l'estimation du centre de courbure.

Remarques

- Les centres de courbure d'un plan ou d'une surface localement plane ne sont pas définis (ils se trouvent à l'infini).

– Les centres de courbure de surfaces de révolution se trouvent sur l'axe de révolution (ceci ne permet donc pas, par exemple, de distinguer directement un cône d'un cylindre).

Autres

D'autres types de description de la forme locale existent : moments [GM93, Ald94], *spin-images* (Système ARTISAN de Carnegie Mellon University) [JH97, JHOH97], etc.

2.1.3 Classification, segmentation sans modèles

Dans cette partie sont présentées les méthodes de classification et de segmentation respectivement sur une ligne de scan («1,5D»), en 2D, sur une image de profondeur (2,5D), et sur un nuage de points 3D.

Sur une ligne de scan

Le cas le plus simple est celui où l'on traite une liste ordonnée de points 2D formée par une ligne de scan. Par analogie avec le terme «2,5D», on peut nommer ce cas «1,5D».

Il existe des techniques de segmentation de courbes (*curve partitioning*) en segments ou en arcs de cercles [RW95], ou encore avec des ellipses [Fit97].

L'une des techniques utilisées est une approche récursive faisant intervenir la notion de corde. A la fin, une ligne polygonale approche la courbe à un écart donné près (au sens de la distance perpendiculaire).

Des méthodes de ce type ont été employées pour la segmentation de lignes de scan (algorithme UB [HJBJ+96, PBJB98, JB94]), ou plus généralement de coupes planes d'un objet numérisé [Ost02, Ost95b, Ost95a, CLB95, Nat98].

En 2D

«Croûte» Comme nous l'avons vu précédemment, la croûte est une représentation issue de la triangulation de Delaunay et du diagramme de Voronoï [ABK98]. En 2D, ceci permet de produire une sorte de segmentation des points. Voir la démonstration en ligne sur les sites Internet de University of Texas à Austin et de University of British Columbia, dont une illustration est donnée figure 2.6 (l'utilisateur ajoute des points en cliquant, et le squelette et la «croûte» sont immédiatement actualisés). Dans [ABK98], cette construction est étendue au cas 3D. Nous avons implémenté et testé cet algorithme sur des données de l'article [HDD+92], ce qui a donné de très bons résultats (comparables à la méthode de [HDD+92]), ainsi que sur des scènes réelles, ce qui donne des résultats un peu plus incertains — en particulier, les zones qui sont reliées (Figure 2.7). Notons que des différences significatives existent entre les cas 2D et 3D concernant la construction de cette structure (squelette, pôles).

Regroupement La première étape de l'algorithme *UpWrite* [MA98, MA97, Ald94] consiste en un regroupement par morceaux (*chunking*) suivant un procédé multi-résolution, utilisant la matrice de variance-covariance locale. Le programme, dont une illustration est donnée figure 2.8, est téléchargeable sur le site Internet de U Western Australia.

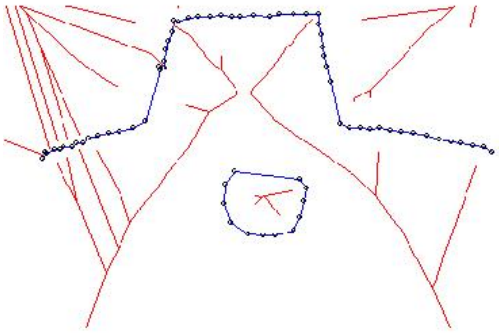


FIG. 2.6 – *Crust*. Programme de U British Columbia

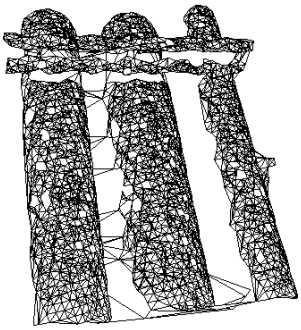


FIG. 2.7 – La croûte sur une zone de scène réelle

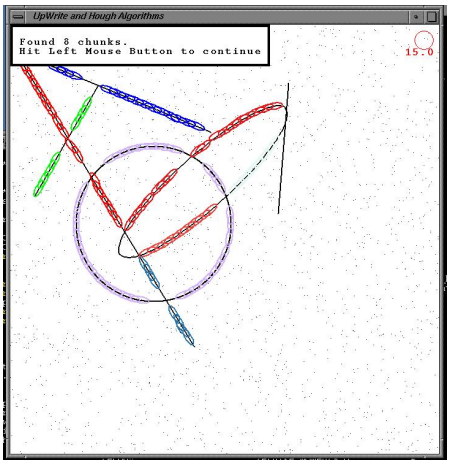


FIG. 2.8 – UpWrite. Programme de U Western Australia

Sur une image de profondeur

Comme nous l'avons dit au début de ce chapitre, le cas «2,5D» correspond au cas d'une image de profondeur, c'est-à-dire à peu près à une vue de scanner. Du fait de la structure en grille, le cas 2,5D se rapproche des techniques désormais classiques de segmentation d'images 2D. Les auteurs distinguent en général deux classes de méthodes : les approches par contours (*edge-based*) et les approches par régions (*region-based*) (et éventuellement les méthodes hybrides, utilisant les deux approches précédentes) [FEF97, VMJ97].

Approches par contours Les contours sont déterminés par la détection des discontinuités de profondeur [Bou94a, BC94a] et d'orientation. La fonction «auto-segmentation» du logiciel Cyra CGP version 2.01 fonctionne sur ce principe. D'autres travaux ont fait intervenir les discontinuités d'ordres supérieurs [Dav92]. Finalement, les contours séparent des zones qui présentent une certaine homogénéité d'orientation (il peut être utile d'effectuer des opérations morphologiques afin de dégager des zones connexes). Ces approches sont intéressantes en particulier pour leur rapidité.

A titre d'exemple, la fonction «auto-segmentation» de Cyra CGP v2.01 détermine d'abord les discontinuités en profondeur et en orientation sur la grille 2,5D. Ensuite, une opération de morphologie mathématique génère des zones «connexes» (les parties de moins de 6×6 pixels sont ignorées). La fonction met environ 30 s (sur PC Pentium II) pour segmenter une scène de 100000 points.

Dans le logiciel Cyra Cyclone (nouveau logiciel de la même société), le principe semble être identique, sur chaque vue. Ensuite, il y a une sorte de fusion des segmentations effectuées dans les vues respectives. Notons que les surfaces détectées ici sont déconnectées les unes des autres.

Cependant, les approches par contours sont souvent considérées comme moins robustes que les approches par régions.

Approches par régions

Le principe des approches par régions est de regrouper les points qui présentent une certaine similarité.

Classification à partir des courbures

- A partir des signes des courbures (H,K)
  On procède à une classification de chaque point à partir des signes de (H,K), en : «plan, hyperbolique, elliptique, cylindrique» (algorithme BJ [BJ88, PBJB98], algorithme UE [HJBJ+96, TF95] ; [Bou99]). Trucco [TF95] fait la remarque que la méthode (H,K) est peu adaptée aux plans.
- Classification à partir des courbures (K_{min},K_{max})
  Classification de chaque point à partir des valeurs de (K_{min},K_{max}) en «ombilic, parabolique, plan» [MV97] ou en «plan, courbe» [GB96].

Segmentation par regroupement (*clustering*)

Regroupement dans un espace de paramètres «bas niveau» (normales + points) dans $\mathbb{R}^6$: algorithme WSU de [HJBJ+96], dans $\mathbb{R}^4$: [GM93](avec apprentissage, à l'aide d'un réseau de Kohonen). Pour les méthodes de *clustering*, voir [CM98].

Sur un nuage de points 3D

La méthode de R. McLaughlin et Alder décrite ci avant a également été appliquée au cas 3D (cf. [McL00]), pour l'extraction de courbes (chaînettes) ou de surfaces.

Le maillage 3D permet de faire à peu près comme dans le cas 2,5D : on peut estimer (même grossièrement) les normales, les courbures, etc. et ainsi faire une classification des points suivant ces informations. Aussi la plupart des auteurs utilisent-ils, plus ou moins explicitement, une structure de maillage au préalable de la phase de segmentation proprement dite.

Hoppe et al. effectuent d'abord une construction de maillage avec la méthode introduite dans [HDD+92], et dont il a été fait référence précédemment (adaptée au cas d'un seul objet). Ensuite, Hoppe et al. [HDD+94] construisent une fonction lisse par morceaux ainsi que des arêtes et des jonctions à l'aide de surfaces de subdivision.

Fisher et al. [FFE97] utilisent la même méthode pour construire un maillage. Ensuite, les angles entre normales voisines sont estimés afin de classer les zones élémentaires en «plan, arête et courbe». Un exemple de scène segmentée est donné figure 2.9 (noter l'absence de segmentation entre cylindre et plan latéraux lors de la première phase).

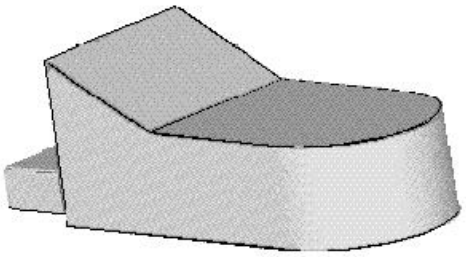


FIG. 2.9 – Pièce «BAe»(U Edinburgh)

Wu et Levine [WL95, WL97] simulent la répartition d'une charge électrique sur le maillage pour dégager les minima de convexité, qu'ils utilisent pour segmenter un objet en parties convexes (*part segmentation*). Les scènes utilisées dans les exemples donnés par les auteurs sont cependant relativement simples.

Chaine et Bouakaz [Cha00, CB00] estiment les normales et dégagent des zones connexes par rapport à leurs orientations. Ceci fournit une segmentation initiale. Celle-ci est complétée par une méthode de fusion de régions (Figure 2.10). Appliquée aux images de [HJBJ+96], cette méthode a donné des résultats comparables et s'est avérée plus rapide que UE, WSU et USF, moins rapide que UB.

Enfin, les méthodes de décomposition d'un nuage de points en composantes δ-connexes [Gou97, PTVF92b] ou en groupements (*clusters*) [CM98] peuvent aussi être considérées comme des méthodes de segmentation particulières.

Dans [Gou99, Gou97], l'estimation des centres de courbure permet de passer du problème de la segmentation entre cylindre et tore telle que sur la figure 2.13 au problème de la segmentation des centres de courbures en segments de droite et cercles. Cette segmentation est faite à l'aide de la valeur de la courbure minimale : les centres correspondant au cylindre ont une courbure minimale nulle. Notons toutefois que la courbure minimale seule n'est pas idéale pour faire cette segmentation car des points sur le tore peuvent aussi avoir une courbure minimale nulle [Gou97, p. 67]. Cependant, une

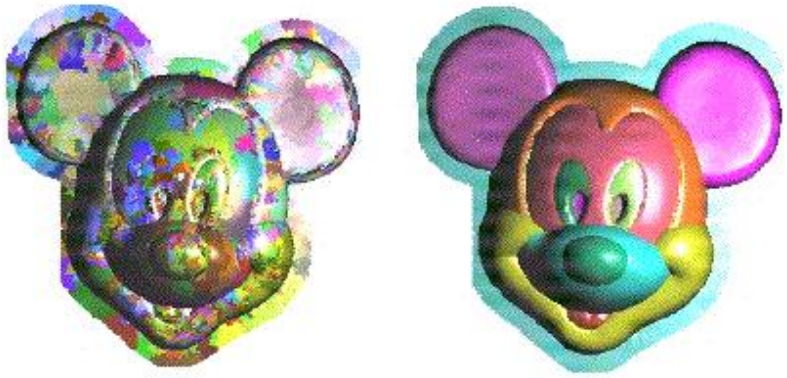


FIG. 2.10 – Segmentation par classification initiale (à gauche) et fusion. Sur le résultat (à droite), chaque grande zone a une couleur : le nez, chaque œil, chaque oreille, etc. (U Lyon1)

difficulté de cette approche réside à mon avis dans l'estimation des courbures (bruitée et lente).

La méthode de Medioni et al. [GM97, Guy95, TM98] permet de construire, en même temps que les surfaces les plus probables, les courbes les plus probables étant donné l'ensemble (peu dense) de points. Aussi peut on trouver les arêtes en même temps que le maillage de l'objet (dans l'exemple de la figure 2.11, la méthode permet d'extraire les points correspondant à la jonction entre la cacahuète et le plan).

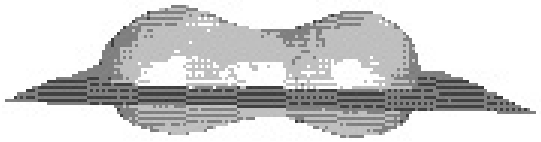


FIG. 2.11 – Tensorvoting de U Southern California. Reconstruction de surfaces et de jonctions

Un test du programme TensorVoting, téléchargé depuis le site de U Southern California, a montré que le temps d'exécution croît rapidement avec le nombre de points initiaux : sur un nuage de 5000 points («pipe.dat») comprenant deux cylindres concentriques (Figure 2.12), la reconstruction des surfaces des deux cylindres a pris 7 min environ (résultat : 28210 triangles) sur PC Pentium III.

D'autre part, ce programme a été testé sur la scène «coude» (Figure 2.13), avec également 5000 points. Notons tout d'abord que cette scène ne peut pas être segmentée par cette méthode, car la jonction entre les deux primitives est lisse. Par contre, il paraissait intéressant de tester la reconstruction de la surface formée par le cylindre et le tore. Les surfaces reconstruites par l'algorithme, qui apparaissent à droite de la figure 2.14, ne forment clairement pas une représentation correcte du coude.

Pourquoi ce résultat ? L'algorithme fonctionne bien lorsque, comme dans la scène «pipe.dat» (fi-

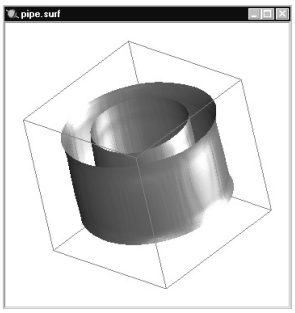


FIG. 2.12 – TensorVoting de U Southern California. Scène «pipe.dat»

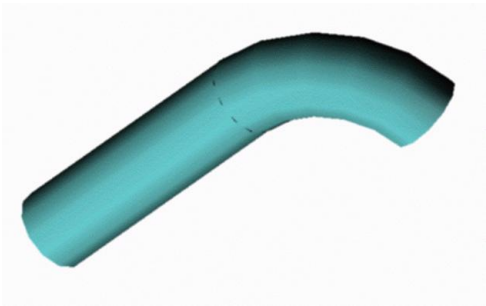


FIG. 2.13 – Jonction lisse entre un tore et un cylindre

gure 2.12), les points sont uniformément répartis sur la surface. Les points peuvent être épars, il peut y avoir du bruit, mais la densité en points doit être à peu près la même partout sur les surfaces. Or, comme cela a été dit au chapitre 1, ce n'est clairement pas le cas pour nos scènes : le coude de la figure 2.13 a été scanné par un, voire deux points de vue de scanner. Il y a donc une grande partie du cylindre et du tore qui est cachée au niveau du nuage de points.

Remarque
Comme remarqué dans [FFE97], les situations comprenant une jonction lisse (par exemple C^2) entre deux primitives restent intraitées par la plupart de ces méthodes : voir par exemple la pièce «BAe» tirée de [FFE97] (figure 2.9) et l'image «coude» (Figure 2.13).

Il semble par conséquent que ce genre de scènes, par ailleurs fort courantes dans la réalité, mène à considérer des procédés de segmentation de plus haut niveau, dans lesquels interviendraient plus directement les modèles des primitives recherchées.

Plus précisément, la segmentation utilisée pour la pièce de la figure 2.9 utilise des modèles de surfaces, mais ceci ne suffit pas : sans doute faut-il faire intervenir les modèles plus tôt dans le processus de reconstruction-segmentation.

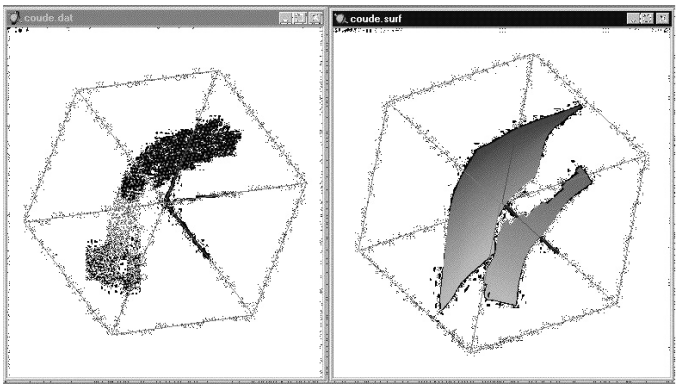


FIG. 2.14 – TensorVoting de U Southern California. Résultat sur la scène «coude» (5000 points). A gauche : les points. A droite : les surfaces reconstruites

2.2 Traitements des données 3D à base de modèles

Les modèles 3D les plus couramment utilisés sont ici brièvement présentés. Ensuite, les traitements des données 3D faisant intervenir de tels modèles sont décrits, par ordre de complexité : ajustement , reconnaissance, extraction, segmentation à base de modèles, et créations de modèles complexes.

2.2.1 Différents types de modèles

Surfaces $z = f(x,y)$ polynomiales

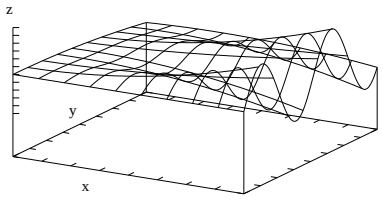


FIG. 2.15 – Surface z= f(x,y)

Les articles traitant de la segmentation purement 2,5D utilisent le plus souvent ce type de modèle, qui exploite justement la structure de grille (Figure 2.15).

> Remarque
> Les surfaces du type $z = f(x,y)$ avec f polynôme de degré au plus 2×2 sont parfois appelés «quadriques». Dans ce document, on utilisera plutôt le terme de «surface biquadratique» pour ce type de surface, et le terme de quadrique pour dénoter les quadriques implicites.

Parmi nos primitives, seul le plan peut être exactement représenté par ce type de surface. Toute surface suffisamment lisse peut certes être approchée localement par de telles fonctions, dans un repère local, par un développement de Taylor. Cependant, cette approximation locale ne fournit pas

forcément une bonne approximation au niveau global (sans parler des problèmes de topologie liés au fait que l'on ne peut pas décrire la totalité d'une telle surface par une fonction $z = f(x,y)$). Aussi paraît-il étonnant que dans divers travaux, on se propose de segmenter des scènes où se trouvent des cylindres, des sphères, des cônes ou des tores (ex. images du NRC ou de Michigan State University) avec des modèles $z = f(x,y)$ de degré faible [YBK94, JB94, HJBJ+96]. On peut certes s'approcher un peu plus des surfaces réelles avec des modèles de degré 4×4 [SB95, BJ88] qu'avec des modèles de degré au plus 2×2 [LGB95, DP97, YBK94, GM93]. Ceci est illustré sur la figure 2.16, où des surfaces de ce type sont ajustées sur des points se trouvant sur un cylindre (les points sont en réalité sur un demi-cylindre, de façon que l'on puisse encore les décrire par une surface $z = f(x,y)$). Dans cet exemple, tous les points ont été considérés dans l'ajustement. On voit que dans les deux cas, l'approximation globale n'est pas tout à fait satisfaisante. On constate d'autre part que le modèle de degré plus élevé est plus proche du cylindre.

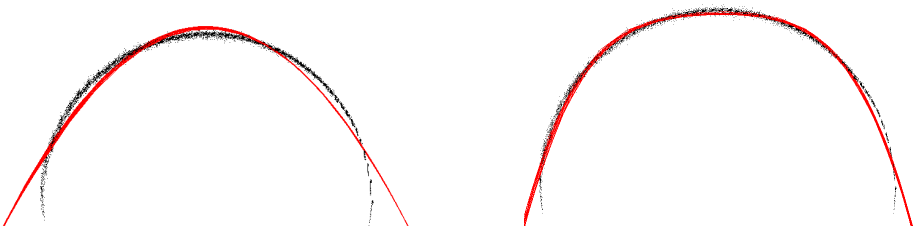


FIG. 2.16 – Ajustement de surfaces $z = f(x,y)$ polynomiales sur un cylindre, de degré 2×2 (à gauche) et 4×4 (à droite).

Surfaces $f(x,y,z) = 0$ polynomiales

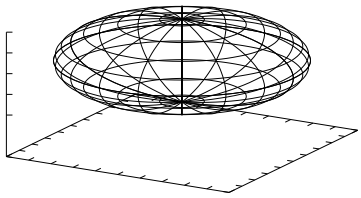


FIG. 2.17 – Surface implicite

Les surfaces algébriques — c'est-à-dire les surfaces implicites $f(x,y,z) = 0$ où f est un polynôme — forment une classe de surfaces plus vaste, contenant la classe des surfaces bivariées précédentes (Figure 2.17). En effet, $z - f(x,y) = 0$ avec f polynôme est bien un cas particulier d'équation algébrique. En particulier, toutes les primitives du chapitre 1 sont contenues dans cette classe de modèles (plan, cylindre, cône et sphère sont des quadriques implicites, le tore est une tétrique implicite). En effet, les équations de ces primitives s'écrivent dans un repère propre :

Plan	$ax+by+cz+d=0$
Sphère	$x^2+y^2+z^2-r^2=0$
Cylindre	$x^2+y^2-r^2=0$
Cône	$x^2+y^2-\tan\alpha^2 z^2=0$
Tore	$4R^2(x^2+y^2)-[x^2+y^2+z^2+R^2-r^2]^2=0$

Ces équations sont obtenues à partir des distances d'un point à chaque primitive (annexe A, page 185). Or, une fois la rotation et la translation effectuées, ces équations s'écrivent sur la base polynomiale des quadriques :

$$(1,x,y,z,x^2,xy,xz,y^2,yz,z^2)$$

pour la sphère, le cylindre et le cône, et sur la base polynomiale des tétriques

$$(1,x,y,z,x^2,xy,xz,y^2,yz,z^2,x^3,x^2y,x^2z,xy^2,xyz,xz^2,y^3,y^2z,yz^2,z^3,$$
$$x^4,x^3y,x^3z,x^2y^2,x^2yz,x^2z^2,xy^3,xy^2z,xyz^2,xz^3,y^4,y^3z,y^2z^2,yz^3,z^4)$$

pour le tore.

> Remarque
> La base polynomiale des quadriques génère un espace vectoriel de dimension 10. Les équations du plan et de la sphère peuvent s'exprimer dans des sous-espaces de dimension strictement inférieure (base $(1,x,y,z)$ de dimension 4 pour le plan, base $(1,x,y,z,x^2+y^2+z^2)$ de dimension 5 pour la sphère). Par contre, il n'y a semble-t-il pas d'espace plus simple que celui des quadriques pour exprimer les équations de cylindres et de cônes quelconques (bien que le nombre de paramètres minimal définissant ces modèles soit respectivement 5 et 6, inférieurs à 9, nombre de paramètres minimal définissant une quadrique quelconque) [LMM98]. De même que pour le plan et la sphère, la base polynomiale des tores est de dimension strictement inférieure à la base polynomiale des tétriques.

Une importante littérature est consacrée aux coniques (en 2D) et aux quadriques (en 3D) [AF96]. Un certain nombre de travaux ont notamment été menés sur l'ajustement de coniques en général, ainsi que sur des coniques ou des quadriques particulières [LC98, Zha96, Zha95, Tau91, Pra87]. Enfin, Taubin [TCS+94] propose également certaines équations algébriques de degré pair et élevé pour représenter des surfaces bornées pouvant avoir des formes complexes.

Autres types de surfaces implicites

On peut également utiliser des surfaces implicites non algébriques, c'est-à-dire des surfaces $f(x,y,z)=0$ où f n'est pas un polynôme.

Super-quadriques La super-quadrique généralise la notion de surface algébrique implicite en introduisant des puissances non-entières. L'équation d'une super-quadrique est, dans un repère propre :

$$\left[\left(\frac{x}{a}\right)^{\varepsilon_1}+\left(\frac{y}{b}\right)^{\varepsilon_1}\right]^{\frac{\varepsilon_2}{\varepsilon_1}}+\left(\frac{z}{c}\right)^{\varepsilon_2}=1$$

Ceci donne les modèles tels que les super-ellipses (2D), les super-quadriques (3D) [LJS97], les cylindres généralisés ([Pon88]).

Modèles déformables On peut également classer les courbes et surfaces de niveaux (*level set*), ou modèles déformables (par EDP) dans les surfaces implicites. On trouve dans cette catégorie les notions de contours actifs, *snakes*, ... Ces modèles sont notamment utilisés en imagerie médicale [HFG00].

Surfaces définies par des critères géométriques On peut par exemple définir les primitives vues au chapitre 1 à l'aide de caractérisations géométriques : le plan peut être défini par un produit scalaire constant, le cylindre par une distance constante (par rapport à un axe), le tore également par une distance constante (par rapport à un cercle), etc. Ceci donne lieu dans le cas général à des équations implicites non-polynomiales (voir chapitre 4 et annexe A).

> Remarque
> Ceci n'est guère employé dans la littérature. L'une des raisons est peut-être que ceci contraint à définir un type de modèle par primitive recherchée. Une autre raison est que cette approche est simple pour les surfaces simples, pour lesquelles on peut définir analytiquement la distance exacte. Toutefois, plusieurs auteurs ont souligné l'intérêt d'utiliser le plus tôt possible le type précis de primitive que l'on cherche à extraire [LMM98, LMM97, FFE97]).

Surfaces paramétrées

Le domaine de la Conception Assistée par Ordinateur (CAO) utilise abondamment des définitions paramétrées pour les surfaces, comme par exemple les *splines*, les *b-splines*, et les *NURBS*. Ces surfaces sont des surfaces d'interpolation satisfaisant certaines contraintes de régularité [Lau72]. Elles sont le plus souvent définies par des points de contrôle, sur lesquels on peut faire un ajustement [Con96]. Les splines sont parfois utilisées pour la segmentation [Lei93].

Objets catalogués

Les modèles utilisés peuvent être, comme en reconnaissance d'objets, des modèles d'objets parmi un ensemble fini et connu (c'est-à-dire un catalogue). Par exemple, le système *Artisan* de Carnegie Mellon University [JH97, JHOH97] (voir site Internet dont l'adresse est donnée en annexe) utilise des modèles de vannes.

Modèles CAO plus complexes

Parmi les modèles plus complexes, on trouve la *b-rep* (pour *boundary representation*), c'est-à-dire représentation par frontières, que l'on peut construire à partir d'une scène déjà segmentée [FEF97, HGB98b, HGB98a, HGB95] ; les modèles CSG (*constructive solid geometry*, représentation par volume et opérations ensemblistes ; ou encore des modèles CAO articulés [WFAR99, AFRW97].

2.2.2 L'ajustement de modèle

L'ajustement ou estimation de modèle est une tâche importante, dans la mesure où elle intervient la plupart du temps dans l'extraction, la reconnaissance et la segmentation de scènes à partir de modèles.

Suivant les domaines, les méthodes de ce type peuvent être nommées différemment : par exemple, l'ajustement de surfaces $z = f(x,y)$ correspond à la notion de régression utilisée en statistiques. Le terme d'approximation est également utilisé.

De manière générale, on constate que peu de méthodes ont été développées pour l'ajustement de modèles «géométriques» ; c'est-à-dire ayant une définition géométrique simple, dans le domaine de la vision. Il existe encore moins (pour ne pas dire pratiquement pas) de modèles contraints dans la recherche en vision, alors que c'est visiblement un besoin en ingénierie inverse.

Expression générale du problème

Etant donné un ensemble de points, il s'agit de déterminer les paramètres du modèle, de type connu, qui s'ajuste «au mieux» sur les points(Figure 2.18).

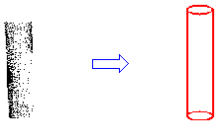


FIG. 2.18 – L'ajustement de modèle

Ajustement par moindres carrés

Le procédé le plus répandu pour traiter ce problème est la méthode des moindres carrés. Celle-ci consiste à chercher le vecteur paramètre **a** du modèle qui minimise la somme des carrés des écarts des points $\mathbf{x_i}$ au modèle :

$$\sum_{i=1}^{N} d(\mathbf{x_i}; \text{Modèle}(\mathbf{a}))^2$$

Pourquoi cette méthode est-elle aussi répandue ? Ceci est dû à plusieurs raisons :

1. elle est conceptuellement assez simple,
2. elle répond souvent efficacement et de manière réaliste au problème,
3. comme nous allons le voir, elle conduit, dans certains cas classiques, à une méthode de résolution directe (par algèbre linéaire).

Parmi les cas classiquement introduits qui donnent lieu à une résolution directe, on trouve : les surfaces algébriques, c'est-à-dire les surfaces dont l'équation s'écrit $P(x,y,z) = 0$ où P est un polynôme (dont en 2D : la droite, le cercle, la conique, et en 3D : le plan, la sphère, la quadrique), la droite 3D, les surfaces de Monge polynomiales c'est-à-dire les surfaces dont l'équation s'écrit $z = P(x,y)$ où P est un polynôme. Dans de tels cas, on montre que la résolution du problème de minimisation revient à résoudre un problème linéaire, d'inversion de matrice ou d'extraction des éléments propres d'une matrice symétrique réelle [Pra87](voir annexe C, page 213).

Lorsque la distance euclidienne au modèle ne présente pas cette propriété intéressante, on peut parfois se ramener à ce cas en utilisant des fonctions distance différentes, qui sont des approximations de la distance euclidienne. Pour les modèles qui peuvent être définis par une équation algébrique implicite, une distance classiquement introduite à cet effet est l'équation implicite du modèle, appelée «distance algébrique». Voir par exemple le *simple fit* dans [Pra87], et l'ajustement spécifique pour quadriques non-planes dans [Bou94b, p. 167]. L'intérêt est une résolution linéaire non-itérative, donc rapide. L'inconvénient est que les distances introduites n'ont pas toujours de bonnes propriétés, surtout près de singularités (près du sommet d'un cône, par exemple). Certains auteurs introduisent

TAB. 2.1 – Tableau récapitulatif des méthodes existantes d'ajustement de modèles par moindres carrés

Modèle	Résolution linéaire méthode	distance	Résolution itérative méthode	distance	Références
En 2D					
Droite	v.p.(1)	exacte			
Cercle	v.p.	appr.(4)			[Pra87]
Conique	v.p.	appr.			[Pra87]
Ellipse			opt.(3)	appr.	[Fit97]
$f(x,y) = 0$ polynomiale	v.p.	appr.	opt.	appr.	[Tau91]
En 3D					
$f(x,y,z) = 0$ polynomiale	v.p.	appr.	opt.	appr.	[Tau91]
Quadrique quelconque	v.p.	appr.	opt.	appr.	[Pra87],[Tau91]
$z = f(x,y)$ polynomiale	inv.(2)	appr.			[BJ88]
Plan	v.p.	exacte			[Pra87]
Sphère	v.p.	appr.	opt.	exacte	[Pra87] cette thèse
Cylindre			opt. opt.	appr. exacte	[MLM01, LMM98] cette thèse
Cône			opt. opt.	appr. exacte	[MLM01, LMM98] cette thèse
Tore			opt. opt.	appr. exacte	[MLM01, LMM98] cette thèse

Légende : (1) : méthode d'extraction de valeurs-vecteurs propres de matrices symétriques réelles, (2) : méthode de calcul d'inverse d'une matrice symétrique réelle, (3) : méthode d'optimisation (en général Levenberg-Marquardt), (4) : approximation de la distance euclidienne, ou exacte.

des fonctions distance plus évoluées (par exemple en normalisant par le gradient de la forme implicite [Tau91]), mais perdent ainsi le caractère quadratique en les paramètres, et sont ramenés à une optimisation non-linéaire.

Hormis les cas les plus simples (point, droite, plan, ...), la distance d'un point à un modèle n'est pas linéaire en les paramètres de ce modèle, et conduit donc la plupart du temps à un problème d'optimisation non-linéaire. C'est le cas en particulier pour toutes les primitives du chapitre 1 à part le plan et la sphère.

D'autre part, même si l'ajustement n'utilise pas l'expression exacte de la distance, l'expression approchée mène le plus souvent à un problème non-linéaire [Tau91, TCS+94, LMM98, LMM97].

Il existe différentes méthodes d'optimisation non-linéaire pouvant être utilisées ici (par exemple, méthode de Newton comme c'est le cas pour l'ajustement de cylindre dans 3Dipsos). L'algorithme de Levenberg-Marquardt [PTVF92b] est une méthode utilisée dans la littérature pour ces situations [RFWA99, Tau91].

Le tableau 2.1 présente une synthèse des méthodes existantes d'ajustement de moindres carrés des surfaces les plus courantes.

Citons au passage l'apparition récente des méthodes d'optimisation par analyse par intervalles, qui peuvent garantir de trouver un optimum global et ne nécessitent pas de solutions initiales (pour

une description et une application en infographie voir [Sny92], pour une application au recalage de maillages voir [PDH+97]).

Notons enfin que les méthodes de moindres carrés se déclinent en diverses variantes, comme par exemple les moindres carrés pondérés [Kan93, p. 336].

Autres types d'ajustement

D'autres types d'ajustement que les moindres carrés sont possibles. En particulier, il existe des méthodes plus robustes que les moindres carrés, qui peuvent être intéressantes en présence de points aberrants (surface voisine, petits objets, bruit, ...).

M-estimateurs La méthode des moindres carrés minimise la somme des carrés des résidus (différence entre le point réel et son estimation). Les méthodes des M-estimateurs minimisent la somme de fonctions des résidus moins croissantes que le carré, ce qui donne un critère moins sensible aux points aberrants [BMG94]. Dans ce cas, on peut toutefois ramener le problème d'optimisation à un problème de moindres carrés itérativement pondérés [Zha95].

Méthodes de Monte-Carlo (à base d'échantillons aléatoires) De même que l'on définit un plan par 3 points, il est possible de définir des surfaces plus générales à partir d'un nombre réduit de points. Pour les surfaces algébriques, on trouve les paramètres du modèle à l'aide d'un déterminant de Gram [Pra87](voir annexe D, page 219). Ceci a été utilisé pour extraire des ellipses (2D), des plans et des sphères (3D) [RL93] (avec algorithme génétique) [RL94, RL92], des plans et des surfaces bi-quadratiques [YBK94]. Une telle méthode, visant à utiliser le nombre minimal de points pour construire une surface, est décrite à la section 6.5, page 174.

2.2.3 La reconnaissance ou sélection de modèle

La nuance par rapport à l'ajustement de modèle est qu'il s'agit ici d'identifier le type du modèle, en général parmi une liste de modèles (paramétriques ou non), auquel les points appartiennent (Figure 2.19).

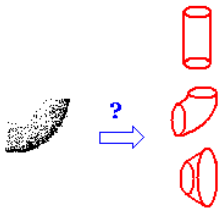


FIG. 2.19 – Sélection de modèle

La reconnaissance d'objets

En général, on calcule une représentation de la scène et on la compare aux représentations précédemment calculées pour les objets que l'on cherche à identifier dans la scène. Ces représentations peuvent être l'image de Gauss (pour reconnaître des objets polyédriques) [Hor86], sa variante la *SAI*

introduite précédemment [HID95, Del94], des *spin images* [JH97, JHOH97], ou encore des moments [WI95].

La reconnaissance de modèles paramétrés

La reconnaissance, ou plus précisément la sélection d'un type de modèle paramétré parmi une liste de modèles possibles est une question difficile, qui reste à bien des égards une question ouverte. Par exemple, distinguer entre un plan et un cylindre n'est pas simple, et ce indépendamment des moyens pour y arriver : vue de très près, une surface cylindrique «a l'air» d'être plane ! Plus précisément, le problème est de choisir un modèle parmi des modèles «emboîtés» : par exemple, une quadrique approchera toujours «mieux» un ensemble de points qu'un plan, et la quadrique trouvée sera très probablement non plane, même si les points sont en réalité sur un plan. C'est une situation de sur-ajustement (*over-fitting*) qui rend difficile l'identification du type de quadrique (plan, cylindre, etc.) en fonction des coefficients trouvés [FEF97]. Cette situation est décrite en détails à la section 5.1.

Un grand nombre de travaux, en statistiques et en vision, traitent de ce problème. Le cas le plus souvent traité se situe dans le cadre de la régression linéaire, ce qui couvre notamment le cas des surfaces $z = f(x,y)$ polynomiales.

Le critère de sélection le plus courant consiste à comparer les erreurs d'ajustement (erreurs quadratiques). Cependant, il existe de nombreux autres critères de sélection de modèles, [BS98, BS97, Aka74, JOM00, PTVF92b]. Une description plus détaillée est donnée à la section 5.1.

2.2.4 Extraction de modèle et segmentation à base de modèles

L'extraction de modèle s'apparente à l'ajustement de modèle, dans la mesure où l'on connaît le type de modèle que l'on cherche à extraire, mais ici tous les points considérés ne sont pas censés appartenir au modèle (Figure 2.20). En cela, on peut rapprocher l'extraction de modèle des mé-

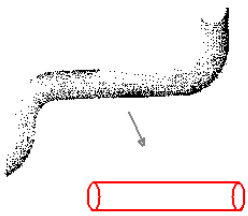


FIG. 2.20 – L'extraction de modèle

thodes robustes d'ajustement [RL93]. Un cas particulier d'extraction concerne l'extraction d'objets. Par exemple, Hebert et al. (Carnegie Mellon University) extraient des objets connus (catalogue) parmi une scène complexe [CHH99].

La segmentation à base de modèles se différencie de l'extraction de modèles en ce que la segmentation donne une description exhaustive de la scène en termes de modèles(Figure 2.21).

Dans cette partie sont présentées les méthodes d'extraction de modèles ou de segmentation à base de modèles successivement en 2D, en 2,5D et en 3D. Les différentes méthodes présentées ont été regroupées par thème : méthodes de vote, méthode par apprentissage, croissance de région, et division-fusion.

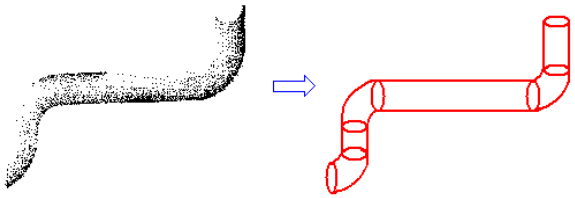


FIG. 2.21 – La segmentation à partir de modèles

Méthodes de vote

Transformée de Hough La transformée de Hough (TH) standard et ses variantes (TH généralisée, aléatoire, hiérarchique) [Dav97, CM91, Fau93] sont des méthodes d'accumulation : chaque point vote dans un espace de paramètres, et l'on extrait des instances de modèle par détection des maxima d'accumulation de votes dans cet espace de paramètres.

Le principe de la transformée de Hough est théoriquement applicable en dimension quelconque. Cependant, pour des raisons de complexité algorithmique combinatoire, la plupart des méthodes d'extraction à base de la transformée de Hough se limitent à des modèles simples et au cas 2D. Cette méthode a par exemple été utilisée avec succès en 2D pour détecter des droites, des cercles ou des ellipses [Dav97].

Pour effectuer l'accumulation, ces méthodes nécessitent une discrétisation de l'espace des paramètres. Or, ceci cause des problèmes au niveau de la précision ainsi qu'au niveau de l'encombrement mémoire lorsque le nombre de paramètres définissant les modèles recherchés augmente, ainsi que lorsque le nombre de points augmente (voir par exemple [BG99, p. 265]). Aussi ces méthodes sont-elles généralement utilisées pour des primitives et des scènes relativement simples (2D). Zhang [Zha96, p.29] remarque que ces méthodes sont rarement appliquées à des problèmes faisant intervenir plus de 3 inconnues.

Méthodes de Monte-Carlo (échantillons aléatoires) Les méthodes à base d'échantillons aléatoires de points évoquées au 2.2.2 peuvent s'étendre facilement à l'extraction de modèles et à la segmentation. Ceci donne lieu à des méthodes proches de la transformée de Hough mais ne nécessitant pas la même discrétisation. Une description détaillée de ces méthodes est faite au 6.5, page 174.

La méthode de Roth et Levine [RL94, RL93, RL92] permet d'extraire des sphères et des plans en 3D. Une méthode analogue a été proposée pour l'extraction de plans dans un nuage de points 3D [BG99]. Pour le cas des cylindres ou des cônes, la proposition de Roth et Levine, qui consiste à partir des bases de Gröbner, n'a pas donné de résultat. Ce problème spécifique est également abordé au 6.5 et à l'annexe D.

Remarque La TH et ses variantes, en tout cas dans leur version standard, mettent en œuvre un vote global, qui ne tient pas compte des arrangements de points locaux, ce qui semble dommage dans notre cas.

Méthodes par apprentissage

La fonction *recognize* de l'algorithme *Upwrite* de U Western Australia est un exemple de reconnaissance par apprentissage. A partir des groupements initiaux obtenus par une segmentation sans

modèle (fonction *chunking* décrite ci avant), l'algorithme fait une reconnaissance des groupements de points dans un espace de moments de Zernike ($\mathbb{R}^7$), à l'aide d'un apprentissage préalable. La fonction permet d'extraire les droites, les cercles et les ellipses. Le programme en 2D [MA98, MA97], téléchargeable depuis le site de University of Western Australia, permet de comparer les performances de la méthode proposée avec celles des transformées de Hough (qui s'avèrent bien moins performantes). Cette méthode a également été utilisée en 3D, pour extraire des lignes électriques (chaînettes) dans des environnements extérieurs numérisés, ainsi que pour extraire des formes approximativement cylindriques dans un espace abstrait [McL00].

Le principe de cet apprentissage est à rapprocher des méthodes de regroupement ou de création d'amas (*clustering methods*). En effet, l'apprentissage qui vient d'être évoqué est une séparation d'amas dans un espace de paramètres. Notons qu'il est possible d'adapter les méthodes de création d'amas pour extraire certaines surfaces, tels les plans [LGS99].

Croissance de régions

Les méthodes dites de croissance de région (*region growing*) se basent sur des régions germes (*seed regions*) qui grossissent par l'ajout de points voisins satisfaisant un critère d'appartenance au modèle. Ce procédé a été beaucoup utilisé pour des images de profondeur [HJBJ+96, Gal01], et parfois sur des maillages 3D. Souvent — mais pas toujours [LJS97] —, cette segmentation s'effectue à l'issue d'une pré-segmentation sans modèle (voir section 2.1.3). Les méthodes varient suivant :

- le choix des régions germes, et
- le type de croissance.

Les régions germes sont réparties sur une grille régulière 2,5D [LGB95, LJS97, LMM98], aléatoirement, ou à l'aide d'une pré-segmentation sans modèle (suivant un critère de taille : algorithme UB [HJBJ+96, PBJB98, JB94]), à l'aide d'un filtre de Kalman [DT96], ...

Parmi les procédés de croissance, on trouve des procédés de croissance simple, de croissance - ajustement - contraction (*grow-fit-contract*) (UE [HJBJ+96]), ou de croissance - sélection (*recover & select*) (sélection du meilleur modèle parmi plusieurs concurrents au bout de n croissances) [LJS97, LGB95]. Cette dernière méthode, de Leonardis et al., dont une illustration est donnée figure 2.22, est disponible en ligne sur le site Internet de U Ljubljana (Slovénie). La scène 2,5D est segmentée en super-quadriques. En distant, l'extraction paraît longue (40 min en distant sur Internet, pour une scène du type de celle donnée figure 2.22).

L'un des travaux principaux traitant spécifiquement le problème de la segmentation 3D à base de modèles plus complexes que le plan est celui de Fisher et al. [FFE97], où est introduite la notion de croissance de surface sur un maillage 3D. Comme on l'a vu précédemment, la méthode de Fisher et al. utilise au préalable l'algorithme de Hoppe et al. [HDD+92] pour construire un maillage à partir des points 3D. Notons encore une fois que cette méthode de construction de maillage nécessite une densité assez uniforme des points sur la surface. C'est généralement le cas des pièces mécaniques qui sont scannées de près. Ensuite, à partir d'une segmentation initiale sur le maillage 3D basée sur une estimation des courbures, l'algorithme fait une croissance de région, avec des fonctions d'ajustement spécifiques à chaque type de primitive (Figure 2.9, page 27).

La méthode générale utilisée dans [LMM98] suit le même principe, «recover & select», que l'on vient de mentionner. Au lieu d'une segmentation initiale, les régions germes sont choisies régulièrement dans l'image 2,5D . La segmentation et la modélisation sont donc produites simultanément. La particularité principale de ce travail par rapport au précédent est de faire intervenir des surfaces plus complexes telles que le cône ou le tore.

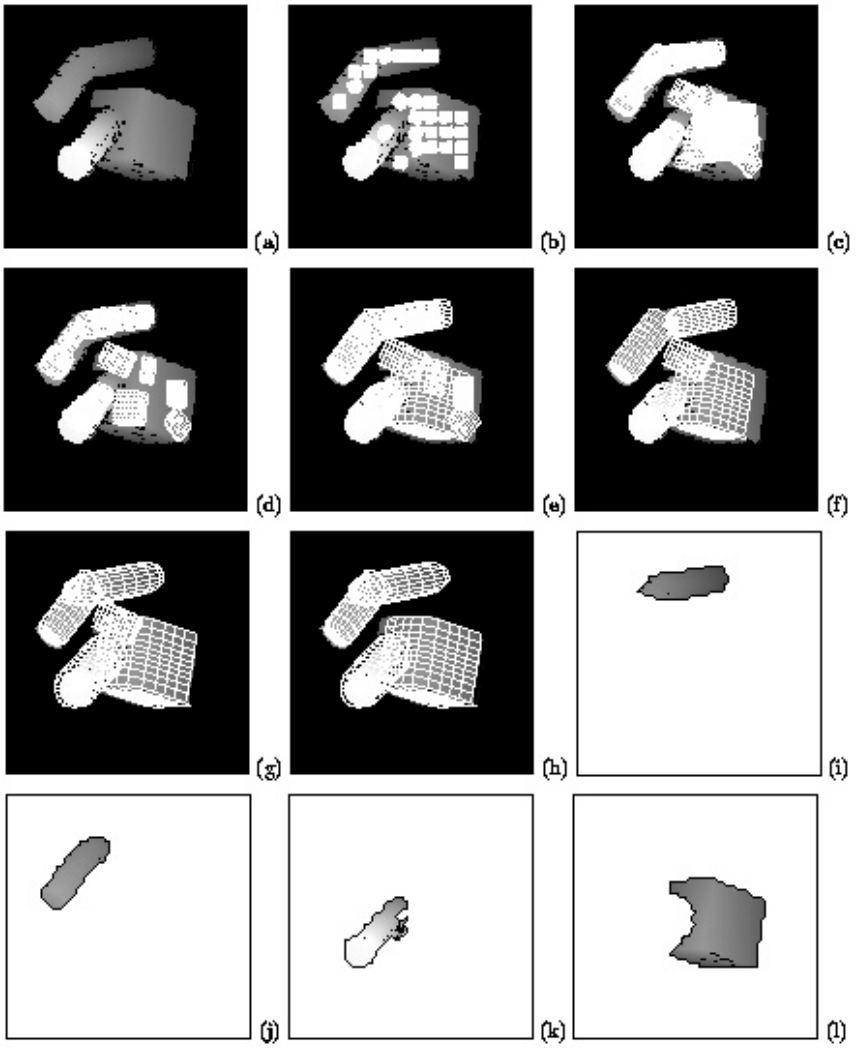


FIG. 2.22 – *Segmentor* de U Ljubljana. Principe du *recover & select* : des modèles sont créés sur des régions germes (image a). Ensuite, a lieu la phase de croissance et de compétition de modèles (images b–h). Ceci donne la segmentation finale(images i–l)

Division-fusion (*split & merge*)

La méthode de division-fusion est une manière différente de gérer les régions, qui a été essentiellement appliquée à des images 2,5D. L'image 2,5D est divisée en morceaux puis des régions sont itérativement fusionnées [Nat98]. La fusion s'arrête lorsque les régions ne sont pas assez similaires deux à deux. La division-fusion va plutôt du global au local (*bottom-up*) alors que la croissance de régions va plutôt du local au global (*top-down*). Faugeras [Fau93] souligne la difficulté de définir un critère cohérent pour diviser la scène.

2.2.5 Création de modèles plus complexes

Il est utile, pour obtenir un véritable modèle CAO, de créer des descriptions plus complexes des scènes ou objets segmentés, où des contraintes sont nécessaires (par exemple, contraintes d'adjacence de deux faces, perpendicularité, symétrie, axe de rotation, etc.). Notons que très peu de travaux introduisent les contraintes — ou connaissance «métier» — lors des étapes précédentes [WFAR99]. Un exemple de descriptions plus complexes est la *b-rep* (d'un objet ou d'une scène complexe), qui peut être reconstruite à partir de scènes segmentées [HGB95, HGB98b, HGB98a, BKV+02]. L'application privilégiée concerne les pièces mécaniques [BKV+02, Mey02]. L'introduction des contraintes fait en général intervenir les multiplicateurs de Lagrange. Dans ces systèmes, les contraintes sont pour l'instant spécifiées par l'utilisateur.

2.3 Tableau récapitulatif des systèmes testés ou disponibles

Le tableau 2.2 présente de manière synthétique les systèmes testés ou disponibles réalisant les traitements que nous avons décrits dans ce chapitre.

TAB. 2.2 – Tableau récapitulatif des systèmes testés ou disponibles

Programme (laboratoire)	Cas	Résultat	Principe	Test et remarques
Transf. de Hough (U Western Australia)	2D	Extraction de droites, cercles, et ellipses. TH standard, aléatoire, hiérarchique	Accumulation dans esp. de paramètres	Démo en ligne. Droite et cercle : OK (qques s), ellipses : long (min ou heures)
UpWrite (U Western Australia)	2D	extraction de droites, cercles et ellipses	morcellement puis reconnaissance par apprentissage	Démo en ligne. Un peu meilleur que la TH
Crust (U Texas et U British Columbia)	2D	Squelette et «croûte» (arêtes entre points)	Diag. de Voronoï et Triang. de Delaunay	Démo en ligne(en 2D). Travaux en 2D
BJ (Besl et Jain) (GM / U Michigan)	2,5D	Surfaces $z = f(x,y)$ de degré au plus 4×4	Croissance de régions	Nettement moins rapide que UB [PBJB98]. Non testé
UB (CurveSegmenter) (U Bern)	2,5D	Régions planes et courbes (choix). Surfaces $z = f(x,y)$ de degré 4×4	Partition de chaque ligne de scan, puis fusion	Environ 100 s pour une image 480×640 (300000 pts) sur SGI Indy
USF (U South Florida)	2,5D	Régions planes	Classification, et croissance de régions	Moins bon et beaucoup moins rapide que UB [HJBJ+96]. Non testé
UE (U Edinburgh)	2,5D	Régions planes et quadriques	Classif. initiale (H,K) puis croissance/contraction	Pour les plans, résultats comparables (un peu mieux ?) que UB, mais nettement moins rapide [HJBJ+96]
WSU (Washington State U)	2,5D	Régions planes et bi-quadratiques	Discontinuités, normales, *clustering* dans $\mathbb{R}^6$	Pour les plans, moins bon et beaucoup moins rapide que UB [HJBJ+96]. Non testé
Segmentor (U Ljubljana)	2,5D	super-quadriques (scène 2,5D mais modèle 3D)	croissance de régions et compétition de modèles(*recover and select*)	Démo en ligne. Semble long : 40 min (en distant, SGI Indy) sur une scène typique...
Logiciels commerciaux CGP v2.01 et Cyclone (Cyra)	2,5D	Segmentation sans modèle (zones connexes)	Discontinuités en profondeur et orientation + morphologie	Environ 30s pour une scène de 100000 points
Hoppe et al.1992 (Microsoft Res./ U Washington)	3D	Maillage (reconstruction sans segmentation)	Estimation des normales puis variante des *marching cubes*	1 min environ sur la pièce de la fig. 2.2 (4000 pts) sur MIPS 20 [HDD+92]. Nécessite une densité de points régulière. Non testé.
TensorVoting (U South California)	3D	Maillage + jonctions . Segmente les surfaces séparées par des plis anguleux.	Vote local et détection des maxima de saillance	7 min pour scène 5000 points avec 2 surfaces (Pentium III 500 MHz)

Chapitre 3

Segmentation de lignes de tuyauterie

La segmentation d'une ligne de tuyauterie indique implicitement un objectif double : extraire les points qui se trouvent sur la ligne de tuyauterie du reste de la scène d'une part, et séparer le sous-nuage de points de la ligne suivant les différentes surfaces d'autre part. Le deuxième point est étroitement lié à la production du modèle CAO de la ligne de tuyauterie. Nous avons choisi de distinguer ces deux étapes au niveau des algorithmes développés.

La section 3.1 présente des informations spécifiques aux lignes de tuyauterie qui sont importantes pour l'extraction et la modélisation. La section 3.2 traite de l'extraction d'une ligne de tuyauterie d'une scène complète. La section 3.3 traite de la modélisation de la ligne de tuyauterie extraite.

Ceci étant, il est à noter que l'extraction développée est elle-même encore basée sur la modélisation, c'est-à-dire sur la notion d'ajustement de primitive.

D'autre part, étant donné la complexité de la tâche à réaliser et l'obligation d'obtenir une méthode qui fonctionne dans un grand nombre de cas, nous avons opté pour un outil semi-automatique, i.e. interactif.

3.1 Connaissances «métier»

3.1.1 Qu'est-ce qu'une ligne de tuyauterie ?

Une ligne de tuyauterie est un ensemble d'éléments tels que tubes, raccorderie (coudes, té, etc.), robinetterie, supports et autres éléments extérieurs (calorifuge, ...), qui permet de véhiculer un fluide d'un point à un autre d'une installation, d'un réservoir à une pompe, d'une pompe à un échangeur, etc. (Figure 3.1).

3.1.2 Eléments constitutifs

Sur une ligne de tuyauterie, on peut trouver les objets suivants (Figures 3.2 et 3.3) :

Objets de liaison
- tube (*tubing*)
- soudure (*weld*)

Objets de changement de direction
- coude (*elbow*)
- cintrage (*bend*)

Objets de changement de diamètre

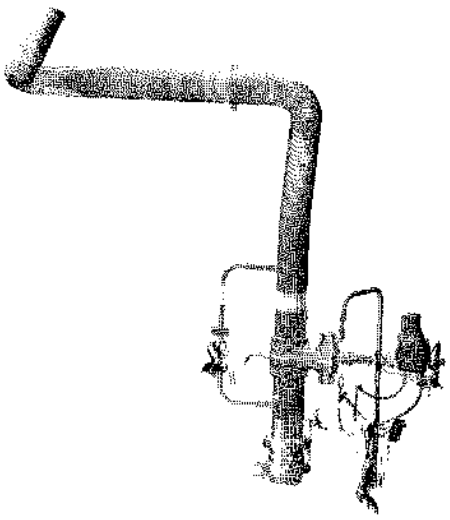


FIG. 3.1 – Ligne de tuyauterie numérisée

FIG. 3.2 – Brides

- réduction concentrique (*cocentric reducer*)
- réduction excentrique (*eccentric reducer*)
- manchon égal ou réduit (*coupling*)

Objets d'obturation
- fond plein (*cap*)
- bride pleine (*blind flange*)
- bouchon obturé

Objets de dérivation
- té (*tee*)
- piquage (*olet*)

Objets de raccorderie
- bride (*flange*)
- joint (*gasket*)
- raccord (*coupling*)

Objets d'instrumentation, d'isolement et autres

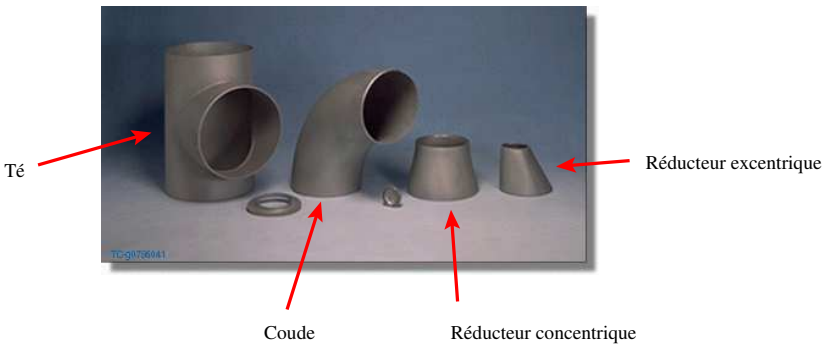


FIG. 3.3 – Raccords

– vanne, clapet (*valve*)
– support
– attache

Il existe un grand nombre d'autres objets pouvant se trouver sur une ligne de tuyauterie.

3.1.3 Compléments d'information

Les informations qui suivent sont issues d'échanges avec un expert tuyauterie d'EDF (Alain Pérard). Si cette partie est importante concernant les informations que l'on peut ou doit prendre en compte lors de la modélisation tel que construit d'une ligne de tuyauterie, le lecteur pressé peut dans un premier temps passer directement à la caractérisation géométrique d'une ligne de tuyauterie (section 3.1.5) et se référer à cette section pour obtenir des renseignements complémentaires.

Les types de matériels utilisés, les technologies retenues en matière d'assemblages et de robinetterie, dépendent pour l'essentiel du fluide véhiculé, des conditions de fonctionnement, et des risques associés compte tenu des conséquences en cas de défaillance et de l'impact sur l'environnement. On trouve différents types de lignes de tuyauterie :

– ligne de tuyauterie normale,
– ligne de tuyauterie spéciale,
– ligne de tuyauterie calorifugée,
– ligne de tuyauterie avec traçage.

Ligne de tuyauterie normale

Une ligne de tuyauterie «normale» est soit peinte (tuyauterie en acier noir), soit brute (ex inox).

Pour ces lignes de tuyauterie, les diamètres externes sont normalisés (tables). La seule variable est l'épaisseur du tube et l'on parle alors de diamètre normalisé (DN). Ceci est valable jusqu'à un certain diamètre, au-delà duquel on a affaire à des fabrications spéciales soit en tôles roulées et soudées (le diamètre intérieur est égal au DN et le diamètre extérieur varie en fonction de l'épaisseur de la tôle), soit un tuyau moulé ou forgé (ex : circuit primaire du bâtiment réacteur pour des raisons d'épaisseur variable du circuit).

Ligne de tuyauterie spéciale

Cette catégorie comprend :

- la tuyauterie en matériaux composites (époxy, résines enrubannées, résines moulées, etc.). Pour ces tuyaux le diamètre intérieur est égal au DN et seule l'épaisseur est variable en fonction de la résistance souhaitée par des couches successives de résine.
- la tuyauterie en acier revêtu à l'intérieur (par exemple de téflon ou de résine), ou revêtu à l'extérieur pour une protection, la tuyauterie en béton avec âme intérieure en acier. Ces types de tuyaux sont souvent utilisés en réseau enterré sauf l'acier revêtu à l'intérieur utilisé dans les sites en bord de mer (pour la corrosion) et en cours d'être remplacés par des matériaux composites.
- la tuyauterie en PVC, utilisée en réseau enterré ou en architecture (gouttières, etc.), mais aussi utilisée par EDF dans certains circuits (par exemple dans la production d'eau déminéralisée en bord de mer, ou circuit pneumatique de circulation de document par navette).

Ligne de tuyauterie calorifugée

Le calorifuge est situé sous une coque aluminium ou acier inoxydable et fabriqué à la demande sans norme dimensionnelle standardisée. On ne peut pas deviner quel objet se trouve sous une tuyauterie calorifugée car l'épaisseur du calorifuge est calculée en fonction de la température du circuit. Il faut donc effectuer un relevé après décalorifugeage des circuits si l'on souhaite effectuer une reconstruction de la tuyauterie elle-même.

Ligne de tuyauterie avec traçage

Il s'agit ici de ligne avec un câble électrique servant de résistance, enrubanné autour de la tuyauterie, en général en spirale.

Objets de liaison

- Tube (*tubing*)
  De diamètre constant et de longueur maximum comprise entre 3, 6, 9 ou 12 mètres. Ces longueurs maximales sont celles fournies par les fabricants et sont utilisées au mieux dans les installations pour diminuer le nombre de soudures et pour l'introduction de ces tubes en phase de montage construction neuve ou de remplacement en phase de maintenance.
- Soudures (*weld*)
  Elles servent à assembler les tubes avec tous les autres éléments, elles peuvent aussi servir à corriger la pente d'une tuyauterie ou à générer une contre-pente et ceci dans une certaine limite angulaire (1 à 2° maxi). Les différents types de soudure que l'on peut rencontrer sont : la soudure simple, la soudure inspectable (en service ou non), la soudure terminale, la coupe biaise, les soudures de préfabrication et les soudures chantiers.

Objets de changement de direction

- Coudes (*elbow*)
  De rayon de courbure fixe et de diamètre constant. Le rayon de courbure est choisi dans des séries de modèle dit 2D, 3D, 5D. Ceci correspond à un rayon de courbure de 2 fois, 3 fois ou 5 fois le diamètre du coude. En amont et en aval du coude, il y a obligatoirement une soudure.
- Cintrages (*bend*)
  De rayon de courbure fixe et de diamètre constant. Le rayon de courbure est fourni par la

machine à cintrer du constructeur choisi. Il existe des tables fournies pour chaque contrat de tuyauterie. Cependant, il existe beaucoup de tables de machines à cintrer.
A la différence des coudes, il faut une certaine sur-longueur en amont et en aval avant de pouvoir mettre soit un autre cintrage, soit une soudure à un nouvel objet.

Objets de changement de diamètre

- Réductions (*reducers*) concentriques ou excentriques
  Le choix de prendre une réduction excentrique est conditionné par le fait de garder le même alignement de la génératrice inférieure du tuyau (fil d'eau) pour une pente constante de la tuyauterie dans le même sens (pas de points bas) et pour garder aussi le même fer de supportage commun à d'autres tuyauterie. Ce choix permet aussi de décentrer une nappe de tuyauterie pour rattraper une autre nappe et libérer de la place pour de nouvelles lignes.
- Manchons égaux ou réduits (*coupling*)
  Ils sont utilisés dans une certaine plage de diamètre (du DN 8 au DN 40-50 maximum). Ils sont en général de type Socket Welding (emmanché soudé) c'est-à-dire que le tube rentre dans la pièce pour renforcer la soudure et la tenue mécanique de ces petits diamètres.

Objets d'obturation

- fonds pleins (*caps*)
  ils sont utilisés dans les cas suivants :
  - tuyauterie en attente de nouvelle installation (ex fond de galerie pour une tranche future),
  - dans les cas où l'on n'a pas la place de mettre un coude ou un cintrage,
  - pour renforcer la tenue mécanique d'une ligne ne passant pas au calcul à cause du coude (dans ce cas on met un té égal connecté par la dérivation et on met un bouchon sur l'une des extrémités).
- Brides pleines (*Blind Flange*)
  Elles sont utilisées de manière à pouvoir inspecter la tuyauterie ou pour raccorder des appareils de nettoyage ou en attente d'extension d'installation.
- Bouchons obturés (coupling)
  Idem fonds pleins (souvent en socket welding et utilisé en diamètre inférieur à 60mm).

Objets de dérivation

- Tés (*tee*)
  ils sont utilisés pour raccorder une autre tuyauterie sur la tuyauterie principale. Ils sont de deux types différents : Tés Egaux ou Tés réduits. En général, c'est le diamètre de la dérivation qui est réduit. On trouve différents modèles de raccordement (par exemple soudé bout à bout, socket welding, emmanché et vissé, manchon idem que les coupling, etc.).
- Piquages (*olet*)
  Ils sont utilisés dans certaines plages de diamètre : dans les grands diamètres où il n'existe plus de té et aussi dans le cas où le rapport du diamètre de la dérivation par rapport au diamètre de la tuyauterie principale est trop élevé. On trouve deux types de piquages :
  - tube que l'on soude sur la tuyauterie principale et où l'on perce le trou ensuite
  - et l'autre cas où l'on soude un renfort (bossage socket) sur la tuyauterie principale et où l'on vient accoster le tube de la dérivation pour soudage (il s'agit de pièces spécifiques avec souvent des renforts).

Objets de raccorderie

- Brides (*flange*)
  Elles sont souvent utilisées pour raccorder une tuyauterie sur un appareil (en extrémité de ligne) ou en assemblage par paire pour faciliter le démontage d'un tronçon de tuyauterie (pour la maintenance), ou encore en ligne pour inclure du matériel (ex filtre, compteur ou mesure, etc.). On trouve dans les brides différents modèles de raccordement (par exemple soudé bout à bout, socket welding, emmanché et vissé, etc.). Les modèles sont les suivants : bride à collerette, bride plate, bride welding neck ou bride pleine.
  La taille des brides est essentiellement liée à la taille des boulons qui servent à la fixer, et non au diamètre du tuyau. Ceci implique par exemple que l'épaisseur radiale n'est ni constante, ni proportionnelle au diamètre du tube. Voir par exemple les tables de tailles standard dans [TC89].
- Joints (*gasket*)
  Ils sont utilisés pour raccorder un objet entre brides avec une étanchéité assurée par le joint.
- Raccords (*coupling*)
  Ils sont utilisés pour un diamètre inférieur à 50 mm ; ils peuvent être de type égal ou réduit, de modèle male ou femelle ou vissé (raccord union type chauffage central).
- Purges ou évents (*olet*)
  Ils permettent de purger ou d'éventer les circuits lors de l'arrêt ou la mise en service de celui-ci. Idem piquage.

Objets d'instrumentation, d'isolement et autres

Il existe un grand nombre d'autres objets pouvant se trouver sur une ligne de tuyauterie.

- Vannes (*valve*)
  elles sont utilisées pour isoler, réguler ou sectionner un circuit de tuyauterie. On trouve dans les vannes différents modèles de raccordement (ex soudé bout à bout, socket welding, emmanché et vissé, etc..).
- Clapets (valve)
  ils sont utilisés pour garantir le non-retour des fluides dans le sens inverse prévu par le fonctionnement (modèle à souder et modèle entre brides).

Les coudes et la pente de tuyauterie

- La pente : toute installation industrielle de tuyauterie possède des pentes pour permettre la vidange complète d'un circuit et l'éventage de celui-ci lors de la mise en service. Elle est complétée au besoin par des évents dans les points hauts et par des purges ou pots de purges (pour les circuits vapeur) pour les points bas. La pente est variable en fonction des circuits et des fluides véhiculés et peut varier dans certaines zones suivant les possibilités de la géométrie des locaux. Elle va de 1 mm par mètres à 2 cm par mètres. Ceci conduit à des changements importants d'élévation alors que l'on pourrait croire qu'une ligne de tuyauterie est dans le même plan. Pour l'installateur, il est absolument nécessaire lors de la reconstruction d'avoir les élévations les plus précises possibles. Ces pentes sont obtenues en fabrication essentiellement par les coudes.
- Le coude est une pièce de raccorderie existant au catalogue des fournisseurs en 30˚, 45˚, 60˚, 90˚ et 180˚ alors que le cintrage est du tube passé dans une machine à cintrer (à chaud ou à froid). Le coude par ses soudures et un meulage particulier du bord permet aux constructeurs de jouer sur l'angle de celui ci de 1 à 2 degrés et de générer les pentes sur les tuyauteries.

Ceci est aussi valable pour les coudes socket welding (emmanché et soudé) où un certain jeu dans l'emmanchement du tube dans le coude permet de générer la pente. Il faut lors de la reconstruction utiliser des tores circulaires avec des angles variables et non des valeurs figés (ex 90˚). Le rayon de cintrage des coudes est figé dans la norme alors que celui des cintres est variable suivant les cintreuses des fournisseurs.

3.1.4 Que doit-on modéliser ?

FIG. 3.4 – Ligne de la figure 3.1 reconstruite et exportée dans le logiciel PDMS

Pour un tuyauteur, qu'est-il intéressant de pouvoir repérer lorsque l'on réalise une modélisation tel que construit de tuyauterie (Figure 3.4) ? Par exemple, les emplacements des brides sont-ils importants ? les attaches ?

En principe pour un tuyauteur tout ce qui est sur une ligne est important, mais dans la pratique cela dépend à quoi sert ensuite le relevé. Ce relevé peut servir à différents types d'applications.

Démantèlement d'installation L'intérêt est dans un premier temps d'avoir une image fidèle de l'installation pour laquelle on n'a plus de plans à jour, pour pouvoir étudier les accès, les métrés approximatif des installations à démanteler, le tronçonnage possible sur les parties droites et/ou le calcul des masses de ces tronçons avec le centre de gravité pour les manipuler par un robot, etc. Même si la modélisation n'est pas très précise, si on a le nuage de point et un visualiseur permettant d'effectuer des mesures ponctuelles pour affiner certaines études, cela peut suffire dans une bonne partie des cas.

Maintenance d'une installation existante Le besoin est d'avoir une image précise de la tuyauterie dans les cas ou l'on doit modifier celle-ci, soit par ajout de nouveaux éléments, soit par modification des éléments existants (remplacement d'un té ou d'un piquage défaillant). Pour la première situation, il est nécessaire de connaître les parties droites et libres d'accès. Dans la seconde situation, il est nécessaire d'avoir une connaissance des éléments en amont et en aval de l'élément à remplacer.

Nouvelle installation dans un environnement existant L'intérêt est dans un premier temps d'avoir une image fidèle de l'installation pour laquelle il n'y a pas de plans à jour, afin d'étudier les accès pour l'introduction du nouveau matériel et du matériel de chantier (ex : poste de soudage, protection, échafaudage, etc.). L'étude est menée sur une maquette numérique en 3D qui permet de simuler différentes installations et d'en étudier les coûts avant de lancer la réalisation. Il n'est


FIG. 3.5 – Photographies prises sur site par la caméra du scanner Soisic

pas fondamental ici d'avoir une modélisation très précise. Dans certains cas, le nuage de point seul peut même suffire (si un visualiseur permettant d'effectuer des mesures ponctuelles pour affiner certaines études est disponible).

Pour les brides et les vannes, il n'est pas toujours nécessaire de reconstruire les vis et boulons. En effet, dans le logiciel PDMS, on pointe sur un élément de catalogues représentant la bride sans ces accessoires, un module particulier de PDMS fournissant le métré des accessoires de la branche de tuyauterie. Pour les vannes, il est intéressant de savoir s'il y a des brides en amont et en aval, ou si celle-ci est soudée sur la tuyauterie.

Entre chaque paire de brides ou brides-vannes, il y a un joint de 2mm d'épaisseur (cas général) sauf pour certaines vannes ou appareils en ligne où la face de bride possède une gorge centrale avec un joint torique.

De nombreux objets ne sont plus aisément modélisables à partir du nuage de points seul (à moins d'avoir effectué un scan dédié à ces objets-là, ce qui est plutôt rare). En pratique, il n'est pas rare de se référer à des photographies prises sur site pendant l'acquisition pour aider la compréhension et la modélisation (Figure 3.5). Ces photographies sont d'ailleurs intégrées dans les scènes 3Dipsos et Realworks. Ceci permet notamment de visionner le nuage de points du point de vue d'où a été prise la photographie.

3.1.5 Caractérisation géométrique d'une ligne de tuyauterie

On déduit des informations précédentes le type des surfaces qui sont susceptibles d'être présentes sur une ligne de tuyauterie, ainsi que des relations entre ces surfaces. Par exemple, après un tube droit (partie cylindrique), on considère que la primitive suivante correspond à l'un des cas suivants (Figure 3.6) :

- cylindre d'axe concourrant et de même rayon [ex : piquage],
- tube sécant d'axe concourrant et de rayon distinct [ex : té, tube soudé],
- tore circulaire joignant continûment le cylindre précédent (même diamètre, directrices tangentes) [ex : coude, cintrage],
- cône de révolution (même axe, jonction continue) [ex : réduction concentrique],
- cône elliptique (axes distincts, tangente commune, jonction continue) [ex : réduction excentrique],
- plan [ex : bords de brides, fin de tuyauterie, tube passant à travers un mur dans une trémie ou un fourreau]

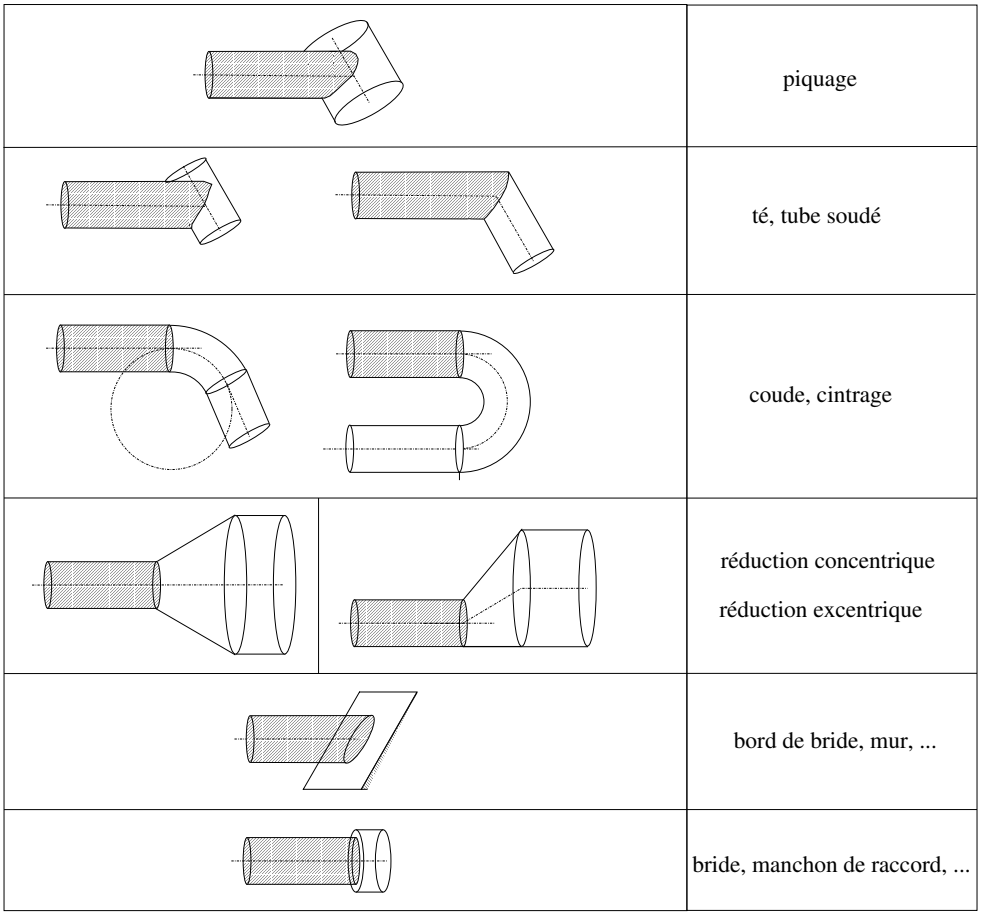


FIG. 3.6 – Connaissances «métier» sur les surfaces le long d'une ligne de tuyauterie

– tube coaxial (même axe, rayon différent) [ex : manchons cylindriques, brides].

D'autre part, certains éléments ont des dimensions standardisées, issues de tables. Or, il existe de nombreuses tables, et tous les éléments n'ont pas de dimensions standardisées (ex : cintrage). Aussi notre approche a-t-elle été de ne pas utiliser de tables. Il est cependant possible d'adapter les méthodes d'ajustement de primitives pour tenir compte de valeurs standard.

3.2 Extraction d'une ligne de tuyauterie

L'objectif est ici d'extraire les points qui se trouvent sur la ligne de tuyauterie, dans un premier temps sans modéliser celle-ci très précisément. C'est pourquoi un seul type de primitive (cylindre) a été considéré ici. Ceci correspond à peu près à la première étape de la segmentation manuelle dans 3Dipsos.

3.2.1 Présentation de l'application développée

La forme de l'algorithme développé est résumée à travers le pseudo-code suivant :

```
Initialisation
    Sélection d'un point
    Construction surface germe
Propagation
    Faire
        Déterminer nouveau voisinage
        Construction surface sur ce voisinage
    Tant que le critère d'arrêt n'est pas atteint
```

Le cadre général de l'approche ainsi que les thèmes qui interviennent dans cet algorithme sont synthétisés figure 3.7.

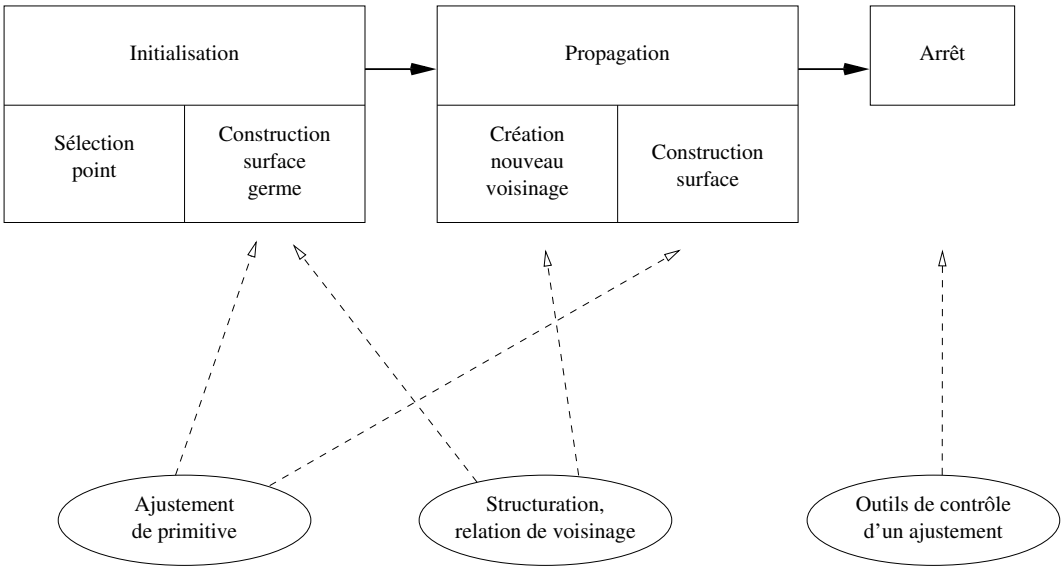
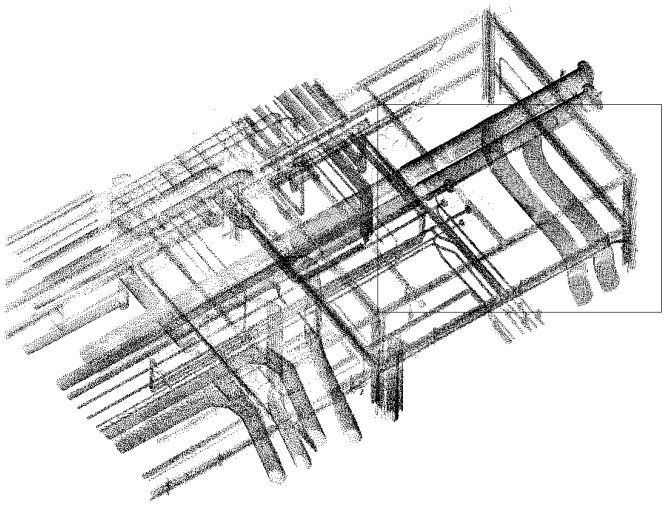


FIG. 3.7 – Cadre général de notre approche

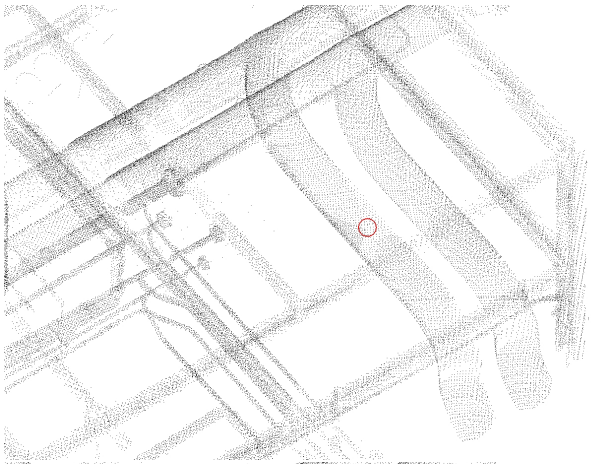
L'application développée se déroule donc comme suit :

Scène initiale (l'exemple contient ici 180000 points environ). Le cadre désigne ici la partie visualisée dans les schémas suivants (pour plus de clarté). Cependant, toute la scène est considérée dans les traitements qui suivent.

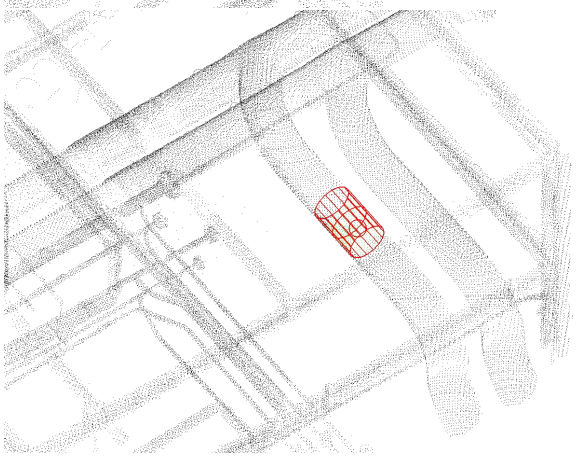


INITIALISATION

Etape 0 : sélection d'un point par l'utilisateur (le point sélectionné est entouré par un cercle sur la figure).

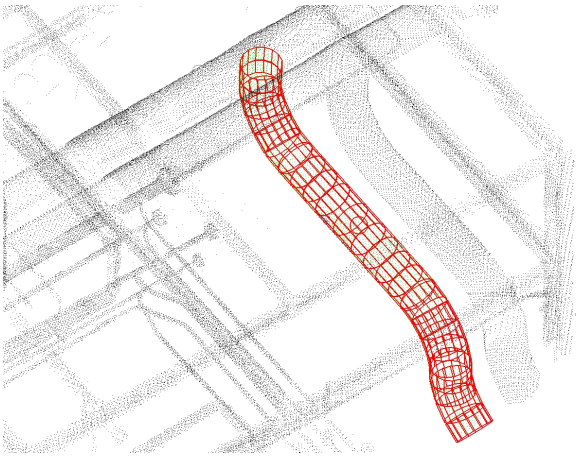


Etape 1 : construction d'un cylindre initial autour du point (procédé multi-résolution).

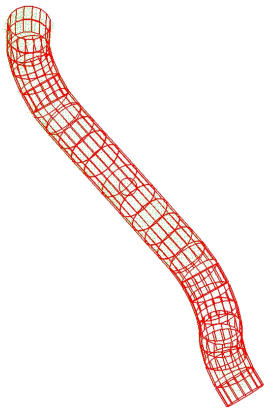


PROPAGATION

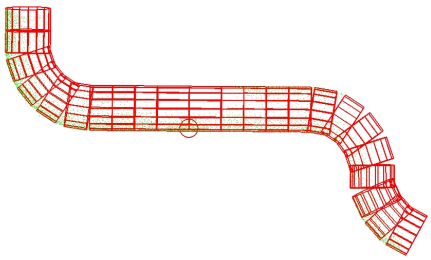
Etape 2 : propagation à partir des deux extrémités du cylindre initial par une succession de cylindres «enchaînés» (cylindres courts et contraints)



Résultat : points extraits (et séquence de cylindres contraints).



(même chose mais d'un autre point de vue)



3.2.2 Détail des différentes étapes

Etape 1 : Construction du cylindre initial

Le but est ici de trouver le cylindre sur lequel se trouve le point sélectionné. Pour cela, le principe est de faire l'ajustement d'un cylindre sur un voisinage autour du point sélectionné, c'est-à-dire un ensemble de points voisins du point sélectionné.

La méthode utilisée pour effectuer l'ajustement d'un cylindre sur un ensemble de points donné est décrite en détail au chapitre 4. Disons simplement ici que cet ajustement est une méthode d'estimation par moindres carrés, non-linéaire. Comme nous le verrons au chapitre 4, cet ajustement nécessite une estimation des normales à la surface en chaque point.

Le problème ici est que l'on ne dispose pas d'un ensemble de points donné à l'avance sur lequel effectuer l'ajustement. Plus précisément, on souhaite que cet ensemble de points constitue un voisinage du point sélectionné dans le nuage, mais on ne sait pas combien de points considérer. Autrement dit, quelle taille de voisinage choisir autour du point ? Si cette taille est trop petite, l'ajustement risque d'être mauvais, car le bruit sur les points prendra beaucoup d'importance. Si par contre cette taille est trop grande, l'ajustement sera vraisemblablement mauvais aussi, car le voisinage englobera des points qui se trouvent sur d'autres surfaces. De plus, contrairement au cas d'une image 2D ou 2,5D, on ne peut pas définir une taille de voisinage fixe sur toute la scène car la densité de points n'est pas uniforme.

D'autre part, il est important ici que cette estimation soit la plus précise possible (car les contraintes qui sont utilisées ensuite font que le résultat de la première étape a des répercussions sur tout le reste).

La méthode introduite ici suit un procédé multi-résolution : on essaie toute une gamme de tailles de voisinage, et on garde la «meilleure» taille, au sens d'un certain critère. En fait, un voisinage aura une taille optimale s'il est le plus gros possible (ce qui conduit à une meilleure estimation des paramètres du cylindre) sans contenir de points aberrants (notamment les points sur d'autres surfaces). Le détail de cet algorithme est décrit section 6.1.2, page 161.

A ce niveau, il y a un contrôle du rayon du cylindre obtenu. On regarde en particulier si les paramètres de ce cylindre sont compatibles avec des valeurs minimales et maximales. Ces valeurs extrémales, qui dépendent de la scène (notamment de l'unité de longueur utilisée), sont spécifiées par l'utilisateur à travers une boîte de dialogue (Figure 5.12, page 139).

Etape 2 : Propagation

Une fois le cylindre initial construit, la propagation s'effectue successivement de chaque côté du cylindre. Pour chacune des deux directions, on suit la ligne de tuyauterie en faisant une propagation à l'aide d'ajustements de cylindres contraints. Le principe de l'algorithme est présenté sous la forme du pseudo-code suivant :

Pseudo-code de l'étape 2

```
NuageRestant ← Tout le nuage
Pour chacun des deux sens faire
    Continuer ← vrai
    CylPrécédent ← Cyl0 (issu de l'étape 1)
    Direction de propagation ← Direction de Cyl0 dans le sens courant
    e_0 ← erreur d'ajustement de Cyl0
    Tant que (Continuer = vrai) faire
        V ← Points de NuageRestant dans demi-sphère de rayon R = ρr
        au bout de CylPrécédent
        Si (Nombre de points(V) < n_min) faire
            Continuer ← faux
        Sinon
            Cyl ← Ajustement de cylindre-rotule sur les points de V
            e ← Erreur d'ajustement de Cyl
            Si (e > βe_0) faire
                Continuer ← faux
            Sinon
                Ajouter Cyl à la liste de cylindres
                Enlever les indices des points de V de NuageRestant
                CylPrécédent ← Cyl
            Fin Si
        Fin Si
    Fin Tant que
    Retourner la séquence de cylindres et les sous-nuages associés
Fin Pour
```

Entrées : Nuage de points, Cylindre initial Cyl0 et son erreur d'ajustement e_0.
Paramètres : $\rho(=\sqrt{2})$, $n_{\min}(=10)$, $\beta(=10)$.
Sorties : Deux séquences de cylindres et sous-nuages associés, sous-nuage NuageRestant.

A chaque itération, on effectue les étapes suivantes :

Création du nouveau voisinage Le nouveau voisinage est déterminé comme suit : l'extrémité du cylindre précédent étant fixée, on considère les points qui se trouvent dans une demi-sphère de rayon $\sqrt{2}r$ dans le sens de la propagation (Figure 3.8). Pourquoi cette taille ? Comme on peut le voir sur la figure 3.8, $\sqrt{2}r$ est le rayon minimum pour que la demi-sphère englobe les points qui se trouveraient sur le cylindre suivant (de longueur de r).

Ajustement d'un cylindre contraint sur ces points Le type de primitive utilisée ici est le «cylindre-rotule» (pour une définition précise, se reporter au chapitre 4). C'est un cylindre dont le rayon est fixé et l'axe est contraint à passer par un point fixe. Autrement dit, seule la direction de l'axe est autorisée à varier (liaison rotule).

Ici, la valeur du rayon est celle du cylindre construit lors de l'étape 1. Le point fixé est l'extrémité du cylindre précédent dans le sens de la propagation (Figure 3.8).

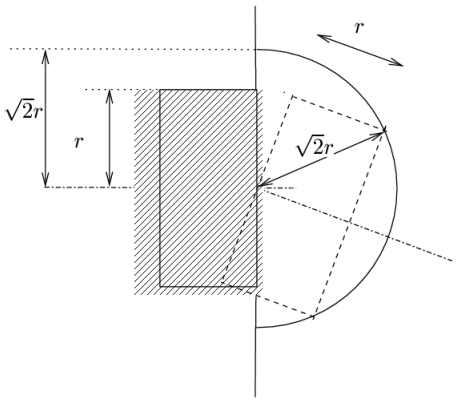


FIG. 3.8 – Création du nouveau voisinage

Le détail de l'ajustement de cette primitive est présenté au chapitre 4. Notons simplement que cet ajustement nécessite une valeur initiale. Cette valeur initiale pour le vecteur directeur de l'axe est choisie ici égale au vecteur directeur du cylindre précédent. Ceci exploite la continuité et la régularité de la ligne de tuyauterie : l'angle entre des cylindres consécutifs (et courts) sur la ligne n'est en général pas très grand.

Critères d'arrêt de la propagation

Les itérations précédentes s'arrêtent dans deux cas.

1. Il n'y a pas assez de points dans la demi-sphère courante. Plus précisément le nombre de points est inférieur à un seuil $n_{\min}$ ($n_{\min} = 10$ en pratique).
2. L'erreur d'ajustement est trop élevée. Plus précisément, le résidu quadratique moyen est supérieur au résidu quadratique moyen du cylindre initial multiplié par un certain facteur β ($\beta = 10$ en pratique).

3.2.3 Résultats sur scènes de différents scanners

Les algorithmes ont été développés principalement à partir de scènes du scanner Soisic de MENSI. Le principe de fonctionnement du scanner Soisic est un procédé de triangulation. Les caractéristiques des scènes obtenues sont associées à ce procédé : points très précis à courte distance, et de moins en moins précis plus la distance est grande. Il y a très peu de points aberrants, ni de points interpolants aux discontinuités des surfaces. Mis à part les éventuels problèmes de recalage, les scènes sont donc propres, précises dans les zones denses.

Il paraissait intéressant de tester les algorithmes présentés également sur des scènes d'environnements industriels issues d'autres scanners. Dans cette section, on montre le résultat de l'algorithme appliqué à quelques scènes d'autres scanners laser.

Scène du scanner Cyrax (société Cyra)

Le scanner Cyrax (Figure 3.9) de la société Cyra fonctionne sur le principe du temps de vol.

FIG. 3.9 – Scanner Cyrax(société Cyra)

Les résultats obtenus sur la scène de la figure 3.10 sont comparables à ceux sur les scènes provenant du scanner Soisic de MENSI.

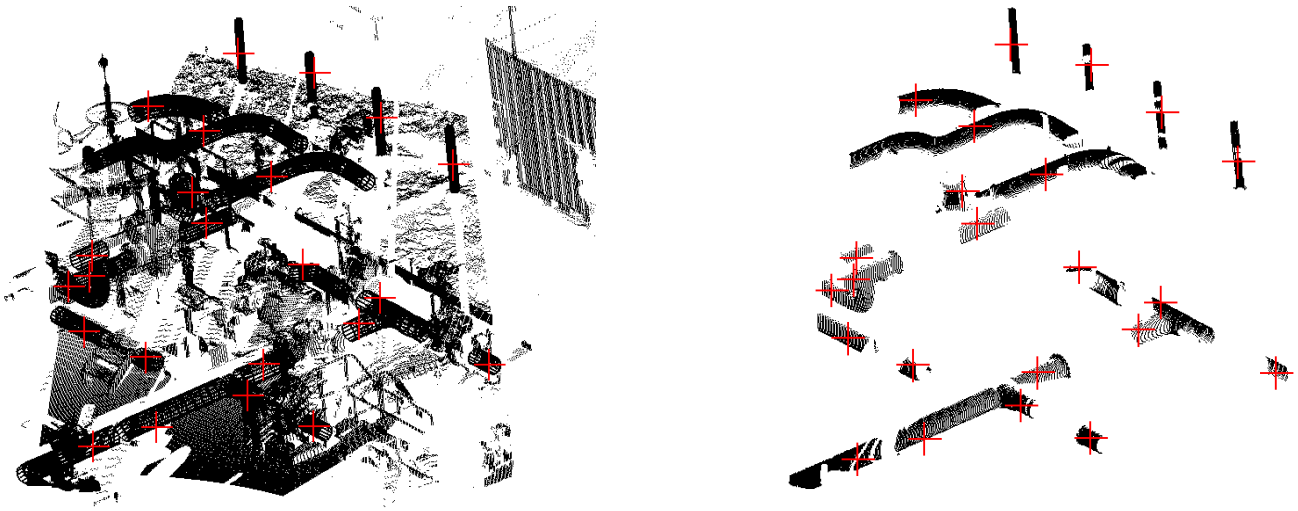


FIG. 3.10 – Résultat de la fonction d'extraction de lignes de tuyauterie sur une scène du scanner Cyrax

Scène du scanner Imager (société Zoller et Fröhlich)

Le scanner Imager (Figure 3.11) de la société Zoller et Fröhlich fonctionne sur le principe du temps de vol (différence de phase).

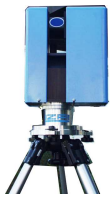


FIG. 3.11 – Scanner Imager (société Zoller et Fröhlich)

Les caractéristiques de la scène sont les suivantes : zones très denses, bruit assez élevé en zones denses, points interpolants aux discontinuités mais le nuage utilisé (Figure 3.12) a déjà subi un filtrage.

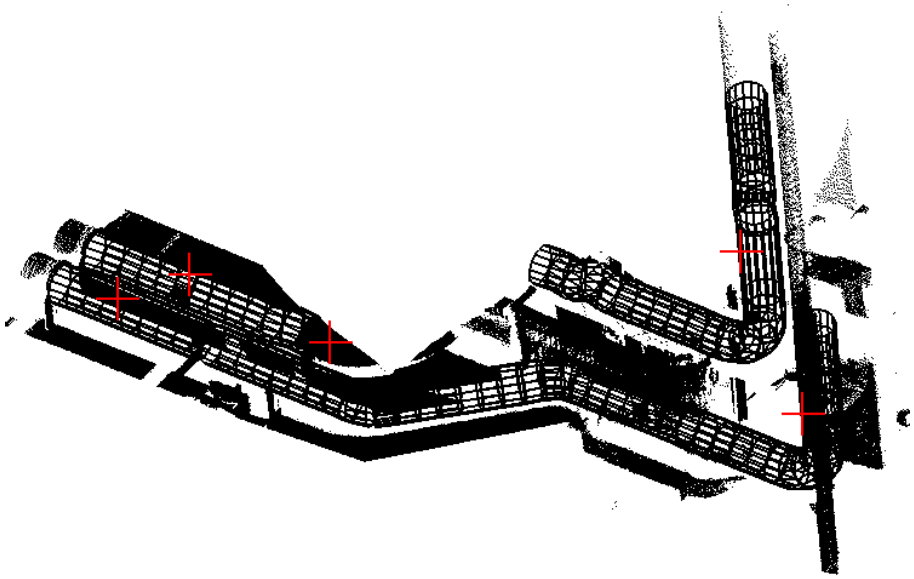


FIG. 3.12 – Scène du scanner Imager

Scène du scanner DeltaSphere (société 3rdTech)

Le scanner DeltaSphere(Figure 3.13) de la société 3rdTech fonctionne sur le principe du temps de vol.

FIG. 3.13 – Scanner DeltaSphere(société 3rdTech)

Les caractéristiques de la scène sont les suivantes : bruit élevé, points interpolants aux discontinuités (Figure 3.14).

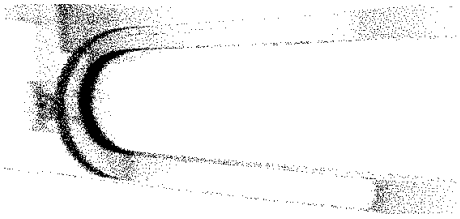


FIG. 3.14 – Détail de la scène issue du scanner DeltaSphere, comportant un tuyau vu en coupe, qui montre que le scanner produit des points aberrants aux zones de discontinuité de profondeur

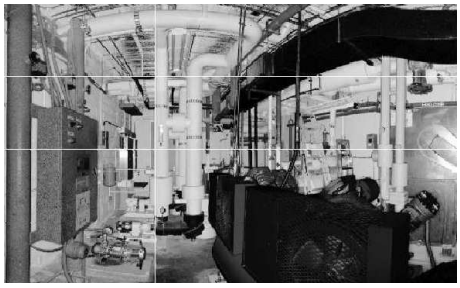


FIG. 3.15 – Scène du scanner DeltaSphere (luminance)

La scène testée est illustrée sur l'image de luminance obtenue par le scanner (Figure 3.15).

On constate que l'algorithme parvient à traiter ce type de scène (Figure 3.16). Cependant, on peut noter une tendance à la sous-estimation du rayon du cylindre initial dans certains cas.

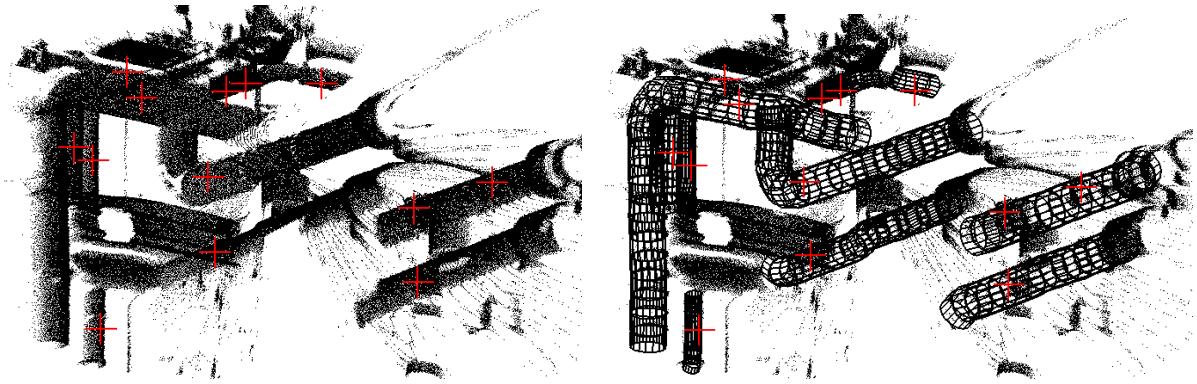


FIG. 3.16 – Scène du scanner DeltaSphere et résultat de la segmentation

Scène du scanner GS100 (société MENSI)

Le scanner GS100 (Figure 3.17) de la société MENSI fonctionne sur le principe du temps de vol.

FIG. 3.17 – Scanner GS100 (société MENSI)

Les caractéristiques de la scène utilisée sont : bruit au niveau des zones denses, pas ou peu de points interpolés aux discontinuités, forte erreur de recalage.

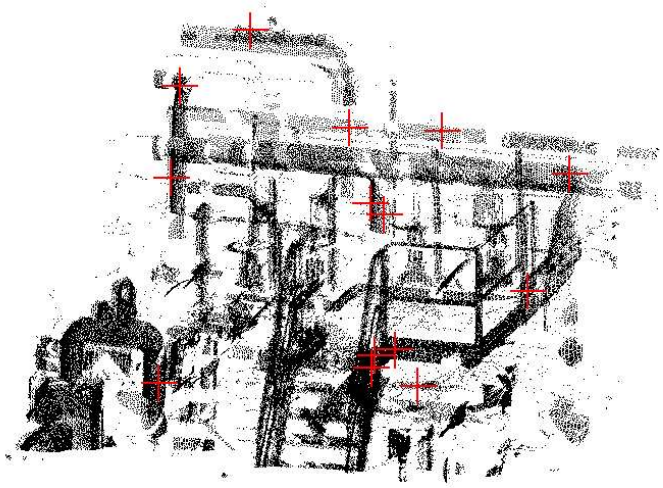


FIG. 3.18 – Scène du scanner GS100 (société MENSI)

L'extraction du cylindre initial ne donne pas toujours de bons résultats. L'erreur de recalage est la premier facteur qui intervient ici, plus que le niveau du bruit.

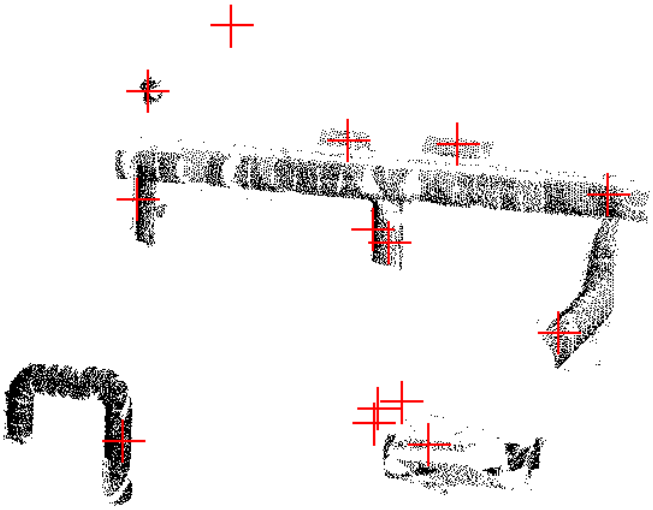


FIG. 3.19 – Quelques tuyaux extraits

Remarque sur les résultats

Les résultats obtenus au cours de ces quelques essais ont permis de voir que la méthode peut s'appliquer à des scènes issues de différents scanners. En particulier, le bruit dans les zones denses propres aux scanner à temps de vol ne semble pas trop gênant (jusqu'à un certain niveau !). Les points interpolés aux discontinuités ne constituent pas toujours un problème non plus. Par contre, les problèmes de recalage, qui sont peut-être plus fréquents avec des scanners à temps de vol, constituent un problème. En effet, l'ajustement de surfaces fonctionne moins bien dans ces situations.

3.2.4 Tests auprès d'utilisateurs experts de 3Dipsos

L'algorithme que nous venons de présenter a été implémenté au sein d'une application, nommée Cool[1]. Des tests de cette application ont été effectués auprès de quatre utilisateurs experts de 3Dipsos, dont deux personnes au département support de MENSI, et deux personnes au département EDF-CNEPE (Division Topographie) à Tours. Ces tests ont été menés les 3 et 4 avril 2002. La préparation de ces tests s'est faîte avec l'aide d'une ergonome d'EDF (Sandrine Tonnoir).

Objectif des tests

L'objectif des tests était de valider l'approche développée en déterminant si celle-ci, dans sa version actuelle, représente un outil véritablement intéressant pour l'étape d'extraction de lignes de tuyauterie. En particulier, il paraissait opportun, afin d'évaluer la valeur ajoutée du nouvel outil, de comparer, pour une même scène, l'extraction effectuée avec le procédé actuel dans le logiciel 3Dipsos d'une part à celle effectuée avec notre logiciel, Cool, d'autre part.

En outre, il s'agissait de proposer un scénario suffisamment réaliste pour que les utilisateurs pussent se faire un avis plus clair qu'une simple première impression. En particulier, une première version de l'algorithme avait déjà été montrée lors du club ingénierie de MENSI le 14/12/2000, et les utilisateurs présents (représentants de cabinets d'ingénierie : Technip, Chleq Froté, etc.) s'étaient montrés intéressés par l'approche. Mais ceci n'était bien entendu pas suffisant pour connaître l'apport réel de l'approche.

Descriptif des tests

Pour chaque utilisateur, le test a compris quatre phases :
- Présentation des tests et fonctionnement du programme
- Phase d'appropriation (de Cool)
- Phase de tests proprement dite (sur Cool et 3Dipsos)
- Réponse au questionnaire et discussion

Fonctionnement du programme La phase de présentation a consisté à présenter les objectifs des tests et le fonctionnement de la fonction d'extraction de lignes de tuyauterie du programme Cool.

L'interaction de l'utilisateur se limite au clic d'un point sur une partie de la ligne de tuyauterie, puis à l'acceptation ou au rejet du cylindre initial. Ensuite, l'algorithme effectue automatiquement la propagation (dans les deux sens) en affichant au fur et à mesure son avancée. Une fois la propagation terminée (critère d'arrêt), l'utilisateur peut visualiser interactivement la ligne de tuyauterie extraite (la séquence de cylindres, les sous-nuages associés, et le nuage de points restant). Puis l'utilisateur doit cliquer sur une autre ligne de tuyauterie pour continuer la segmentation. Notons que pour améliorer la clarté, les lignes de tuyauterie extraites sont automatiquement désaffichées lors du clic sur la ligne suivante.

Les utilisateurs testés ont tous une grande habitude du fonctionnement du logiciel 3Dipsos. En particulier, ils sont habitués à la manière dont la scène se déplace — ou de manière équivalente, dont le point de vue se déplace par rapport à la scène — lors des actions de la souris. Pour que ces questions interviennent le moins possible au cours des tests (au niveau du temps mais aussi et surtout au niveau

[1]Le nom de «Cool» n'a d'autre origine qu'une référence, certes peu imaginative, à Fun, nom d'une application de visualisation de nuages de points, ancêtre de 3Dipsos, initialement développée par X.Chen.

de la sensation de pénibilité), l'interface homme-machine du logiciel Cool a été adaptée à celle de 3Dipsos. Plus précisément, les actions des boutons et des mouvements de la souris ont été reproduits à l'identique dans Cool [2]. Seul le clic du point initial est légèrement différent de la même opération dans 3Dipsos, en ce qu'il nécessite l'activation d'un «mode sélection».

Appropriation Lors de la phase d'appropriation, chaque utilisateur manipule lui-même le logiciel Cool pour intégrer son fonctionnement. Ceci s'est fait sur la scène simple a_1 et une scène plus complexe a_2 (Figure 3.20).

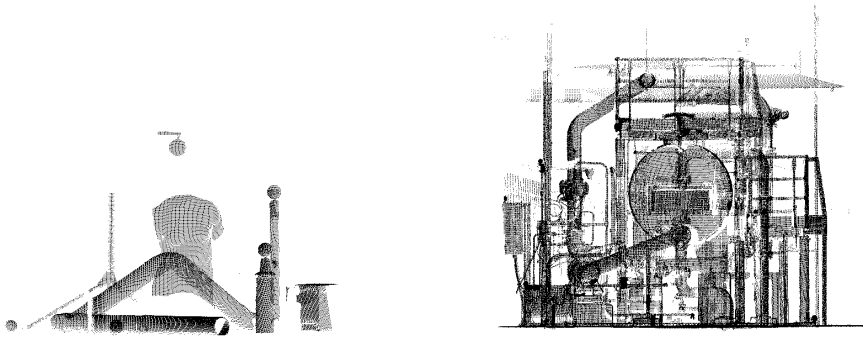


FIG. 3.20 – Scènes a_1 et a_2 pour la phase d'appropriation

Phase de tests proprement dite La phase de tests proprement dite comprend deux scénarii.

Scénario sur scène b_1 (Figure 3.21) Tâche à réaliser : extraire les deux principales lignes de tuyauterie.

Particularités de la scène : la scène b_1, également connue sous le nom de «minicouloir», comporte deux lignes de tuyauteries principales dont le parcours est assez complexe. Il existe de nombreuses surfaces proches des lignes (murs, ...). Le nuage contient 170 000 points environ.

Scénario sur scène b_2 (Figure 3.22) Tâche à réaliser : extraire toutes les lignes de tuyauterie.

Particularités de la scène : la scène b_2 est assez claire (pas de murs, peu de surfaces gênantes pour la perception), mais comporte un plus grand nombre de lignes de tuyauterie que b_1(17). Le nuage contient 190 000 points environ.

Ces deux scénarii étaient à réaliser successivement sur Cool et 3Dipsos. Or, la connaissance préalable d'un nuage de points 3D facilite et accélère sa compréhension et sa manipulation. Par conséquent, si l'on traite une même scène, la phase de compréhension de la scène qui a lieu avec le premier programme est (plus ou moins consciemment) utilisée par l'opérateur lors du traitement avec le deuxième programme. L'ordre des programmes testés sur une même scène introduit donc un biais, que l'on pourrait appeler «biais d'initié», à la fois en temps et en pénibilité. Aussi l'ordre d'utilisation des deux programmes a-t-il été alterné d'un utilisateur à l'autre afin d'éviter ce biais.

[2] En particulier, l'une des spécificités du mouvement de rotation dans 3Dipsos est que le mouvement est différent selon que la souris se trouve dans la zone centrale de la fenêtre ou dans les zones latérales.

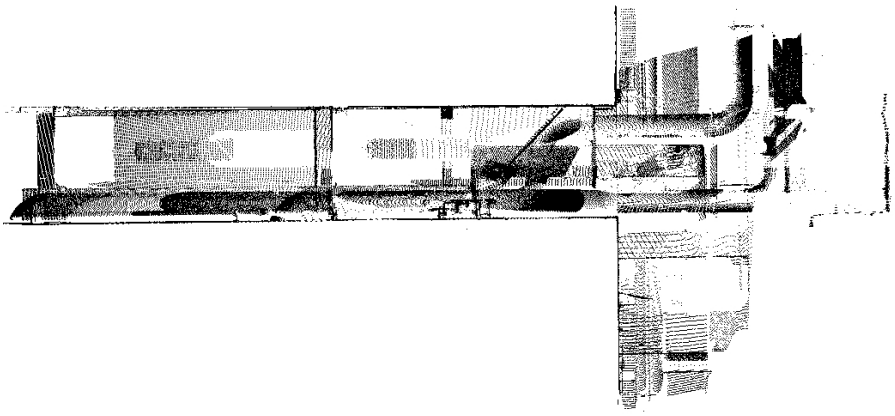


FIG. 3.21 – Scène b_1 de la phase de tests.

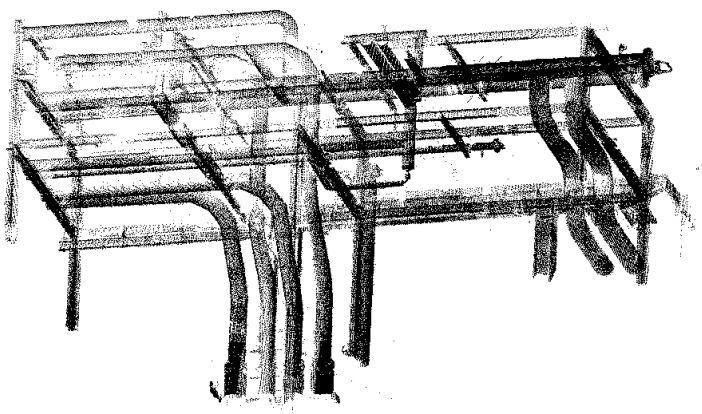


FIG. 3.22 – Scène b_2 de la phase de tests.

	Utilisateur 1	Utilisateur 2
Scène b_1	3Dipsos Cool	Cool 3Dipsos
Scène b_2	Cool 3Dipsos	3Dipsos Cool

Chaque scénario a été chronométré. Les résultats des segmentations sur 3Dipsos et sur Cool ont également été sauvegardées (pour Cool : sous-nuages, cylindres construits, points cliqués).

Réponse au questionnaire Enfin, pour chaque utilisateur, le test s'est terminé par une phase de discussion et par un questionnaire, reproduit à l'annexe F, qui a permis d'obtenir les réactions et suggestions des utilisateurs.

Résultats et analyse

Les résultats présentés ici sont une synthèse de ce qui a été observé lors des tests et des réponses au questionnaire qu'ont fourni les participants.

Qualité des résultats Les utilisateurs ont jugé les résultats de l'extraction dans Cool (Figures 3.23 et 3.24) exploitables en vue de la phase de modélisation qui suit dans 3Dipsos. Autrement dit, les résultats sont comparables en qualité à ceux obtenus avec une segmentation manuelle type dans 3Dipsos. Bien entendu, la qualité d'une segmentation manuelle dans 3Dipsos dépend du soin de l'utilisateur. Ce soin dépend d'ailleurs des habitudes des utilisateurs : certains réalisent une segmentation très propre, d'autres réalisent une segmentation plus grossière. De manière générale, il n'est pas toujours indispensable de produire une segmentation trop fine pour les traitements ultérieurs dans 3Dipsos. En particulier, ces traitements comprennent encore une partie de segmentation (séparation de chaque surface sur la ligne) [3].

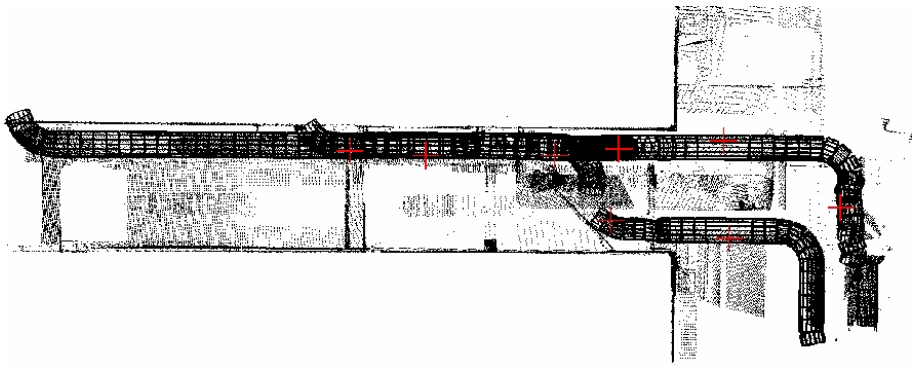


FIG. 3.23 – Résultat de l'extraction de Cool sur la scène b_1 (pour l'un des utilisateurs).

Durées observées Il paraît bien entendu intéressant, à terme, de comparer les performances en temps des deux procédés. En effet, l'objectif est bien d'augmenter la productivité de la reconstruction. Les temps de segmentation dans Cool et 3Dipsos ont donc été enregistrés, et donnent les résultats suivants (Les tests ont été effectués sur deux machines différentes : un PC Pentium III 733 MHz et un PC Pentium IV 1,13 GHz — dans chaque case, les deux premières durées correspondent à la première machine, et les deux dernières correspondent à la seconde machine) :

Temps pour les quatre utilisateurs (en minutes)

	3Dipsos	Cool
Scène b1	3 ; 5,2 ; 8 ; 7,5	3,5 ; 4,7 ; 6 ; 7,5
Scène b2	9 ; 17 ; 32 ; 14	12 ; 16 ; 13 ; 11,5

[3] Cette dernière segmentation a été simplifiée dans l'outil *smartline* de 3Dipsos. Cependant, tous les utilisateurs qui ont participé au test n'utilisent pas cet outil.

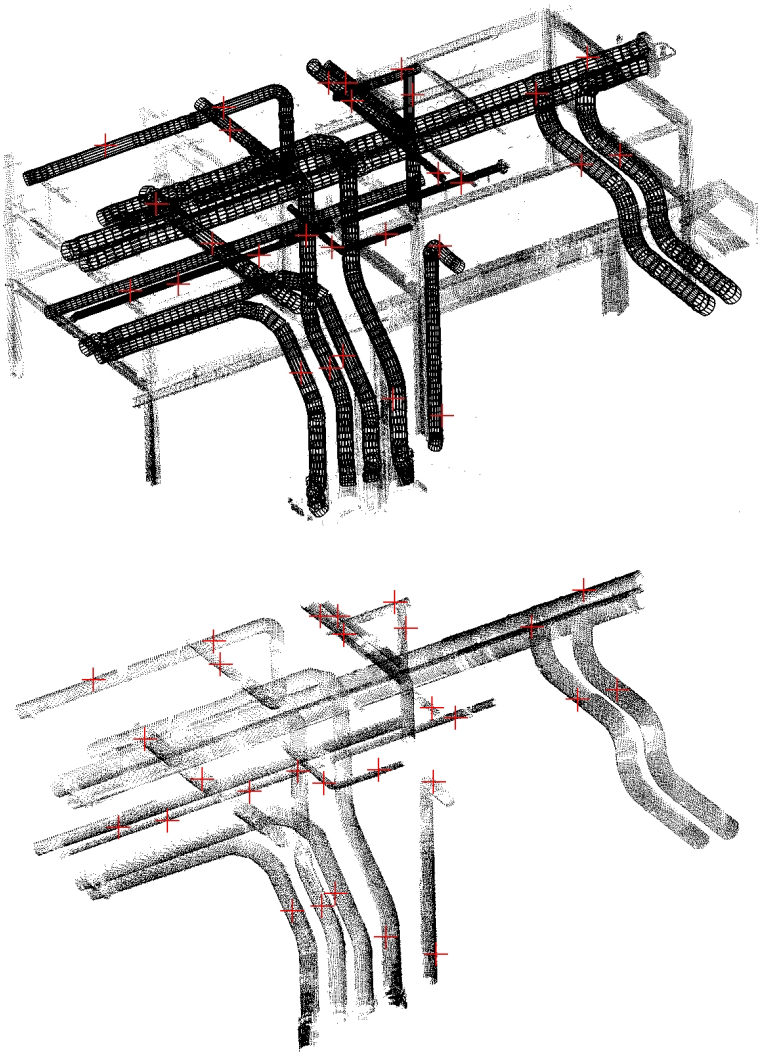


FIG. 3.24 – Résultat de l'extraction de Cool sur la scène b_2 (pour l'un des utilisateurs). En haut : les séquences de cylindres et le reste du nuage. En bas : ensemble des sous-nuages extraits. Les points où l'utilisateur a cliqué pour le cylindre initial sont également visibles (indiqués par des $+$).

Temps moyens sur les quatre utilisateurs (en minutes)

	3Dipsos	Cool
Scène b1	5,9	5,4
Scène b2	18	13,1

On constate tout d'abord que les durées moyennes d'extraction dans la version actuelle de Cool sont comparables, voire meilleures, à celles de l'extraction manuelle dans 3Dipsos (notons cependant que le temps 3Dipsos sur b_2 est très influencé par la donnée de 32 min, et que la moyenne des trois autres durées s'élève à 13,3 min).

On remarque ensuite que les durées de segmentation manuelle sont plus variées que les durées dans Cool.

Ecart-type sur les quatre utilisateurs (en minutes)

	3Dipsos	Cool
Scène b1	2,3	1,7
Scène b2	10,0 (4,0 sans 32)	2,0

Ceci était prévisible. De la même façon, on peut s'attendre à ce que les différences de durées de segmentation entre utilisateurs expert et utilisateurs occasionnels ou débutants soient atténuées avec cet outil. Notons à ce propos que la qualité de segmentation manuelle produite varie en fonction des utilisateurs : certains visent à produire une segmentation très propre, alors que d'autres produisent une segmentation plus grossière, mais qu'ils jugent suffisante pour les traitements ultérieurs. Aussi, les temps observés ne sont-ils pas tous les mêmes. Les différences entre durées mesurées sur 3Dipsos sont également révélatrices de différentes stratégies pour segmenter le nuage.

Enfin, il convient de garder à l'esprit que les données en temps mesurées ne sont pas de même nature : il s'agit d'un temps d'utilisateur dans 3Dipsos, pendant lequel l'utilisateur est concentré sur sa tâche, alors qu'il s'agit d'un temps utilisateur + logiciel dans Cool. Ceci signifie que l'optimisation des temps de calcul aura une incidence directe sur les temps de segmentation avec Cool, alors qu'avec 3Dipsos pas vraiment. Notons que les temps de calcul dans Cool n'ont pas été optimisés (par exemple, le temps d'accès aux points voisins est lourd, au sens où la complexité algorithmique est linéaire en le nombre total de points). Une fois optimisé, on peut donc escompter des gains en temps beaucoup plus importants. L'évolution des performances des machines va également dans ce sens.

Réactions des utilisateurs à l'application Les quatre utilisateurs considèrent que l'outil de segmentation interactive est moins pénible que le procédé de segmentation actuel. En particulier, les utilisateurs considèrent ce «pisteur» comme intéressant pour dégager un tuyau du nuage complet, et ce d'autant plus lorsque le nuage est compliqué.

Points jugés importants par les utilisateurs Quelques points que les utilisateurs experts ont considérés comme importants sont listés ici.

- Nombre de clics. L'algorithme ne doit pas demander à l'utilisateur de cliquer un trop grand nombre de fois pour extraire un même tuyau.
- Points extraits. Si le but est uniquement d'extraire la ligne, il vaut mieux que l'algorithme extraie trop de points que pas assez de points. Par exemple, en présence de brides, il vaut mieux que les points correspondants soient extraits. La raison de ceci est qu'il est plus coûteux en temps et plus pénible pour l'utilisateur de rechercher des points qui n'auraient pas été inclus que de les enlever une fois la ligne extraite.
- Elimination des points restants. L'algorithme actuel n'extrait pas toujours tous les points qui se trouvent sur la ligne de tuyauterie. En effet, le procédé utilisant les demi-sphères peut laisser des bandes de points à chaque pas de la propagation (Figure 3.35, page 81). Ceci est particulièrement le cas lorsque le rapport du bruit sur le rayon est important par rapport au facteur $\sqrt{2}$. Une fois une ligne extraite, il reste donc des points, ce qui peut dans certains cas donner l'impression à l'utilisateur que cette ligne n'a pas été extraite. Ceci est donc gênant pour la compréhension de la scène et de la progression de la segmentation. De plus, il peut être intéressant d'éliminer ces points pour le comportement lors de l'extraction d'une ligne voisine.

– Vitesse d'exécution, en particulier de l'extraction du cylindre initial. Si l'algorithme est trop lent, l'utilisateur s'ennuie ou a la sensation qu'il pourrait aller plus vite manuellement. Il est nécessaire, tant que l'intervention de l'utilisateur est requise, de maintenir une certaine vitesse de traitement.

Comportement intrinsèque de l'algorithme lors des tests Les tests ont également permis de mieux percevoir comment l'algorithme se comporte (sachant que l'utilisation d'un logiciel par la personne qui le développe est quelque peu biaisée !).

Concernant le cylindre initial, on a remarqué une instance de cylindre complètement mauvais en zone claire (*bug* résolu depuis), deux instances de cylindres avec rayons visuellement respectivement sur-estimé et sous-estimé.

Nombre moyen de points cliqués par tuyaux pour les deux scènes (moyenne des quatre utilisateurs)

	nb. moyen de pts cliqués	Nb de tuyaux	nb moyen de pts cliqués / tuyau
Scène b1	8,5	2	4,25
Scène b2	28,7	17	1,7

L'idéal serait bien sûr d'atteindre un clic par tuyau. Notons qu'il est assez différent de considérer des tuyaux longs et tortueux en environnement confiné comme dans la scène b_1 ou des tuyaux assez dégagés et droits comme la plupart de ceux de la scène b_2.

3.2.5 Développements suite aux tests

Comportement en présence d'ombre

Quelques développements ont été réalisés suite aux tests. En particulier, l'attention a été portée sur le nombre de points à cliquer pour extraire une ligne. Les clics multiples sont la conséquence d'arrêts inopinés de la propagation. Or, il se trouve que ces arrêts sont souvent dus à des ombres, créées par des objets proches qui étaient placés au premier plan vis à vis du scanner. C'est le cas par exemple des poutres servant de support aux lignes de tuyauterie(Figure 3.25). Dans ce genre de cas, la propagation

FIG. 3.25 – Exemple de cas où la version initiale de l'algorithme s'arrête à cause d'une ombre (due à la poutre qui sert de support à la ligne de tuyauterie). Dans le voisinage, il n'y a plus de points, et l'utilisateur est obligé de spécifier un point initial à nouveau.

s'arrête (car il n'y a pas de points dans la demi-sphère), alors que la ligne de tuyauterie continue. Le souhait clairement exprimé par les utilisateurs lors des tests est que l'algorithme puisse continuer (soit automatiquement, soit interactivement), sans que l'utilisateur doive sélectionner un point germe à nouveau.

L'idée développée pour traiter cette question a été de considérer une zone un peu plus grande (plus longue pour passer les ombres éventuelles, plus large pour gérer les changements de direction),

et de chercher un cylindre contraint dans cette zone. Comme il est alors peu probable que le tuyau soit la seule surface dans cette zone, on est amené à réaliser non plus un ajustement mais une extraction de cylindre. Contrairement à l'extraction du cylindre initial, on ne connaît pas de point qui se trouve sur la surface à extraire. Pour reprendre la terminologie du chapitre 6, il s'agit donc d'une extraction «globale» (même si celle-ci est réalisée sur un petite zone de la scène entière) de primitive(s) de type fixé.

De plus, on souhaite extraire un cylindre contraint, soit un cylindre-rotule, soit plus généralement un cylindre sécant de rayon fixé (coudes, tés égaux), soit un cylindre sécant (té, piquage, changement de diamètre). Deux solutions sont envisageables lorsque l'on souhaite extraire des primitives contraintes :

– soit extraire directement la primitive contrainte et vérifier la validité de ses paramètres,
– soit extraire une primitive libre et examiner si cette primitive est compatible avec la contrainte.

Ces deux voies ont été explorées pour le cas des cylindres sécants de rayon fixe. La première solution fait intervenir l'extraction globale d'une primitive de type fixé, dont le principe est décrit au 6.2, page 168. La primitive à extraire est dans ce cas directement un cylindre sécant de rayon fixe. Le résultat de cette extraction est montré figure 3.26 : sur le même exemple que figure 3.25, le cylindre (sécant de rayon fixe) extrait est proposé à l'utilisateur avant de continuer la propagation. La zone introduite est une demi-sphère de rayon $5r$. L'intérêt de cette solution est de mener au cylindre contraint

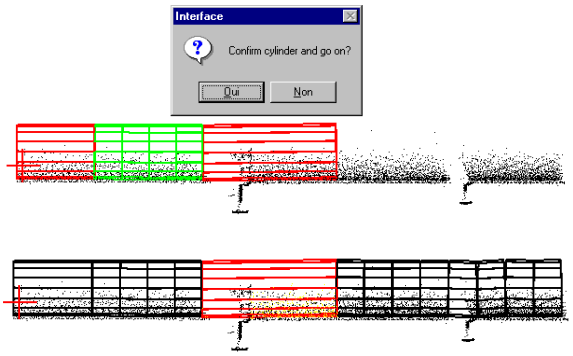


FIG. 3.26 – Adaptation de l'algorithme en présence d'ombres

directement. Par contre, la méthode d'extraction produisant des faux-positifs (voir section 6.2) : un faux cylindre peut être obtenu ici. En particulier, la méthode trouve un cylindre contraint la plupart du temps, même lorsque aucun cylindre n'est présent dans la zone.

La deuxième solution, encore à l'étude, met en œuvre l'extraction d'un nombre indéterminé de primitives de type fixé, dont le principe est décrit au 6.3, page 170. On produit donc d'abord une description de la zone par des primitives libres et l'on examine ensuite si l'une de ces primitives libres est compatible avec la contrainte. L'avantage de cette méthode est qu'elle permet d'éliminer facilement la plupart des fausses solutions issues de la première étape. En effet, lors de la première étape, le même problème se pose : on trouve toujours des faux positifs parmi les cylindres non-contraints extraits (Figure 3.27). Par contre, il est peu probable que ces faux positifs soient compatibles avec la contrainte. On peut donc espérer obtenir une faible probabilité de présence d'un faux positif au niveau du cylindre contraint obtenu.

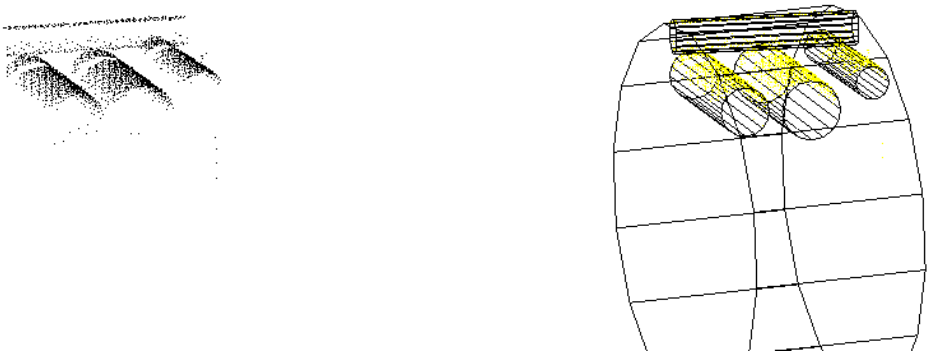


FIG. 3.27 – Résultat de l'extraction de cylindres libres

3.2.6 Discussion et perspectives

Concernant la construction du cylindre initial (étape 1)

Sur une zone suffisamment dense, le cylindre extrait est correct en pratique, notamment parce qu'il n'est pas crucial ici d'obtenir la taille optimale. Qu'entend-on par «suffisamment dense» ? On a vu que le procédé multi-résolution fait intervenir une taille minimale T_{min}. Ce paramètre T_{min} intervient ici. En effet, «suffisamment dense» signifie ici que les T_{min} plus proches voisins du point sélectionné se trouvent tous sur le cylindre. Dans ce cas, l'algorithme parvient en général à trouver la taille adéquate. Par contre, lorsque le point est situé sur une zone telle que ses T_{min} voisins ne se trouvent pas tous sur le même cylindre, l'algorithme ne peut pas trouver la bonne taille de voisinage, car les tailles cherchées sont plus grandes que T_{min}. Dans ces cas, le cylindre trouvé est faux. Une solution à ce problème consisterait à choisir T_{min} très petit. Or, l'extraction multi-résolution est plus délicate pour les très petites tailles. Pour plus de détails sur cette question, voir la discussion à la section 6.1.2, page 167.

En pratique, lorsque la zone est scannée de près, ceci ne pose de problème que pour les très petits diamètres de tuyaux. Par contre, si le cylindre se trouve dans une zone très peu dense en points (zone qui se trouvait très loin du scanner pendant l'acquisition) et que d'autres surfaces sont très proches du cylindre (d'autres tuyaux par exemple), alors ce problème peut se produire. Ceci est fonction du rapport entre le nombre de points et le nombre de surfaces présents dans le même volume. Cependant, on peut considérer que la plupart du temps, les zone d'intérêt, où l'on souhaite effectuer une reconstruction précise de l'environnement, ont été scannées correctement.

Outre cette limitation par rapport aux zones de très faible densité et où de nombreuses surfaces sont présentes, la propriété suivante est valable sur toute la scène : plus le point spécifié par l'utilisateur (clic) est éloigné des autres surfaces (et des éventuels points aberrants), et meilleure est l'estimation du cylindre initial (puisque le voisinage final contient davantage de points).

Concernant la propagation (étape 2)

Lorsqu'il n'y a pas, ou lorsqu'il y a peu d'éléments perturbateurs le long de la ligne de tuyauterie, l'algorithme parvient à suivre la ligne correctement. En particulier, les parties coudées sont la plupart du temps bien gérées. Ceci est le cas même lorsque le coude est serré, c'est-à-dire quand ses grand et petit rayon sont égaux, comme l'illustre la figure 3.28. Ceci est une conséquence du choix de longueur

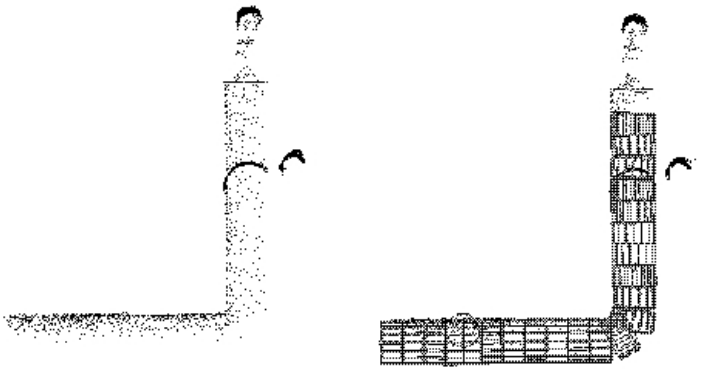


FIG. 3.28 – Exemple de situation comprenant un coude «serré»

des cylindres fixée égale au rayon.

Concernant l'arrêt de la propagation, on remarque expérimentalement que le comportement est variable. Ceci est illustré sur la figure 3.29, où l'extraction ne s'arrête pas (vers le bas de la figure) aux mêmes endroits pour les quatre lignes de tuyauterie, pourtant pratiquement identiques : pour la ligne située à gauche, la propagation continue au niveau de la bride et du changement de diamètre (calorifuge) sans détecter que les cylindres ne représentent plus les points de manière satisfaisante. Pour la ligne située immédiatement à sa droite, la propagation s'arrête au niveau de la bride et du changement de diamètre, ce qui est plus logique dans la mesure où l'on se limite pour l'instant aux parties ayant un diamètre fixé.

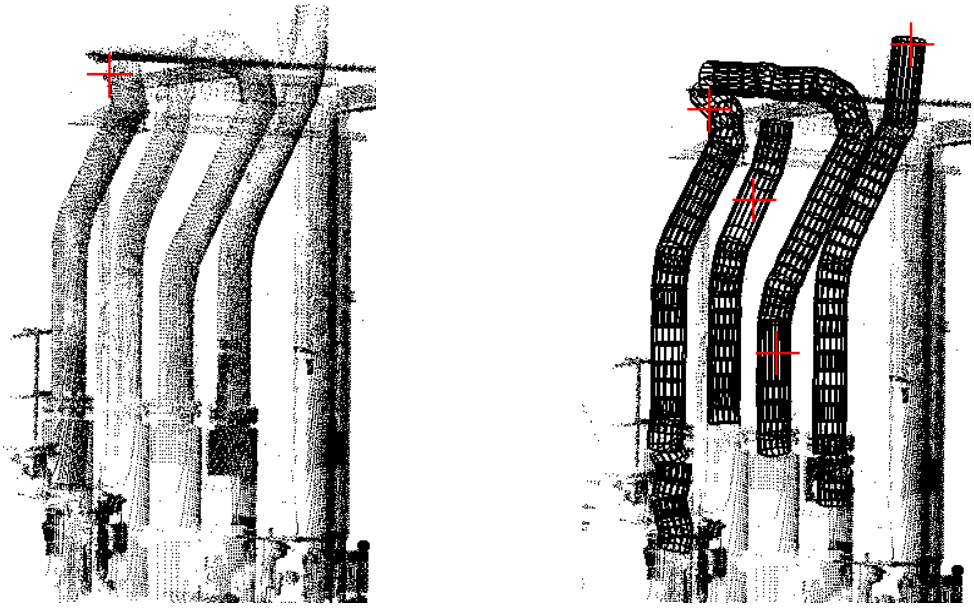


FIG. 3.29 – Sur les quatre tuyaux similaires, la propagation s'est stoppée différemment.

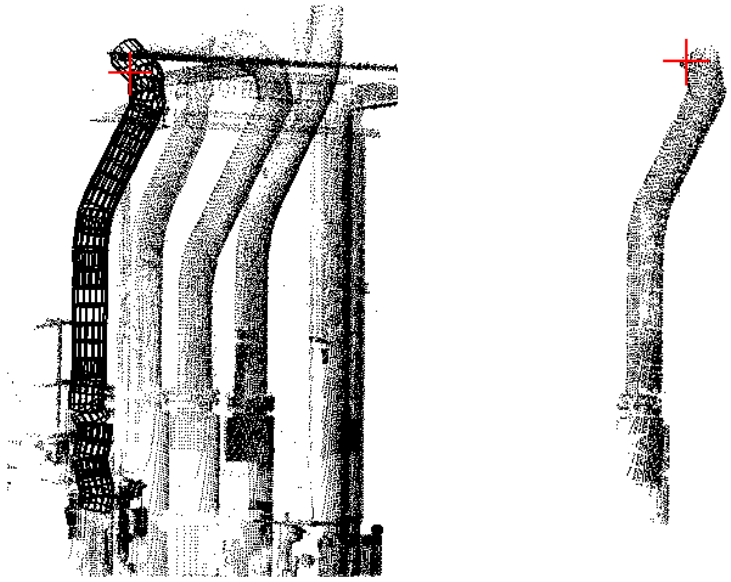


FIG. 3.30 – Exemple «Elf4». On voit ici que la condition d'arrêt actuelle n'est pas satisfaisante.

En réalité, deux situations distinctes se présentent :

– des éléments mineurs se trouvent sur la ligne (brides, attaches, poutre, etc.),
– la fin de la ligne de tuyauterie de section constante a été atteinte (bride finale, réduction, calorifuge, mur, ...).

Le comportement de l'algorithme actuel en présence de ces deux types d'éléments est variable. L'une des raisons est que selon le diamètre du tuyau et la forme de l'élément, mais aussi de la densité de points, les points dans le voisinage contiennent plus ou moins d'information sur l'élément perturbateur. Par exemple, comme cela a été dit section 3.1.3, page 45, les brides peuvent avoir des tailles différentes et les tailles possibles ne varient pas linéairement avec le diamètre du tuyau, alors que le rayon de la demi-sphère de voisinage est proportionnel au diamètre de la tuyauterie. Ceci signifie que, selon les cas, les points se trouvant sur des brides seront englobés ou non lors de la création du nouveau voisinage.

Dans le cas d'un élément mineur, c'est-à-dire un élément perturbateur court le long de la ligne, le comportement souhaité est de continuer la propagation (éventuellement en avertissant l'utilisateur) ; dans le second cas, celui d'un changement de diamètre ou de la fin de la ligne, le comportement souhaité est de stopper la propagation (puisque l'algorithme ne traite actuellement pas les changements de diamètre).

Ces deux situations illustrent toutes deux le besoin d'un meilleur contrôle local de la forme au niveau du critère d'arrêt.

Nous avons vu que le critère actuel décidant de l'arrêt de la propagation examine

1. si le nombre de points dans le voisinage est suffisant, et
2. si «on se trouve toujours sur le tuyau», c'est-à-dire si le cylindre construit est une description satisfaisante des points du voisinage.

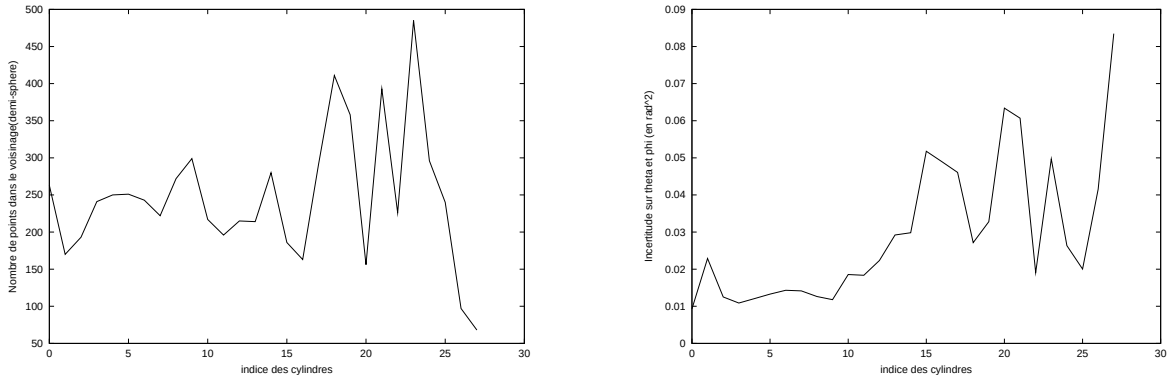


FIG. 3.31 – Graphe du nombre points (à gauche) et graphe de l'incertitude sur les paramètres de direction de cylindres (à droite) sur l'exemple «Elf4»(Figure 3.30)

Critère sur le nombre de points dans le voisinage Le critère sur le nombre de points est quelque peu arbitraire. Certes, le paramètre $n_{\min}$ a été fixé à une valeur très faible (10), pour laquelle ajuster un cylindre n'a pratiquement plus de sens. Une alternative à ce critère sur le nombre de point consiste à considérer l'incertitude sur l'estimation des paramètres du cylindre (sur les angles θ et ϕ pour le cylindre-rotule). L'algorithme utilisé pour l'ajustement des primitives fournit en effet une estimation de l'incertitude sur les paramètres. Voir à ce propos la section 4.2.2, page 101. Cette incertitude est liée au nombre de points : plus le nombre de points est faible, plus cette incertitude est grande. Cependant, l'incertitude est sans doute plus cohérente que le nombre de points dans la mesure où une différence est faite entre des points qui définissent précisément un cylindre et des points qui définissent peu précisément un cylindre. Ceci dépend du bruit sur les points, mais aussi de la répartition (en particulier angulaire) des points sur le cylindre. Se reporter à la figure 3.31, qui présente le graphe de l'incertitude sur les paramètres θ et ϕ des cylindres contraints le long de la ligne extraite dans l'exemple «Elf4» de la figure 3.30. On voit en particulier que l'incertitude dépend du nombre de points mais pas uniquement (cela dépend également de la répartition des points). On a vu au 3.2.5 que l'on peut, dans ces situations, remplacer l'arrêt pur et simple de la propagation par une extraction pour trouver un nouveau cylindre initial.

Critère d'arrêt sur l'erreur On a vu que le critère d'arrêt de la propagation se fonde simplement sur une comparaison entre l'erreur d'ajustement e_i du cylindre courant et l'erreur d'ajustement e_0 du cylindre initial :

$$\text{Arrêt si } (e_i > \beta e_0)$$

Ce critère n'est pas satisfaisant, et ce pour deux raisons essentielles. D'abord il donne trop d'importance à l'ajustement du cylindre initial à travers e_0. Ensuite, la valeur du paramètre β est arbitraire (pourquoi choisir 10 et pas 5 ?). Or, ce paramètre est critique, au sens où l'algorithme s'arrête trop facilement ou pas assez suivant sa valeur. Enfin, l'erreur e_i fait intervenir une moyenne sur le nombre de points présents dans la demi-sphère courante. Or, ce nombre de points, qui dépend de la densité locale, n'est pas constant le long de la ligne. Le caractère aléatoire associé à l'estimation e_i est d'autant plus important que le nombre de points est petit.

Plusieurs idées ont été envisagées pour définir un meilleur critère d'arrêt.

Définition d'un intervalle de confiance Tout d'abord, on peut remplacer le paramètre β par un facteur plus cohérent, issu d'une approche statistique, qui prend en compte le nombre de points courant.

Rappelons que l'on estime les erreurs $e_i (1 \leq i \leq r)$, définies comme étant le résidu quadratique moyen par rapport au cylindre courant Cyl_i :

$$e_i = \sqrt{\frac{1}{n_i}\sum_{k=1}^{n_i} d^2(\mathbf{x_{i,k}}; \text{Cyl}_i)} = \sqrt{\frac{1}{n_i}\sum_{k=1}^{n_i} d_{i,k}^2} \tag{3.1}$$

où $\{\mathbf{x_{i,k}}\}_{1\leq k\leq n_i}$ désigne les points associés au cylindre numéro i, et n_i désigne le nombre de ces points.

La quantité e_i^2 étant la moyenne d'une partie des termes $d_{i,k}^2$, le théorème de la limite centrée nous indique que la variable e_i^2 tend en loi, lorsque n_i tend vers l'infini, vers la loi normale $\mathcal{N}(\mu, \sigma^2)$, où μ est l'espérance et σ^2 est la variance des $\{d_{i,k}^2\}_{1\leq k\leq n_i}$. Par conséquent, pour n_i grand (en pratique supérieur à 30) [4] , on peut écrire l'intervalle de confiance α sur l'estimation e_i^2 comme suit

$$\text{Prob}(e_i^2 \in [\mu - k_\alpha \frac{\sigma}{\sqrt{n_i}}; \mu + k_\alpha \frac{\sigma}{\sqrt{n_i}}]) = \alpha$$

où k_α est issu d'une table de loi normale $\mathcal{N}(0,1)$. Par exemple, pour $\alpha = 0,95$ (95%), on trouve $k_\alpha = 1,96$.

Cependant, les paramètres μ et σ^2 de la loi des $\{d_{i,k}^2\}_k$ sont ici inconnus. Or, tant que l'algorithme de propagation fournit des cylindres cohérents avec les points, la population $\{d_{i,k}\}_{1\leq i\leq r, 1\leq k\leq n_i}$ peut être considérée comme homogène. Plus précisément, les observations $\{d_{r,k}\}_{1\leq k\leq n_r}$ du modèle local ont la même loi statistique que les observations $\{d_{i,k}\}_{1\leq i\leq r-1, 1\leq k\leq n_i}$. Par conséquent, les $d_{i,k}^2$ peuvent être considérées comme les réalisations de variables aléatoires indépendantes, identiquement distribuées (c'est-à-dire de même loi). Au cylindre numéro i ($i \geq 2$), l'espérance et la variance peuvent alors être estimées à l'aide des moyennes et variances empiriques sur tous les points des cylindres précédents :

$$\mu_i = \frac{1}{\sum_{1\leq j<i} n_j} \sum_{1\leq j\leq i} \sum_{1\leq k\leq n_j} d_{j,k}^2$$

et

$$\sigma_i^2 = \frac{1}{\sum_{1\leq j<i} n_j} \sum_{1\leq j\leq i} \sum_{1\leq k\leq n_j} (d_{j,k}^2 - \mu_i)^2$$

On peut ensuite remplacer l'expression de l'intervalle de confiance sur e_i^2 par

$$\forall i \geq 2, \text{Prob}(e_i^2 \in [\mu_i - k_\alpha \sqrt{\frac{\sigma_i^2}{n_i}}; \mu_i + k_\alpha \sqrt{\frac{\sigma_i^2}{n_i}}]) = \alpha$$

Ainsi, un contrôle possible de l'ajustement du cylindre numéro i consiste à tester si e_i^2 est bien dans l'intervalle de confiance $[\mu_i - k_\alpha \sqrt{\frac{\sigma_i^2}{n_i}}; \mu_i + k_\alpha \sqrt{\frac{\sigma_i^2}{n_i}}]$. La figure 3.32 illustre ce test sur l'exemple «elf4». On peut y remarquer des déviations assez franches au niveau du changement de diamètre visible en bas de la ligne, figure 3.30. Cependant, on remarque également que même en début de ligne, où les modèles sont acceptables, les e_i^2 ne sont pas toujours dans l'intervalle de confiance. Ceci conduit à augmenter le facteur k_α, ce qui revient à introduire un facteur multiplicatif, dont on ne s'affranchit donc pas totalement. Enfin, une critique que l'on peut faire à cette approche est qu'elle inclut tous les points précédents dans l'estimation de μ_i et σ_i, dont les points correspondants aux modèles qui n'étaient pas très bons.

[4] Pour des tailles n_i plus petites, il est nécessaire de faire des hypothèses sur la loi des d_i. Si les d_i suivent une loi normale, on pourra consulter une table de loi du χ^2.

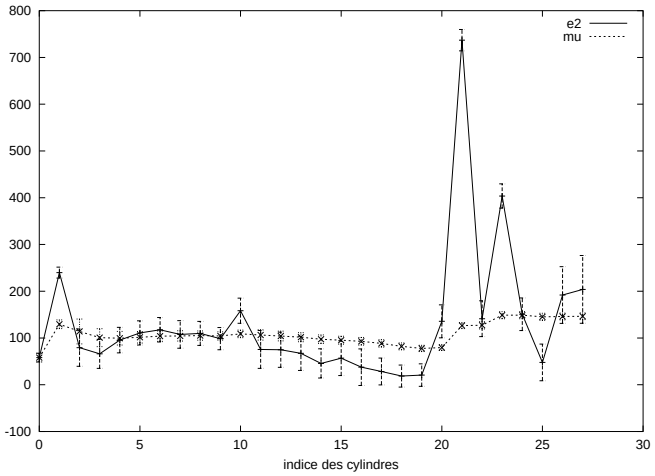


FIG. 3.32 – Contrôle de l'erreur quadratique par intervalles de confiance (sur l'exemple «Elf4» de la figure 3.30). La figure comprend le graphe de μ_i ainsi que celui de e_i^2 entouré de l'intervalle de confiance $[e_i^2 - k_\alpha\sqrt{\frac{\sigma_i^2}{n_i}}; e_i^2 + k_\alpha\sqrt{\frac{\sigma_i^2}{n_i}}]$ (ici avec $\alpha = 95\%$, c'est-à-dire $k_\alpha = 1,96$)

Analyse plus fine de la qualité d'ajustement des cylindres En outre, l'utilisation de l'erreur d'ajustement comme seule information sur la qualité d'approximation des points par le cylindre est également discutable. En effet, l'erreur d'ajustement traduit le niveau moyen des écarts des points par rapport au modèle. Or, cette information ne suffit pas toujours pour quantifier la qualité d'un ajustement : deux situations totalement différentes peuvent produire des erreurs d'ajustement identiques (voir chapitre 5, l'exemple de la figure 5.17, page 144, et l'exemple de la figure 5.9, page 137). Ainsi, l'erreur d'ajustement est une information importante mais partielle sur la manière dont la primitive représente les données. D'autres moyens de décrire la qualité d'un ajustement, ou d'évaluer un ajustement, ou encore de valider ou non un modèle, ont donc été étudiés (voir section 5.2). Parmi ceux-ci, le ratio obtenu par lissage à partir du modèle (ici le cylindre), présenté en détails section 5.2.6, page 152, donne des résultats prometteurs. La figure 3.33 montre le graphe de ce ratio pour chaque cylindre le long de la ligne de tuyauterie extraite dans la scène «Elf4» de la figure 3.30. On perçoit clairement un pic au niveau du changement de diamètre. Notons qu'ici les valeurs de ce ratio sont indépendantes entre elles : il s'agit simplement d'un contrôle du cylindre i, indépendamment des cylindres précédents.

Enfin, l'introduction de plusieurs modèles, avec sélection, peut également aider à résoudre le problème (dans l'exemple elf4, on trouverait un cylindre plus gros...).

Alternative à l'arrêt

Ceci rejoint un autre point, qui consiste à dire qu'en cas d'erreur importante, on peut, plutôt que de stopper la propagation, passer à une phase d'extraction comme celle réalisée en présence d'ombres (section 3.2.5).

En effet, l'un des points mis en exergue par les utilisateurs lors des tests est qu'il faut éviter, autant que faire se peut, de sélectionner à nouveau un point initial. Il s'agit donc de faire en sorte que l'algorithme s'arrête le moins possible. Ceci concerne les ombres comme cela a été vu au 3.2.5, mais en fait également les éléments perturbateurs assez courts : brides, poutres, mais aussi vannes, clapets,

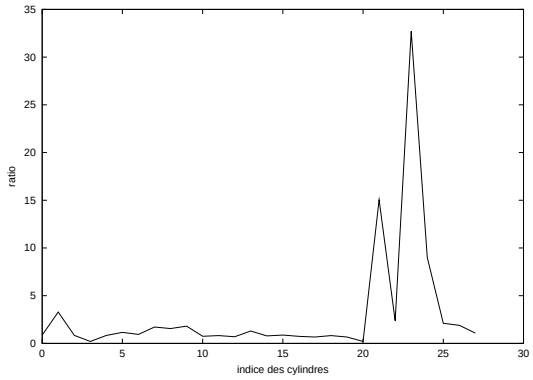


FIG. 3.33 – Analyse de la qualité de chaque cylindre (approche par lissage sur l'exemple «Elf4»)

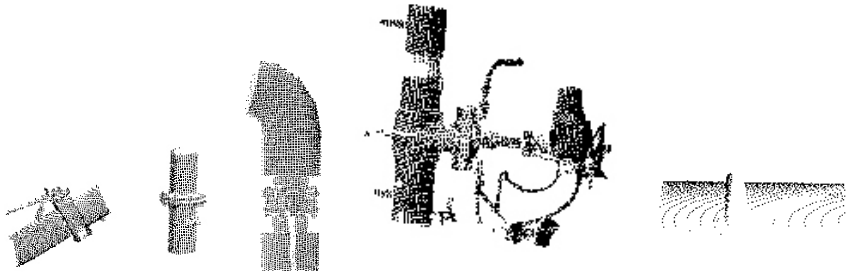


FIG. 3.34 – Exemple d'éléments perturbateurs (brides, robinet, pompe, attache) en présence desquels il est souhaitable que la propagation continue (car le même tuyau se trouve des deux côtés)

pompes, tés, etc. En effet, il existe de nombreuses situations où il est intéressant de continuer : après une bride, une poutre, une attache, mais aussi après une vanne ou une pompe, etc.

Il peut donc être intéressant d'étendre l'action effectuée en cas d'ombres à d'autres cas (Figure 3.34), par exemple dès qu'il y a un doute sur la forme locale. L'algorithme réaliserait ainsi du «saut d'obstacle». Notons que cela ne change pas le problème du critère d'arrêt : l'arrêt est simplement remplacé par une extraction (avec ensuite arrêt ou continuation).

Du reste, pour les ombres créées par des poutres comme dans l'exemple de la figure 3.25, il semble cohérent de souhaiter que l'algorithme ait un comportement similaire dans un sens comme dans l'autre. Or, dans un sens on arrive d'abord sur l'ombre puis sur la poutre, et dans l'autre sens on arrive d'abord sur la poutre puis sur l'ombre.

Points restants

Lors de la propagation, les points qui se trouvent sur la ligne de tuyauterie ne sont pas nécessairement tous extraits à l'issue de l'algorithme. Plus précisément, les voisinages hémisphériques présentent l'inconvénient de ne pas englober tous les points, comme l'illustre la figure 3.35. Ceci est d'autant plus le cas lorsque les points sont très bruités, car les points forment une certaine épaisseur. Les points qui se trouvent dans cette épaisseur et entre deux demi-sphères ne sont pas pris en compte lors de la propagation. Or ceci peut s'avérer gênant car il n'est pas clair si ces points sont sur une ligne

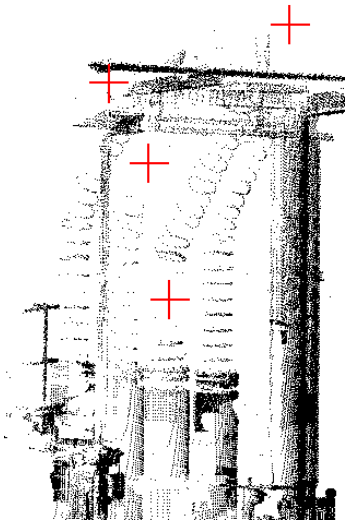


FIG. 3.35 – Points restants après extraction des quatre tuyaux de la figure 3.29.

déjà extraite ou non. Ceci gène la visualisation et la compréhension de la scène, mais peut aussi poser problème pour l'algorithme de propagation sur une ligne de tuyauterie connexe à celle précédemment extraite.

Ce point a été souligné par certains utilisateurs lors des tests. L'un des moyens que l'on peut envisager pour «nettoyer» la ligne de tuyauterie des points restants est de rajouter tous les points se trouvant à une distance inférieure à un certain seuil de la ligne de tuyauterie. On est donc amené à définir la distance à une séquence de cylindres, qui peut être perçue comme une primitive composée. La section 4.5.5, page 123 traite de la définition de telles primitives.

Limitations

La version actuelle de l'algorithme ne gère pas les changements de diamètre. L'idée est pour l'instant que la propagation doit s'arrêter aux changements de diamètre. Ces changements de diamètres peuvent être pris en compte, soit en introduisant une primitive «cône-rotule», soit plus simplement en relâchant le rayon dans la primitive «cylindre-rotule».

D'autre part, en présence de tés, on peut souhaiter que la propagation s'effectue des deux côtés ! ! Pour l'instant, la propagation suit l'une des directions ou s'arrête.

Perspectives d'intégration de l'application

Il est prévu d'intégrer l'algorithme existant comme outil complémentaire à la segmentation manuelle (dans RealWorks Survey). L'intégration nécessite une ergonomie améliorée, en vue de baisser la pénibilité. Des adaptations sont alors envisageables dans ce contexte. Par exemple, la possibilité de définir plusieurs méthodes pour le cylindre initial : un clic, deux clics, polygone élastique, etc.

L'algorithme que nous avons présenté est perçu comme utile pour faciliter l'étape d'extraction, où l'on extrait le sous-nuage correspondant à la ligne de tuyauterie. Cependant, on voit clairement que l'algorithme fournit plus que le nuage de points seul. En effet, la séquence de cylindres liés fournit un modèle approximatif, qui peut aider à réaliser l'étape suivante : la modélisation de la ligne.

3.3 Modélisation d'une ligne de tuyauterie

Cette section présente deux applications développées en vue de réaliser la segmentation-modélisation finale [5] . La première propose de construire directement le modèle CAO final comprenant des cylindres et des tores à partir du sous-nuage de points. La section 3.3.1 présente les résultats et limitations de cette approche. En particulier, l'un des problèmes auxquels nous nous sommes confrontés est la sensibilité au choix du point initial, ainsi qu'au sens de propagation.

La deuxième application, exposée section 3.3.2, a un objectif moins ambitieux : il s'agit de partir de la séquence de cylindres pour produire une segmentation suivant les surfaces présentes le long de la ligne.

3.3.1 Modélisation directe avec cylindres et tores

L'objectif est ici de segmenter les points d'une ligne de tuyauterie en plusieurs primitives géométriques, afin de construire un modèle plus proche du modèle CAO final que celui construit dans l'algorithme présenté au 3.2.

Nous nous sommes pour l'instant limités au cas de lignes de section constante. Ceci signifie que les surfaces présentes sont des cylindres et des portions de tores circulaires.

D'autre part, l'objectif final est qu'il y ait une continuité le long de la ligne. Ceci implique que les primitives se suivent et que chaque primitive a une longueur déterminée. Plus précisément, la jonction entre les primitives a ici plus d'importance que pour la méthode d'extraction du 3.2.

Les spécificités de cette méthode par rapport à la méthode présentée section 3.2 sont donc de deux types :

- différentes primitives sont ajustées (ce qui nécessite un critère de sélection entre les deux primitives),
- les extrémités de chaque primitive sont ajustées (ce qui est effectué par une méthode de croissance de surface).

Notons pour finir que l'application actuelle a été développée pour traiter des scènes comportant une ligne dégagée. En particulier, ceci permet de recourir à des méthodes plus simples en ce qui concerne la croissance et l'arrêt de la propagation.

Le principe général de l'application développée est le même que celui de l'application présentée à la section 3.2.

Détail des différentes étapes

La construction du cylindre initial est identique à celle de la méthode présentée au 3.2. Par contre, il s'agit ensuite de déterminer les extrémités du cylindre initial sur la ligne de tuyauterie. En particulier, on a vu au 3.2 que le cylindre initial obtenu n'avait en général pas les dimensions maximales. C'est pourquoi on applique une croissance au cylindre initial. Le principe de la croissance est le suivant.

[5] Le premier algorithme a été développé avec un étudiant de l'option Robotique (Sébastien Vincent) lors de son projet de fin d'études en 1999-2000. Cet algorithme a donc été développé avant l'algorithme d'extraction de lignes de tuyauterie. Les problèmes dont il est question plus loin ont fait que nous n'avons pas jugé souhaitable d'adapter cet algorithme à un environnement complet.

Le second algorithme a été développé plus récemment avec une stagiaire provenant de l'Université du Liban à Tripoli (Nouha Mourad).

Croissance du cylindre initial La phase de croissance consiste à rajouter les points qui se trouvent à une distance de ce cylindre inférieure à un certain seuil δ. Ceci revient à considérer tous les points de la scène qui se trouvent à l'intérieur d'un manchon cylindrique d'épaisseur 2δ (Figure 3.36).

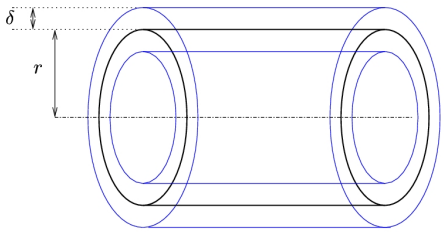


FIG. 3.36 – Croissance du cylindre

Le premier point est de savoir quelle valeur donner à ce paramètre δ. Or, la construction du cylindre initial a permis de trouver un ensemble de n points, ainsi qu'un cylindre $\mathcal{C}$ ajusté sur cet ensemble de points, dont on peut évaluer la qualité d'ajustement par le résidu quadratique moyen e, défini par l'équation 3.1. L'idée est alors de choisir une épaisseur proportionnelle à e :

$$\delta = ke$$

De plus, si l'on suppose que les écarts radiaux d_i suivent une loi normale centrée[6], $\mathcal{N}(0, e^2)$, les d_i/e suivent une loi $\mathcal{N}(0,1)$. On peut alors choisir un k dans une table de loi normale tel que la probabilité qu'un point soit dans le manchon soit de 95%, ou 98% ($k_{0.95} = 1,96$ pour 95%). Ceci revient à choisir un manchon cylindrique d'épaisseur $\delta = k_{0.95}e$.

Le second point consiste à déterminer la longueur du manchon cylindrique. Dans l'application développée, où l'on suppose que la ligne est seule, cette longueur dépend uniquement des points se trouvant dans le manchon.

Propagation

Création du nouveau voisinage La création de voisinage est identique à celle dans l'algorithme d'extraction de la section 3.2. Le principe est de considérer les points dans un volume dans la direction souhaitée. On peut choisir pour cela une demi-sphère comme pour l'algorithme d'extraction précédent. Cependant, le résultat de la croissance étant un peu approximatif (d'autant plus dans le cas de jonctions lisses comme une jonction «tube-coude»), on va souvent un peu trop loin. Ceci nous mène à envisager plutôt un voisinage isotrope, c'est-à-dire une sphère.

Ajustement Dans sa version actuelle, l'algorithme met en œuvre l'ajustement de surfaces contraintes de deux types :

– «Tore après cylindre» (Figure 3.37),
– «Cylindre sécant de rayon fixé» (Figure 3.38),

dont nous donnons une brève description ici.

[6] l'espérance des d_i peut être nulle car la distance d d'un point au cylindre est signée : elle est positive si le point est à l'extérieur, et négative si le point se trouve à l'intérieur. Il en est de même pour le tore. Ceci est décrit à la section 4.3.2.

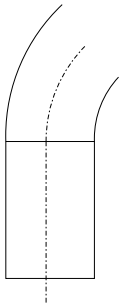


FIG. 3.37 – «Tore après cylindre»

Un «tore après cylindre» est un tore qui continue de manière tangentielle un cylindre donné (son petit rayon est fixé et l'un des points du cercle directeur également).

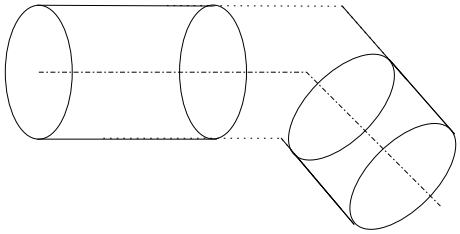


FIG. 3.38 – «Cylindre sécant de rayon fixé»

Un «cylindre sécant de rayon fixé» est un cylindre dont l'axe est sécant avec l'axe d'un autre cylindre et de même rayon (Figure 3.38).

La définition ainsi que la méthode d'ajustement de ces deux primitives sont présentées en détails au chapitre 4. Retenons ici que l'ajustement de ces primitives nécessite l'estimation de la normale à la surface en chaque point.

Pour aider à visualiser comment se comportent ces primitives, on peut caractériser la position de ces primitives par rapport à l'axe de la primitive précédente par des types de liaison dynamique utilisées en mécanique. Ainsi, un «tore après cylindre» forme une liaison de type «pivot glissant» ; et le «cylindre sécant» une liaison de type «glissière + rotule» par rapport à l'axe du cylindre précédent. Ceci signifie qu'une certaine souplesse est autorisée sur la position de ces primitives le long de l'axe du cylindre précédent. On parle ici seulement de la position est non de la forme (taille) des primitives.

La sélection entre ces deux modèles se fait sur le critère suivant :

- après un tore, on ajuste un cylindre sécant avec le cylindre précédant le tore
- après un cylindre sécant, on ajuste un «tore après cylindre». S'il se trouve que le tore obtenu ne semble pas réellement être un tore (en pratique si $R > 20r$), on le remplace par un cylindre sécant.

Pourquoi ce critère $R > 20r$? La lecture de tables de coudes standards [TC89] semble indiquer une valeur limite de cet ordre sur le ratio grand rayon sur petit rayon des coudes.

D'autre part, afin d'améliorer l'ajustement d'une séquence «cylindre-tore-cylindre» déjà construite, la primitive «tore entre cylindres» a été introduite (Figure 3.39). Cette primitive est un tore qui fait la jonction entre un cylindre et un «cylindre sécant de rayon fixé». Après l'ajustement de ce «tore entre

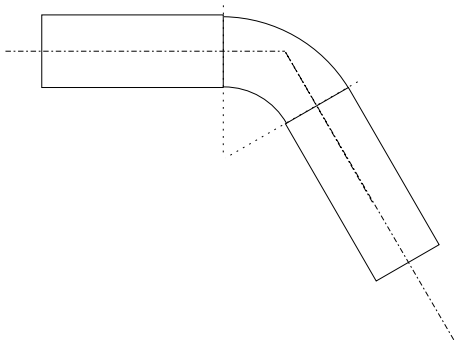


FIG. 3.39 – Séquence cylindre-tore-cylindre

cylindres», l'affectation des points à chaque primitive est mise à jour (voir la définition précise de cette primitive au chapitre 4).

Croissance

Cas du cylindre La méthode de croissance du cylindre sécant est la même que celle décrite pour le cylindre initial, excepté le fait qu'elle est effectuée dans une seule direction (direction de propagation).

Cas du tore La méthode de croissance du tore est l'équivalent de celle du cylindre, mais au niveau angulaire. En effet, on ajoute de la même façon les points qui se situent à une distance inférieure à *ke* du tore.

Critères d'arrêt de la propagation De la même manière que pour l'algorithme d'extraction, les itérations précédentes s'arrêtent :

- si on ne trouve plus assez de points dans le voisinage, ou/et
- si l'erreur d'ajustement est trop élevée. Plus précisément, si le résidu courant est supérieur au résidu du cylindre initial multiplié par un certain facteur (égal à 10 en pratique).

Résultats et discussion

Les figures 3.40, 3.41, et 3.42 montrent quelques résultats de cet algorithme.

La discussion concernant la construction du cylindre initial a déjà été présentée à la section 3.2.6, page 74.

Concernant la propagation, les résultats ont permis de valider la méthode d'ajustement des primitives tore et cylindre contraints, ainsi que la possibilité de réaliser la modélisation avec cylindres et tores contraints de cette façon.

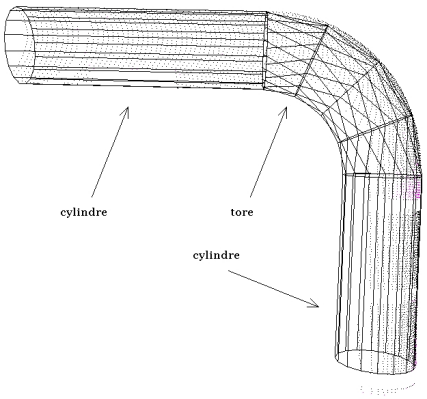


FIG. 3.40 – Ligne de tuyauterie reconstruite à l'aide de cylindres et de tores

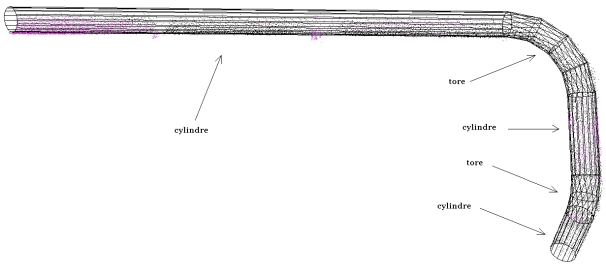


FIG. 3.41 – Résultat sur une scène constituée uniquement de cylindres

Cependant, on constate expérimentalement que la méthode est très sensible au point sélectionné au départ. En effet, le résultat obtenu avec un point cliqué à un bout de la tuyauterie est rarement le même que le résultat obtenu avec un point cliqué à l'autre bout. Aussi la qualité du résultat est-elle très variable, comme l'illustre la figure 3.42. En particulier, il existe même des cas où on ne parvient pas à suivre la ligne en entier.

Cette instabilité a plusieurs origines.

1. La qualité du cylindre initial dépend du point sélectionné : s'il s'agit d'une zone où la partie cylindrique est longue, le cylindre initial est estimé sur un grand nombre de points, et on peut attendre une estimation précise de ses paramètres ; s'il s'agit d'un cylindre court, les paramètres sont estimés avec moins de précision.
2. Les contraintes utilisées sont fortes, et ceci du fait même de la définition des primitives. Ceci a pour conséquence que l'algorithme peut difficilement «rattraper» des écarts sur une primitive lors de la suite de la propagation. Au contraire, les erreurs tendent à se propager en s'amplifiant le long de la ligne de tuyauterie.
3. Enfin, la croissance est la plupart du temps imparfaite. D'une part, on ajoute des points par rapport à une primitive estimée localement, ce qui produit potentiellement des erreurs. D'autre part, comme cela a été mentionné au chapitre 2, la jonction entre tube et coude est une jonction lisse. Ceci implique que la croissance d'un cylindre sur le coude qui suit déborde sur le tore,

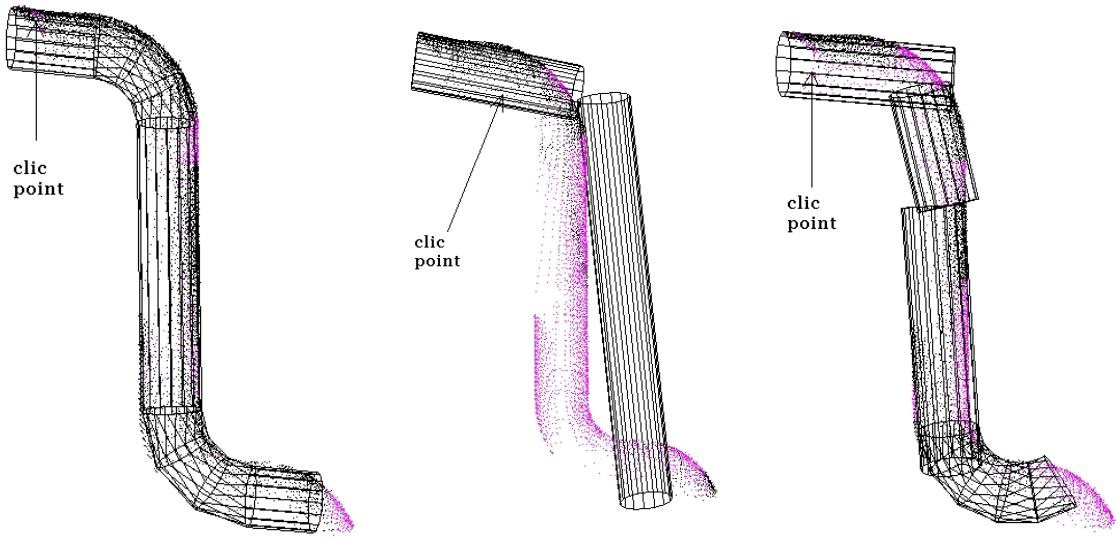


FIG. 3.42 – Caractère instable de l'algorithme de modélisation : selon le point de départ spécifié, le résultat peut être fort différent.

comme l'illustre l'exemple de la figure 3.43. Il résulte de ces erreurs de position que le voisinage construit pour la primitive suivante n'est pas très bon.

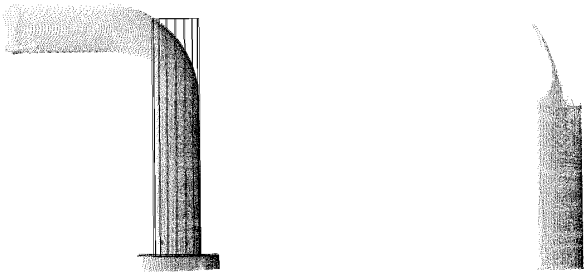


FIG. 3.43 – Exemple de croissance qui va trop loin au niveau d'une jonction lisse. A gauche : la scène et le cylindre construit ; à droite : les points considérés dans la croissance du cylindre.

On vient de voir que la méthode proposée est sensible au point initial. Cependant, si la séquence de primitives qui composent la ligne a été correctement trouvée, on peut envisager de raffiner *globalement* (et non plus localement) cette séquence sur les points. Ceci correspond à considérer une primitive composée, formée de plusieurs primitives simples. L'ajustement de ce type de primitive est un peu plus détaillé au chapitre 4, section 4.5.5. A priori, la principale difficulté pour ajuster des primitives composées réside dans les valeurs initiales. Dans cette optique, ce qui a été fait dans cette partie peut être perçu comme un moyen de fournir de telles valeurs initiales.

La sélection de modèle entre tore et cylindre décrite ici est un peu sommaire et ne correspond pas toujours à la réalité. En effet, il existe des tuyauteries qui sont effectivement constituées de cylindres sécants, et qui produisent par cet algorithme des tores qui vérifient le critère sur le ratio du grand rayon sur le petit rayon. C'est le cas de la ligne calorifugée de la figure 3.41. Une approche qui paraît plus prometteuse est de réaliser une sélection de modèle après ajustement des deux primitives «cylindre sécant de rayon fixe» et «tore après cylindre» (voir section 5.1), tout en gardant la contrainte $R < 20r$ pour éliminer les tores douteux.

Une autre critique que l'on peut faire à ce procédé est qu'il ne garantit pas de trouver les orientations des parties rectilignes. Or, ces orientations devraient pouvoir être déterminées, soit à partir de l'image de Gauss (qui est estimée ici), soit par une méthode d'extraction des cylindres principaux (par exemple par une méthode d'extraction globale du type de celles décrites au chapitre 6). Il paraît notamment possible d'exploiter l'image de Gauss de manière plus globale.

Enfin, a posteriori, l'algorithme de modélisation étant destiné à être utilisé après la phase d'extraction, il semble dommage de ne pas y avoir exploité toute l'information obtenue à l'issue de la phase d'extraction. Nous avons tenté de combler ce vide à travers la méthode que nous présentons maintenant, qui fait le lien entre extraction et modélisation.

3.3.2 Segmentation à partir du résultat de l'algorithme d'extraction

Lors du procédé d'extraction présenté au 3.2, l'objectif était d'obtenir un sous-nuage de points correspondant à la ligne de tuyauterie que l'on souhaite extraire. Cependant, comme on l'a vu, pour réaliser l'extraction, l'algorithme se fonde sur des modèles. Il en résulte qu'à l'issue de l'algorithme, un modèle grossier, consistant en une séquence de cylindres contraints, est également disponible. Comment tirer parti de cette information pour produire le modèle final ? Si l'on reste à l'échelle du cylindre, on peut dire que l'algorithme d'extraction produit une sur-segmentation de la ligne. Bien entendu, ceci ne veut pas dire que les jonctions entre les primitives finales se trouvent exactement aux limites des cylindres élémentaires. Toutefois, dans les cas rencontrés en pratique, il semble que l'on puisse souvent faire une segmentation satisfaisante en restant à l'échelle des cylindres. Ceci est en particulier dû au fait que la taille des cylindres a été fixée égale au rayon.

Notre approche a donc été de traiter la séquence des cylindres en procédant à des regroupements afin d'obtenir une description contenant un ensemble de segments correspondant à la «réalité visuelle». Le problème s'exprime donc comme une segmentation de ligne brisée en 3D. Plusieurs idées sont envisageables pour traiter ce problème. Notons que l'attention s'est plus portée sur des procédés de segmentation traitant la séquence d'un point de vue global plutôt que sur des procédés locaux, fondés par exemple sur la notion de propagation (voir remarque faite à ce sujet dans la section 3.3.1).

Une voie est de considérer les techniques de partitionnement de courbes, comme par exemple les approches mettant en œuvre la corde [RW95]. Cependant, dans notre cas, la courbe est déjà une ligne brisée, en général simple, et constituée de grandes parties rectilignes. Notons de plus que les parties coudées ne sont en général représentées que par deux voire trois cylindres, ce qui rend difficile la reconnaissance de cercles dans la ligne brisée. L'idée a donc été de se limiter à dégager des segments rectilignes, en déterminant les directions présentes le long de la ligne.

Pour ce faire, les vecteurs directeurs unitaires des axes des cylindres ont été examinés sur la sphère unité. Le problème se ramène ainsi à dégager des amas (*clusters*) parmi des points sur la sphère unité, comme l'illustre la figure 3.44. Un procédé de vote local, proche d'une transformée de Hough (mais qui évite la discrétisation de la sphère), permet de réaliser cette tâche. Notons que l'on pourrait également procéder via une discrétisation de la sphère de Gauss (telle que dans [BJ98]), et

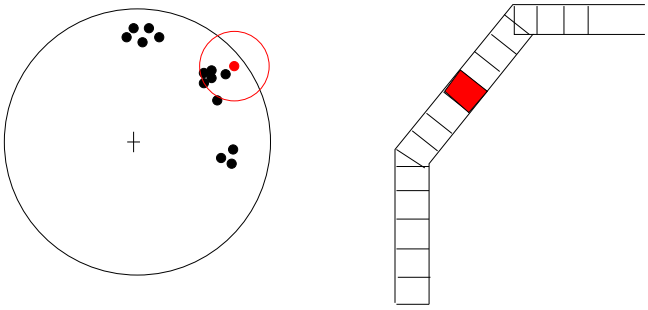


FIG. 3.44 – Sphère unité des vecteurs directeurs

ainsi obtenir un procédé plus performant en temps de calcul. Ceci étant, le nombre d'axe étant en général inférieur ou de l'ordre d'une centaine, ceci n'est pas utile. De plus, pour une taille de cellule donnée, le procédé utilisé ici rend un résultat plus précis car la position de la cellule est fixée par les points et non par une discrétisation fixe. Ce procédé, qui s'apparente également aux méthodes d'extraction globale décrites au chapitre 6 (à ceci près qu'il n'y a pas besoin ici de tirer des points aléatoirement puisque les points ne sont pas en grand nombre), est décrit dans le pseudo-code suivant :

Pseudo-code de la création des classes sur la sphère unité

Pour i=1 à N faire
  $L(i) \leftarrow$ Indices j des points $\mathbf{u_j}$ tels que $\|\mathbf{u_j} - \mathbf{u_i}\|^2 < \delta^2$
  $n_i \leftarrow$ nombre d'éléments dans $L(i)$
Fin Pour
$(i'_1, \ldots, i'_N) \leftarrow$ liste des indices $1, \ldots, N$ triée par ordre des n_i décroissants
$C_1 \leftarrow L(i'_1)$
Pour k=2 à N faire
  $C_k \leftarrow$ éléments de $L(i'_k)$ sauf ceux qui sont aussi dans $L(i'_1), \ldots, L(i'_{k-1})$
  $n_k \leftarrow$ nouveau nombre d'éléments dans C_k
Fin Pour
Retourner les classes C_j dont le nombre d'éléments n_j est au moins égal à 1.

Entrées : N points $\mathbf{u_i}$ sur la sphère unité.
Paramètres : rayon δ de la sphère de voisinage.
Sorties : classes $C_1, \ldots, C_r$ (r : nombre de classes, inconnu au départ, $r \leq N$).

On obtient ainsi r classes disjointes formant une partition de $\{1, \ldots, N\}$ (indices des cylindres). Notons que cet algorithme nécessite la donnée d'un paramètre, δ, rayon des voisinages (sphère) servant à définir chaque classe. Ce paramètre est à relier au niveau de bruit sur les vecteurs directeurs des cylindres. Si l'on spécifie une valeur trop faible, on obtient une sur-segmentation. Si l'on spécifie une valeur trop forte, on obtient une sous-segmentation. Notons qu'une sur-segmentation est moins problématique qu'une sous-segmentation pour les traitements ultérieurs. En pratique, δ a été fixé à 7 degrés. Un choix cohérent de cette valeur dépend bien entendu du niveau de bruit sur l'estimation des cylindres, qui lui dépend du bruit sur les points, de la densité, etc. Une perspective possible est de considérer non seulement les points sur la sphère unité, mais également leurs covariances respectives.

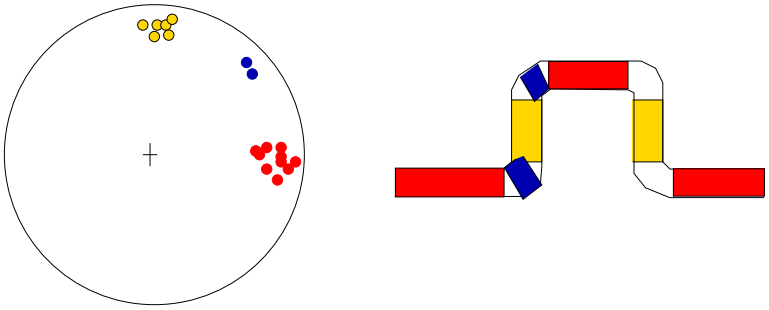


FIG. 3.45 – Classes obtenues sur la sphère unité

Il est à noter qu'une classe n'est pas forcément connexe sur la ligne de tuyauterie. Chaque classe contient en effet un certain nombre de composantes connexes sur la ligne de tuyauterie (Figure 3.45). Deux cas sont possibles entre deux composantes consécutives d'une classe :

- les deux composantes correspondent réellement à deux tuyaux parallèles mais distincts,
- les deux composantes correspondent à la même partie cylindrique, mais des composantes bruitées ont séparé ces composantes (dans le cas de cylindres bruités, e. g. au voisinage de poutres, ...).

Un second traitement vise :

- à séparer les composantes d'une même classe qui représentent en fait des parties cylindriques différentes, et
- à fusionner les composantes d'une même classe qui représentent en fait la même partie cylindrique.

Le premier cas peut se résoudre par une simple critère de distance des axes correspondant à deux composantes consécutives d'une même classe.

Le deuxième cas est actuellement traité par une lecture de toute la liste de composantes dans leur ordre d'importance (nombre de cylindres). Le principe est de fusionner les composantes d'une même classe si elles sont séparées par des composantes comportant un seul cylindre. Un deuxième paramètre intervient ici : c'est le nombre maximal de cylindres élémentaires que l'on autorise entre deux composantes à fusionner (fixé à quatre en pratique).

Enfin, on remplace la séquence de cylindres d'une composante par le cylindre formé par les extrémités du premier et du dernier cylindre de la composante.

On obtient ainsi une séquence de cylindres respectant les mêmes contraintes que les cylindres de départ («cylindre-rotule»).

3.3.3 Résultats et perspectives

Les résultats de cet algorithme dépendent bien entendu fortement des données d'entrée, à savoir la séquence de cylindres issue de l'algorithme d'extraction. Pour des données peu bruitées, les grandes parties rectilignes sont bien séparées des parties de changements de direction (Figures 3.46 et 3.47). Lorsque les données sont bruitées (notamment lorsque les erreurs sur l'angle dépassent le seuil δ), les résultats actuels sont satisfaisants au niveau des parties rectilignes (ou la fusion entre composantes fonctionne bien). Ils sont plus aléatoires au voisinage des changements de directions (coude, etc.), où la scène paraît encore sur-segmentée.

On peut envisager d'appliquer le même algorithme de manière répétée en vue d'améliorer les

FIG. 3.46 – Résultat du traitement de la séquence de cylindres issues de l'extraction (scène «pipe3»).

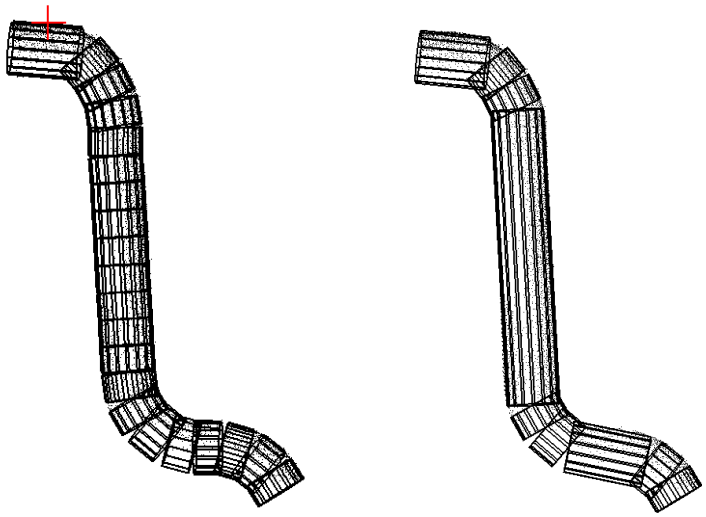


FIG. 3.47 – Résultat du traitement de la séquence de cylindres issue de l'extraction (scène «elf6pipe1»).

résultats. En effet, on a vu que les cylindres obtenus à la fin sont du même type que ceux de départ. Or, les angles entre eux ont diminué étant donné que le cylindre final est une sorte de moyenne des

cylindres de la composante correspondante.

Dans le cas général, ce qui est fait ici ne produit pas le modèle final (en tout cas pas si l'on se trouve en présence de coudes). Il reste effectivement à réaliser la modélisation proprement dite, en revenant aux points et à l'ajustement de surfaces.

3.3.4 Perspectives sur la modélisation

Comment produire le modèle final en évitant les instabilités ? Il semble qu'une solution réside en une combinaison des deux méthodes que nous avons présentées. La première étape consisterait à appliquer la méthode décrite au 3.3.2 afin de dégager des cylindres et les sous-nuages points correspondants. A l'issu de cette étape, on peut envisager une modélisation du type de celle présentée au 3.3.1 non plus en définissant des voisinages locaux, mais en utilisant directement les sous-nuages de points issus de l'étape de segmentation. En particulier, ceci mettrait toujours en œuvre l'ajustement de primitives contraintes utilisées dans 3.3.1.

Une autre perspective concernant l'algorithme de segmentation serait de l'intégrer à l'algorithme d'extraction. Le regroupement de cylindres pourrait alors se faire «en ligne» au fur et à mesure que l'on extrait la ligne de tuyauterie. L'intérêt serait de pouvoir utiliser la connaissance que l'on a du tuyau pour mieux extraire la suite.

En ce qui concerne la modélisation proprement dite, des primitives supplémentaires sont à inclure pour prendre en compte tous les cas décrits figure 3.6 : cylindre coaxial, réductions concentrique et excentrique, éventuellement primitive «bride» (voir chapitre 4). En introduisant plus de primitives, la notion de sélection de modèle, développée au chapitre 5 section 5.1, prend toute son importance.

3.4 Conclusion

Dans ce chapitre, des méthodes d'extraction et de modélisation de lignes de tuyauterie ont été présentées pour les tuyauteries de diamètre constant.

Nous avons vu notamment que les méthodes développées font intervenir des paramètres, même si nous avons tâché d'en restreindre le nombre. Il semble que ce soit le cas de tout algorithme de segmentation. Dans [HJBJ+96], qui présente une comparaison expérimentale d'algorithmes de segmentation d'images 2,5D, il y a un ensemble d'apprentissage (*training set*) et un ensemble de test (*test set*), ce qui permet de déduire des valeurs optimales pour les paramètres et les seuils. La difficulté ici est que les algorithmes développés requièrent (encore) une intervention de l'utilisateur. Ceci fait qu'il est délicat de définir des valeurs optimales pour les paramètres. Cette définition est plus aisée lorsqu'il s'agit d'un algorithme complètement automatique, car on peut réaliser de nombreux tests.

Chapitre 4

Ajustement de primitives

4.1 Introduction

On a vu aux chapitres 2 et 3 que la segmentation et la modélisation sont étroitement imbriquées lorsque l'on souhaite segmenter une scène à partir de modèles. On est donc amené à modéliser le nuage de points, ce qui se fait par des méthodes d'ajustement de surfaces. Ceci consiste à ajuster au mieux un modèle, ici une primitive géométrique, sur des données.

Peu de travaux de recherche existent sur ce sujet pour ce que nous appelons des «primitives géométriques», c'est-à-dire pour des modèles tels que le plan, la sphère, le cylindre, le cône et le tore. Les primitives contraintes sont encore plus rares dans la littérature. D'autre part, il existe un petit nombre de logiciels commerciaux qui permettent d'effectuer ces opérations d'ajustement de primitives : à notre connaissance 3Dipsos (Mensi), Cyclone (Cyra), Catia V5R8 (Dassault Systèmes).

4.1.1 Primitives traitées

Les primitives pour lesquelles nous avons réalisé des fonctions d'ajustement correspondent aux primitives introduites au chapitre 1. Cependant, pour la construction de modèle CAO, il est également intéressant de faire intervenir des contraintes, par exemple pour assurer des jonctions continues ou lisses (voir chapitre 3). Nous avons donc introduit des primitives libres, c'est-à-dire non-contraintes, ainsi que des primitives contraintes.

Primitives libres

Les primitives libres — c'est-à-dire non-contraintes— qui ont été introduites sont : le plan, la sphère, le cylindre de révolution, le cône de révolution, et le tore circulaire (Figure 4.1).

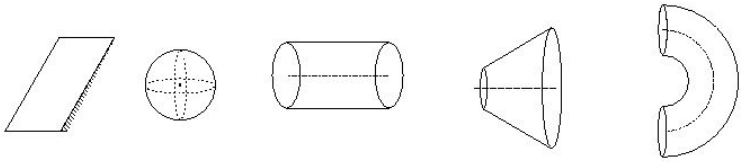


FIG. 4.1 – Primitives libres

Primitives contraintes

Les primitives contraintes introduites sont le «cylindre-rotule», le «cylindre sécant», le «cylindre sécant de rayon fixé», le «tore après cylindre», le «tore entre cylindres», et le «cylindre après tore» (Figure 4.2).

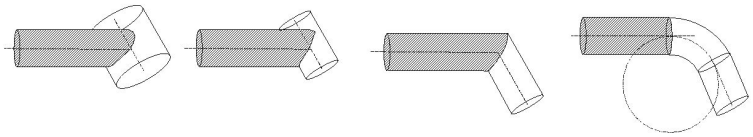


FIG. 4.2 – Primitives contraintes

4.1.2 Motivation

Dans 3Dipsos, de telles fonctions d'ajustement existent pour la plupart des primitives introduites au chapitre 1. Cependant, il existe certains défauts dans ces procédés d'ajustement. En effet, l'algorithme utilisé dans 3Dipsos ne parvient pas toujours à ajuster la primitive de manière acceptable. C'est le cas en particulier de certaines primitives non-contraintes (notons que les primitives non-contraintes interviennent relativement peu dans une utilisation standard de 3Dipsos). Les figures 4.3 et 4.4 montrent de tels exemples, où les méthodes que nous avons développées fournissent le bon résultat.

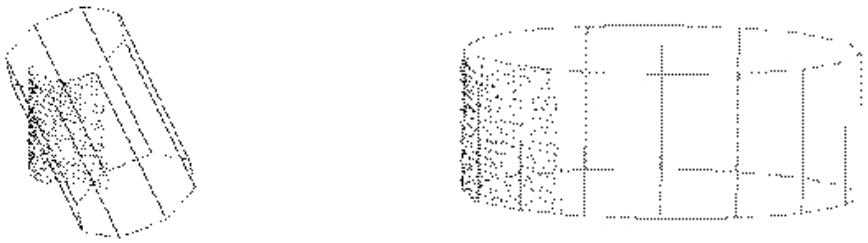


FIG. 4.3 – Ajustement de cylindre libre. Dans cet exemple, la fonction d'ajustement de 3Dipsos ne fournit pas une solution acceptable (à gauche). Pourtant les points se trouvent bien sur un cylindre, sans points aberrants. Sur le même exemple, la méthode développée et présentée dans ce chapitre donne un résultat satisfaisant (à droite).

Ceci était un problème, puisque nous avions besoin de fonctions d'ajustement de surfaces susceptibles d'être utilisées de manière automatique, c'est-à-dire qui fonctionnent également dans des conditions moins bonnes que la plupart du temps lorsqu'un utilisateur spécifie une région. En effet, dans 3Dipsos, l'utilisateur ajuste souvent des primitives non-contraintes sur des zones assez «claires». Si la fonction d'ajustement est appliquée à une zone dont on ne connaît rien, le comportement de la fonction d'ajustement devient plus critique. En particulier, ceci nécessite des fonctions d'ajustement qui soient stables vis à vis de la répartition des points, ainsi que vis à vis des situations où les points ne sont pas réellement sur une primitive du type de celle qui est ajustée. Or, ceci n'est pas le cas pour les fonctions dans 3Dipsos.

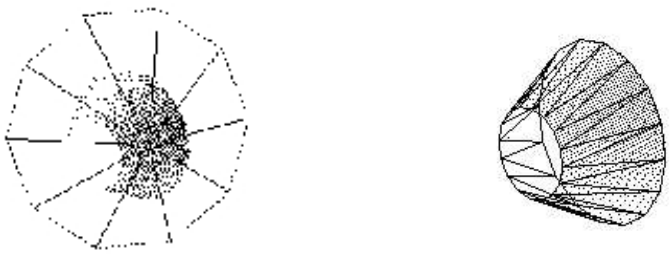


FIG. 4.4 – Ajustement de cône libre. Dans cet exemple, la fonction d'ajustement de 3Dipsos ne fournit pas une solution acceptable (à gauche). Sur le même exemple, la méthode développée et présentée dans ce chapitre donne un résultat satisfaisant (à droite).

Enfin, il n'y a actuellement pas de fonction disponible pour l'ajustement de tore sans contraintes dans 3Dipsos.

Il était donc nécessaire de définir de nouvelles fonctions d'ajustement, plus robustes vis à vis de la répartition des points et vis à vis des situations de mauvais modèle (en particulier dans un contexte de sélection de modèle, lorsque l'on essaie plusieurs types de modèles).

> Remarque
> Notons qu'il ne s'agit pas ici d'ajustement robuste par rapport aux points aberrants : on parle toujours ici d'ajustement de moindres carrés (mais il se trouve que le minimum de la fonction coût des moindres carrés n'est pas facile à trouver, et que l'on n'est pas sûr de le trouver). En particulier, les exemples ci avant montrent des cas où la fonction de 3Dipsos ne parvient pas à atteindre le minimum. En ce qui concerne l'ajustement robuste par rapport aux points aberrants, ceci est une autre question, abordée au 4.5.2.

4.1.3 Expression du problème et choix

Problématique générale de l'ajustement

De manière générale, l'ajustement de modèle consiste à chercher le modèle M, de type connu, défini par les paramètres $\mathbf{a}$ (de forme et de position), qui ajuste «au mieux» un ensemble de données $\{\mathbf{x_i}\}_{1\leq i\leq N}$. On définit «au mieux» par le minimum d'une fonction coût F. Le problème de l'ajustement revient donc à résoudre :

$$\min_{\mathbf{a}} F(\mathbf{x_1},\ldots,\mathbf{x_N};\mathbf{a})$$

Par conséquent, une méthode d'ajustement d'un modèle peut se caractériser par :
- le type de modèle et ses paramètres $\mathbf{a}$,
- une fonction coût F, faisant intervenir une métrique mesurant l'écart entre un point $\mathbf{x}$ et le modèle $M(\mathbf{a})$,
- la méthode d'estimation des paramètres $\mathbf{a}$.

Remarquons dès à présent que ce problème n'entre pas dans le cadre d'un problème de régression. En effet, en régression, on cherche à prédire une variable y en fonction de variables explicatives x sous la forme d'une fonction $y = f(x)$ de type connu. Or, on voit ici qu'il n'y pas de variables explicatives ou expliquées.

Approche proposée

Les choix qui ont été faits ici peuvent se synthétiser comme suit :
- des modèles spécifiques définis par la distance exacte,
- la fonction coût des moindres carrés,
- des modèles contraints : nouveaux modèles, avec nouveaux paramètres non-contraints,
- estimation des paramètres par algorithme d'optimisation non-linéaire (Levenberg-Marquardt). Initialisation produite à l'aide d'informations de géométrie différentielle (normales, image de Gauss, éventuellement courbures).

En clair, cela veut dire que l'on considère la fonction coût

$$F = \sum_{i=1}^{N} d(\mathbf{x_i}; \text{Primitive}(\mathbf{a}))^2$$

où la métrique d est la distance exacte du point à la primitive (propre à chaque primitive).

Pour résoudre ce problème, sauf dans le cas du plan, il est nécessaire de recourir à une méthode numérique pour trouver les paramètres $\mathbf{a}_{\text{min}}$ réalisant le minimum.

La méthode de résolution proprement dite, qui ne dépend pas des primitives, est décrite à la section 4.2. Ensuite, les parties 4.3 et 4.4 présentent les éléments propres à chaque primitive : les paramètres et les métriques introduits sont d'abord décrits à la section 4.3 et dans l'annexe A ; puis la section 4.4 traite de l'étape cruciale de l'initialisation des paramètres. Enfin, la section 4.5 regroupe les différents points de discussion et perspectives sur les méthodes développées.

Remarque
On peut envisager de résoudre un tel problème d'estimation non-linéaire par la résolution d'un système d'équations plutôt que par optimisation directe de la fonction coût. En effet, les paramètres **a** étant libres, au minimum, le gradient de la fonction coût par rapport aux paramètres est nul. On obtient ainsi un système d'équations, appelées équations normales. La résolution d'un tel système est numérique, et ne semble pas plus simple à effectuer que l'optimisation que nous effectuons.

4.2 Méthode de résolution du problème d'estimation

4.2.1 Algorithme d'optimisation utilisé

L'algorithme de Levenberg-Marquardt est une méthode d'optimisation couramment utilisée pour l'estimation de paramètres [PTVF92b]. Le principe de l'algorithme est d'abord présenté. La section décrit ensuite comment cet algorithme a été adapté et utilisé dans notre situation.

Principe

Cette méthode non-linéaire procède itérativement pour trouver le vecteur paramètre **a** ($\mathbf{a} \in \mathbb{R}^p$) minimisant la fonction coût F. Proche du minimum, on peut s'attendre à ce que cette fonction soit bien approchée par une forme quadratique du type :

$$F(\mathbf{a}) = \gamma + \mathbf{d} \cdot \mathbf{a} + \frac{1}{2}\mathbf{a} \cdot \mathbf{D} \cdot \mathbf{a} \tag{4.1}$$

où γ, **d** et **D** désignent respectivement un scalaire, un vecteur de $\mathbb{R}^p$ et une matrice de $\mathcal{M}_p(\mathbb{R})$ constants.

Si cette approximation locale est valide au voisinage du point courant $\mathbf{a_{cur}}$, on peut parvenir en une seule itération aux paramètres $\mathbf{a_{min}}$ minimisant la fonction F, par la relation :

$$\mathbf{a_{min}} = \mathbf{a_{cur}} - \mathbf{F''}(\mathbf{a_{cur}})^{-1} \cdot \mathbf{F'}(\mathbf{a_{cur}}) \tag{4.2}$$

où $\mathbf{F'}$ et $\mathbf{F''}$ désignent respectivement le gradient et la matrice hessienne de F par rapport aux paramètres $\mathbf{a}$. Rappelons que le gradient de f s'écrit

$$(\mathbf{f'})_k = \frac{\partial f}{\partial a_k}$$

et que la matrice hessienne de f s'écrit

$$(\mathbf{f''})_{kl} = \frac{\partial^2 f}{\partial a_k \partial a_l}$$

Preuve:
Par un développement de Taylor local sur un voisinage de $\mathbf{a_{cur}}$, la fonction coût s'écrit (si F est $\mathcal{C}2$ sur ce voisinage) :

$$F(\mathbf{a}) = F(\mathbf{a_{cur}}) + \mathbf{F'}(\mathbf{a_{cur}}) \cdot (\mathbf{a} - \mathbf{a_{cur}}) + \frac{1}{2}(\mathbf{a} - \mathbf{a_{cur}}) \cdot \mathbf{F''}(\mathbf{a_{cur}}) \cdot (\mathbf{a} - \mathbf{a_{cur}}) + o(\|\mathbf{a} - \mathbf{a_{cur}}\|^2)$$

En prenant le gradient de cette expression, on obtient :

$$\mathbf{F'}(\mathbf{a}) \approx \mathbf{F'}(\mathbf{a_{cur}}) + \mathbf{F''}(\mathbf{a_{cur}}) \cdot (\mathbf{a} - \mathbf{a_{cur}})$$

A l'optimum $\mathbf{F'}(\mathbf{a_{min}}) = \mathbf{0}$ donc

$$\mathbf{0} \approx \mathbf{F'}(\mathbf{a_{cur}}) + \mathbf{F''}(\mathbf{a_{cur}}) \cdot (\mathbf{a_{min}} - \mathbf{a_{cur}})$$

soit

$$\mathbf{a_{min}} - \mathbf{a_{cur}} \approx -\mathbf{F''}(\mathbf{a_{cur}})^{-1} \cdot \mathbf{F'}(\mathbf{a_{cur}})$$

ce qui conclut. □

En revanche, si l'approximation locale de la fonction par la forme quadratique (4.1) s'avère mauvaise au point courant $\mathbf{a_{cur}}$, on procède simplement à une étape de descente de gradient (plus grande pente), à savoir :

$$\mathbf{a_{next}} = \mathbf{a_{cur}} - \lambda \mathbf{F'}(\mathbf{a_{cur}}) \tag{4.3}$$

où λ est une constante suffisamment petite pour ne pas dépasser la région où il y a une descente. Nous verrons plus loin comment l'algorithme décide entre les deux possibilités. Le calcul du gradient et de la hessienne est également présenté plus loin.

Posons $\mathbf{H} \in \mathcal{M}_p(\mathbb{R})$ et $\mathbf{g} \in \mathbb{R}^p$ tels que $\forall k, l$:

$$g_k = -\frac{1}{2}\frac{\partial F}{\partial a_k} = -\frac{1}{2}(\mathbf{F'})_k \tag{4.4}$$

$$H_{kl} = \frac{1}{2}\frac{\partial^2 F}{\partial a_k \partial a_l} = \frac{1}{2}(\mathbf{F''})_{kl} \tag{4.5}$$

L'équation (4.2) se réécrit alors sous la forme

$$\mathbf{H} \cdot \delta\mathbf{a} = \mathbf{g} \tag{4.6}$$

Ce système linéaire se résout en $\delta\mathbf{a}$ qui incrémente l'actuelle approximation pour donner la suivante. D'autre part, l'équation (4.3) de descente de gradient devient :

$$\delta\mathbf{a} = \text{constante} \times \mathbf{g} \tag{4.7}$$

L'algorithme de Levenberg-Marquardt

Marquardt a suggéré une méthode élégante qui tient compte à la fois de la méthode de la descente de gradient et de la méthode d'inversion de la hessienne en passant de l'une à l'autre de manière continue.
Le principe de cette méthode repose sur deux observations.
D'après l'équation 4.4, g_k est homogène à $1/a_k$. Donc la constante de l'équation 4.6 doit être homogène à a_k^2 ou encore, ce qui est équivalent, à $1/H_{kk}$. Cette remarque permet de donner une échelle à la constante. Toutefois, il peut arriver que cette échelle soit trop grande. C'est pourquoi on utilise une variable supplémentaire λ qui permet de la moduler. L'équation 4.6 est remplacée par :

$$\delta a_l = \frac{1}{\lambda H_{ll}} g_l \quad \Leftrightarrow \quad \lambda \cdot H_{ll} \cdot \delta a_l = g_l \tag{4.8}$$

La seconde observation faite par Marquardt est que les équations 4.6 et 4.7 peuvent être combinées si l'on définit une nouvelle matrice $\bar{\mathbf{H}}$ comme suit

$$\bar{H}_{jj} = H_{jj}(1+\lambda) \tag{4.9}$$
$$\bar{H}_{jk} = H_{jk} \qquad (j \neq k) \tag{4.10}$$

Les deux équations 4.6 et 4.3 deviennent alors

$$\bar{\mathbf{H}} \delta \mathbf{a} = \mathbf{g} \tag{4.11}$$

Ainsi, lorsque λ est grand, la matrice devient diagonalement supérieure, et l'équation 4.11 devient identique à l'équation 4.3 , ce qui signifie que l'on retrouve la méthode de la descente de gradient. A l'inverse, si λ est petit, l'équation 4.11 tend vers l'équation 4.6, à savoir la méthode d'inversion de la hessienne.

L'algorithme proposé par Marquardt, tel qu'implémenté dans la fonction `mrqmin` de [PTVF92b], prend la forme du pseudo-code suivant :

Pseudo-code de l'algorithme d'optimisation

```
Départ
    a ← a0 (valeur initiale)
    λ ← 0.001
    Continuer ← vrai
Tant que (Continuer = vrai) faire
    Calculer F(a)
    Calculer g et H au point a
    Résoudre Hδa = g en δa
    Calculer F(a + δa)
    Si F(a + δa) ≥ F(a) faire
        λ ← 10λ
    Sinon faire
        λ ← 0.1λ
        a ← a + δa
    Fin Si
    Critère d'arrêt : Continuer ← faux ou Continuer ← vrai
Fin Tant que
Retourner a et H(a)
```

La fonction `xmrqmin` de [PTVF92a] (qui utilise la fonction `mrqmin` de [PTVF92b]), prévoit un simple critère d'arrêt sur la variation de F. Tout d'abord, l'algorithme utilise une fonction coût normalisée par le niveau du bruit sur les données :

$$F = \sum_{i=1}^{N} \frac{d_i^2}{\sigma_i^2}$$

où chaque σ_i correspond à l'écart-type du bruit sur le point i, spécifié par l'utilisateur. La quantité F est donc adimensionnelle. L'algorithme s'arrête lorsque le résidu diminue de manière stable et de moins en moins. Plus précisément, la fonction `xmrqmin` s'arrête lorsque F diminue pour 4 appels consécutifs de la fonction `mrqmin` et que la variation de F entre deux appels successifs est inférieure à un certain seuil (=0,1).

Adaptation des critères d'arrêt à notre problème

Dans notre application, nous avons vu au chapitre 1 qu'il semble difficile de spécifier un bruit sur les données a priori, c'est-à-dire avant de réaliser l'ajustement de surfaces. Or dans la version classique de l'algorithme de Levenberg-Marquardt qui vient d'être présenté, l'écart-type du bruit sur les données intervient dans la fonction coût (σ_i). Il a donc été nécessaire d'adapter légèrement l'algorithme à notre situation[1]. La seule différence se situe au niveau de la condition d'arrêt. Tout d'abord, la fonction coût utilisée n'est pas normalisée par un niveau de bruit :

$$F = \sum_{i=1}^{N} d_i^2$$

[1] Notons cependant que s'il devenait techniquement possible d'obtenir une estimation du niveau de bruit (issu de l'acquisition et du recalage) en chaque point du nuage, il serait évidemment judicieux de l'utiliser à ce niveau, en suivant le formalisme de [PTVF92b].

Ensuite, les critères d'arrêt utilisés dans notre implémentation sont les suivants :

– Critère sur la variation de F : l'algorithme s'arrête lorsque F diminue de manière très faible :

$$\frac{F_{(k)} - F_{(k+1)}}{F_{(k+1)}} < c$$

où $c = 10^{-3}$ en pratique ($F_{(k)}$ désigne la valeur de la fonction coût à l'itération k). On utilise la normalisation par $F_{(k+1)}$ pour obtenir un critère sur une donnée adimensionnelle.

– Critère sur la valeur de λ : l'algorithme s'arrête lorsque λ est devenu soit très petit ($\leq 10^{-20}$), soit très grand ($\geq 10^{10}$).

En pratique, on remarque que la convergence est en général atteinte rapidement (au bout de 3 ou 4 itérations).

Calcul du gradient et de la hessienne dans le cas des moindres carrés

Les méthodes précédentes requièrent le calcul du gradient $\mathbf{F}'$ et de la hessienne $\mathbf{F}''$ de la fonction F à minimiser. Rappelons que $\mathbf{F}'$ a pour composantes :

$$(\mathbf{F}')_k = \frac{\partial F}{\partial a_k}$$

et que $\mathbf{F}''$ a pour composantes :

$$(\mathbf{F}'')_{kl} = \frac{\partial^2 F}{\partial a_k \partial a_l}$$

Dans le cas de la méthode de moindres carrés, la fonction coût s'écrit :

$$F(\mathbf{a}) = \sum_{i=1}^{N} d_i(\mathbf{a})^2$$

où d_i est une certaine distance entre le point i et le modèle. Le gradient $\mathbf{F}'$ par rapport à $\mathbf{a}$, qui sera nul au minimum, a alors pour composantes :

$$(\mathbf{F}')_k = 2 \sum_{i=1}^{N} d_i \frac{\partial d_i(\mathbf{a})}{\partial a_k}$$

La matrice hessienne $\mathbf{F}''$ s'exprime sous la forme :

$$(\mathbf{F}'')_{kl} = 2 \sum_{i=1}^{N} \left[\frac{\partial d_i(\mathbf{a})}{\partial a_k} \frac{\partial d_i(\mathbf{a})}{\partial a_l} + d_i(\mathbf{a}) \frac{\partial^2 d_i(\mathbf{a})}{\partial a_k \partial a_l} \right]$$

On voit que le second terme de cette expression dépend des dérivées secondes de F. Au voisinage de la solution, on a :

$$d_i(\mathbf{a}) \approx 0$$

si l'on suppose que les points sont effectivement sur un modèle du type de celui ajusté et que l'on a atteint le minimum. De plus, dans certains cas, la moyenne des d_i est nulle[2]. En outre, il a été reporté

[2] Pour les surfaces libres du 4.1.1, on peut montrer en théorie que la moyenne des d_i est nulle à l'optimum. Ceci provient des paramètres de forme et des équations normales. En pratique, cette moyenne n'est pas tout à fait nulle, ne serait-ce que parce que l'optimum n'est jamais tout à fait atteint numériquement. Pour les primitives contraintes pour lesquelles les contraintes affectent les paramètres de forme, cette propriété n'est plus vérifiée.

que le terme faisant intervenir la dérivée seconde peut avoir pour effet de créer de l'instabilité dans l'algorithme [PTVF92b]. En conséquence, dans la plupart des travaux, le terme retenu est généralement :

$$(\mathbf{F}'')_{kl} \approx 2\sum_{i=1}^{N} \frac{\partial d_i(\mathbf{a})}{\partial a_k}\frac{\partial d_i(\mathbf{a})}{\partial a_l}$$

où seules les dérivées premières apparaissent. C'est également ce que nous faisons ici[3].

Le terme d_i est une mesure de l'écart entre un point $\mathbf{x_i}$ et la primitive P (dépendant de paramètres) :

$$d_i = d(\mathbf{x_i}; P)$$

La métrique d utilisée pour chaque primitive est présentée section 4.3.2. Les gradients de ces métriques par rapport aux paramètres sont présentés en annexe A. Comme on peut le voir section 4.3, pour les primitives utilisées, la distance d utilisée et son gradient ont une formulation analytique. Il était donc intéressant de tirer parti de cette propriété en utilisant les expressions analytiques dans l'algorithme d'optimisation.

4.2.2 Incertitude sur les paramètres estimés

Lorsque l'on estime un modèle à partir de données, il est important de pouvoir quantifier l'incertitude sur l'estimation fournie. Dans notre cas, il s'agit de fournir des intervalles de confiance, ou tout au moins une variance sur les paramètres estimés.

Or, l'algorithme de Levenberg-Marquardt fournit une telle information. En effet, la fonction `mrqmin` rend la matrice $\mathbf{H}^{-1}(\mathbf{a}_{\min})$ (avec λ =0), qui permet d'estimer la matrice de covariance sur le vecteur de paramètres $\mathbf{a}$:

$$\mathrm{Cov}(\mathbf{a}) = \sigma^2\mathbf{H}^{-1}(\mathbf{a}_{\min}) \tag{4.12}$$

Preuve:
Au voisinage du minimum, on a :

$$\mathbf{a} = f(\mathbf{x_1}, \ldots, \mathbf{x_N})$$

où f est une fonction inconnue de $\mathbb{R}^{3N}$ dans $\mathbb{R}^p$. Localement, les variations tangentielles de chaque point $\mathbf{x_i}$ n'ont pas d'influence directe sur la valeur de $\mathbf{a}$. Ainsi seules les variations normales d_i ont une influence. De plus, en utilisant une approximation linéaire de $\mathbf{a}$ au voisinage de $\mathbf{a}_{\min}$, on peut écrire le lien entre des petites variations δa_j et des petites variation δd_i comme suit :

$$\begin{pmatrix} \delta a_1 \\ \vdots \\ \delta a_p \end{pmatrix} = M \begin{pmatrix} \delta d_1 \\ \vdots \\ \delta d_N \end{pmatrix}$$

où la matrice M est définie par :

$$M_{i,j} = \frac{\partial a_i}{\partial d_j}$$

[3] Notons que la présence de la hessienne dans l'algorithme est surtout un moyen d'accélérer la convergence. Aussi une grande précision n'est-elle pas indispensable.

De manière plus concise, on peut écrire $\delta\mathbf{a} = M\delta\mathbf{d}$. On estime la covariance de $\mathbf{a}$ au voisinage de $\mathbf{a}_{\min}$ par l'espérance $E(\delta\mathbf{a}\delta\mathbf{a}^T)$. Par conséquent :

$$\begin{aligned} \mathrm{Cov}(\mathbf{a}) &= E(\delta\mathbf{a}\delta\mathbf{a}^T) \\ &= E(M\delta\mathbf{d}\delta\mathbf{d}^T M^T) \\ &= ME(\delta\mathbf{d}\delta\mathbf{d}^T)M^T \\ &= MCov(\mathbf{d})M^T \end{aligned}$$

On suppose ici que $Cov(\mathbf{d}) = \sigma^2 I_N$, ce qui donne $Cov(\mathbf{a}) = \sigma^2 MM^T$. Or,

$$MM^T = \Big(\sum_{k=1}^{p} \frac{\partial^2 a_k}{\partial d_i \partial d_j}\Big)_{i,j}$$

Or, on a $MM^T = H^{-1}$. □

Comme estimation de l'incertitude sur le paramètre a_i, on peut par conséquent utiliser le ième terme diagonal de la matrice $\mathbf{F}''^{-1}$ à l'optimum. Plus précisément, l'écart-type sur la ième composante du vecteur de paramètres $\mathbf{a}$ est estimé par :

$$\sigma(a_i) = \sigma\sqrt{(H^{-1})_{i,i}}$$

où σ est l'écart-type des résidus d_i par rapport à la surface et H est la hessienne de la fonction coût, ces deux quantités étant estimées à l'optimum $\mathbf{a}_{\min}$.

Ensuite, on peut si besoin examiner également les termes non diagonaux de la matrice de covariance. Ceci permet de définir une région de confiance autour de $\mathbf{a}_{\min}$ dans l'espace de paramètres du type :

$$\frac{1}{2}(\mathbf{a} - \mathbf{a_{min}}) \cdot \mathbf{H}(\mathbf{a_{min}}) \cdot (\mathbf{a} - \mathbf{a_{min}}) \leq k_\alpha \sigma^2$$

4.3 Paramétrisation et distance pour chaque primitive

On définit ici chaque primitive, avec les paramètres choisis pour la représenter.

Remarquons dès à présent que les extrémités des primitives ne sont pas prises en compte dans les paramètres. Par exemple, la hauteur du cylindre et le lieu de ses extrémités ne sont pas considérés, car ils ne conditionnent pas son rayon. En effet, comme nous allons le voir, les extrémités n'interviennent pas dans l'expression de la distance.

Remarque
Nous avons proposé ici des paramètres qui semblaient avoir le plus de sens du point de vue géométrique pour définir les primitives. Ces paramètres ne sont en général pas les seuls possibles. En particulier, on trouve d'autres paramétrisations possibles pour certaines des primitives qui suivent dans [LMM98, Tau91].

4.3.1 Paramètres pour chaque primitive

Primitives non-contraintes

Plan Le plan de moindres carrés se trouve de manière directe par un algorithme d'algèbre linéaire (voir annexe C). Il n'est donc pas utile ici de réaliser l'ajustement par une méthode non-linéaire.

Sphère

Définition On définit simplement une sphère par son centre **c** et son rayon r (Figure 4.5).

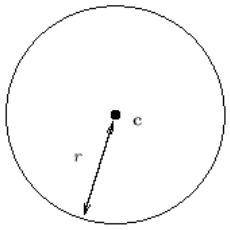


FIG. 4.5 – Paramètres de la sphère

Paramètres libres

Cylindre

Définition Un cylindre de révolution se définit simplement par un axe et un rayon.
La façon la plus intuitive de définir l'axe est de donner un point **p** et un vecteur unitaire **u** pour la direction (Figure 4.6).

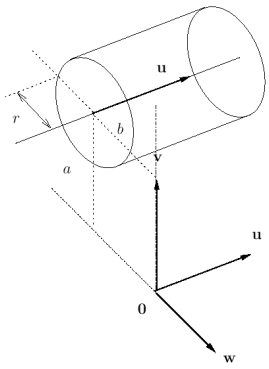


FIG. 4.6 – Paramètres du cylindre de révolution

Cependant, ces paramètres ne sont pas en nombre minimum : il est clair que le point peut se déplacer sur la droite sans pour autant modifier cette droite. On peut donc contraindre, par exemple, le point **p** à être le point de l'axe le plus proche de l'origine (c'est-à-dire la projection orthogonale du point **0** sur l'axe). Le point **p** peut alors se représenter par deux coordonnées a et b dans un repère orthonormé du plan $\mathbf{0}+\mathbf{u}^{\perp}$ (plan passant par **0** et perpendiculaire à **u**). Le repère orthonormé utilisé ici est construit à partir du trièdre $(\mathbf{u},\mathbf{v},\mathbf{w})$ défini à partir de la représentation en coordonnées sphériques (θ,ϕ) de **u** (les expressions des vecteurs $\mathbf{u},\mathbf{v},\mathbf{w}$ en fonction de (θ,ϕ) sont écrites à l'équation 1). Ceci conduit à la définition suivante :

Définition Vecteur directeur **u**, projection de O sur l'axe : $\mathbf{c} = a\mathbf{v} + b\mathbf{w}$, rayon r.

Paramètres libres $\mathbf{a} = (\theta, \phi, a, b, r)^T$ (dimension : 5)

Cône

Définition On définit le cône par un sommet **c**, un vecteur unitaire **u** et un demi-angle au sommet α ($0 \leq \alpha \leq \pi/2$) (Figure 4.7).

Paramètres libres $\mathbf{a} = (\theta, \phi, c_x, c_y, c_z, \alpha)^T$ (dimension : 6)

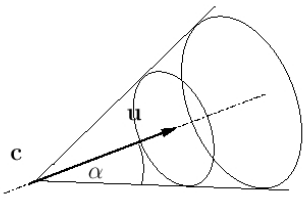


FIG. 4.7 – Paramètres du cône

Tore Un tore circulaire se définit par un cercle directeur et un rayon (Figure 4.8).

On définit le cercle directeur par son centre **c**, son rayon R (grand rayon du tore), dans un plan $\mathbf{c} + \mathbf{u}^\perp$ (plan orthogonal à une direction **u** passant par **c**).

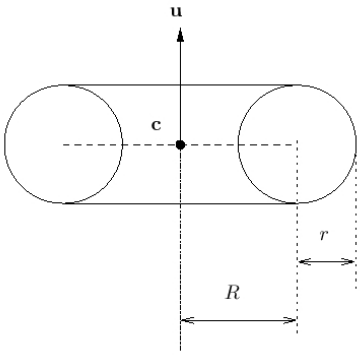


FIG. 4.8 – Paramètres du tore

Définition Vecteur normal **u**, centre **c**, grand rayon R, petit rayon r.

Paramètres libres $\mathbf{a} = (\theta, \phi, c_x, c_y, c_z, R, r)^T$ (dimension : 7)

Modèles contraints

Choix des paramètres L'idée que nous avons suivie a été d'exprimer les contraintes en réduisant le nombre de variables libres plutôt qu'en réalisant un ajustement sous contraintes (multiplicateurs de Lagrange), ce qui conduirait à des problèmes plus complexes (cf. commentaire [LMM98]).

Tous les ajustements ont donc été traités sans imposer de contraintes, même lorsque les primitives sont dites «contraintes», c'est-à-dire lorsqu'elles vérifient des contraintes géométriques imposées par des informations «métier» (jonction lisse entre un tore et un cylindre, cylindres sécants, ...).

Il existe bien entendu des cas où l'on peut exprimer les contraintes en fixant certains paramètres du modèle libre. Ce serait le cas par exemple d'un cylindre de rayon fixé.

Cependant, toutes les contraintes ne s'expriment pas aussi simplement. Par exemple, dans le cas du cylindre sécant, le point n'est pas fixé, mais contraint à se déplacer sur une droite fixe . Dans ce cas-là, si l'on souhaite travailler avec des paramètres libres, on est amené à considérer d'autres paramètres que ceux de la primitive non-contrainte correspondante.

«Cylindre sécant»

Définition Un «cylindre sécant» est un cylindre de révolution dont l'axe est concourant avec l'axe d'un cylindre de référence nommé «cylindre précédent» (Figure 4.9).

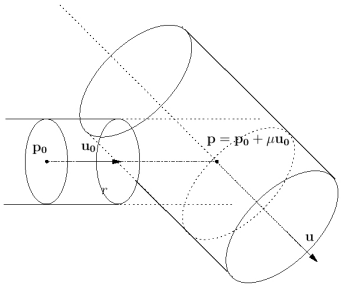


FIG. 4.9 – Paramètres du «Cylindre sécant»

La contrainte consiste donc ici à imposer qu'un point de l'axe du cylindre appartienne aussi à l'axe du cylindre précédent. Si l'axe du cylindre précédent est défini par un point $\mathbf{p_0}$ et un vecteur directeur $\mathbf{u_0}$, le point d'intersection des deux axes peut s'écrire :

$$\mathbf{p} = \mathbf{p_0} + \mu\mathbf{u_0}$$

avec $\mu \in \mathbb{R}$. La position de l'axe du cylindre peut donc être représentée par ce scalaire μ. Si on a $\mathbf{u} = \mathbf{u_0}$, on prendra $\mu = 0$, c'est-à-dire $\mathbf{p} = \mathbf{p_0}$ par convention.

Enfin, un vecteur directeur $\mathbf{u}$ et un rayon r sont nécessaires pour spécifier complètement le cylindre.

Paramètres libres $\mathbf{a} = (\mu, \theta, \phi, r)^T$ (dimension : 4)

«Cylindre sécant de rayon fixé»

Définition Un «cylindre sécant de rayon fixé» est un «cylindre sécant» dont le rayon est fixé égal au rayon du cylindre précédent (Figure 4.10).

Paramètres libres $\mathbf{a} = (\mu, \theta, \phi)^T$ (dimension : 3)

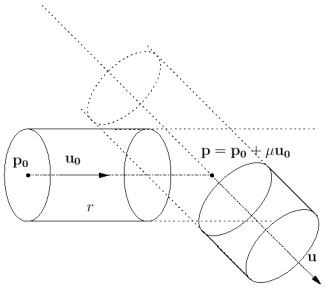


FIG. 4.10 – Paramètres du «Cylindre sécant de rayon fixé»

«Cylindre-rotule»

Définition Etant donné un premier cylindre dont on connaît les extrémités, un «cylindre-rotule» est un cylindre sécant dont l'axe est contraint à passer par l'extrémité du premier cylindre (liaison rotule) et dont le rayon est fixé à celui du premier cylindre (Figure 4.11). Seule la direction de l'axe est autorisée à varier.

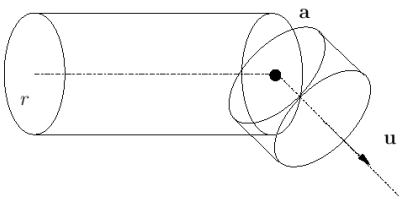


FIG. 4.11 – Paramètres du «cylindre-rotule»

Paramètres libres $\mathbf{a} = (\theta, \phi)^T$ (dimension : 2)

«Tore après cylindre»

Définition Un «Tore après cylindre» est un tore circulaire joignant un cylindre de même (petit) rayon (Figure 4.12). Le cylindre est défini par un axe Δ_0, lui-même défini par un point $\mathbf{p}_0$ et un vecteur directeur unitaire $\mathbf{u}_0$ (dirigé vers le tore), et un rayon r. R dénote le grand rayon du tore (on suppose que $R > r$), $\mathbf{n}$ le vecteur normal du tore et $\mathbf{x}$ un point courant.

Ici les coordonnées du centre du tore $\mathbf{c}$ sont prises comme seuls paramètres indépendants. En effet, les autres paramètres s'en déduisent :

$$R = \|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|$$

et

$$\mathbf{n} = \frac{(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}}{\|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|}$$

La convention d'orientation de $\mathbf{n}$ est telle que : $((\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}).\mathbf{n} > 0$, ce qui définit le sens de propagation comme le sens direct dans le plan $\mathbf{c} + \mathbf{n}^\perp$.

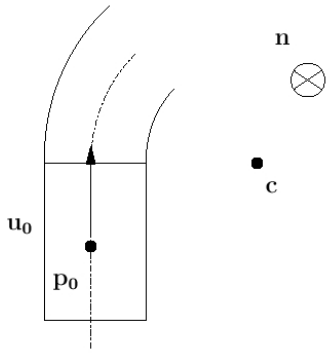


FIG. 4.12 – Paramètres du «Tore après cylindre»

Paramètres libres $\mathbf{a} = (c_x, c_y, c_z)^T$ (dimension : 3)

«Tore entre cylindres» Le but de cette primitive est d'améliorer la qualité du tore dans une séquence «cylindre-tore-cylindre» déjà existante, en laissant fixes les deux cylindres (l'ajustement de cylindre est moins délicat que l'ajustement de tore). Comme nous le verrons, dans le cadre de ces contraintes, le seul paramètre libre est le grand rayon du tore.

Définition Un «Tore entre cylindres» est un tore joignant un premier cylindre $(\mathbf{u_0}, \mathbf{p_0}, r)$, et un «cylindre sécant de rayon fixé» avec le premier cylindre $(\mathbf{u_1}, \mathbf{p_1}, r)$ (Figure 4.13). On suppose que $\mathbf{p_1}$ est le point de concours des deux axes et que $\mathbf{u_0}$ pointe vers $\mathbf{p_1}$ et $\mathbf{u_1}$ pointe dans le sens opposé à $\mathbf{p_1}$.

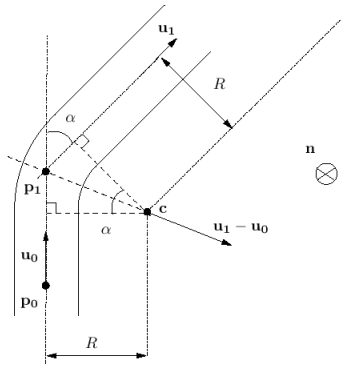


FIG. 4.13 – Paramètres du «Tore entre cylindres»

Remarque
Ici seul le cas d'axes sécants est traité, ce qui n'inclut pas le cas du demi-tour.

La normale au plan du cercle directeur est complètement déterminée par les axes des deux cylindres :

$$\mathbf{n} = \frac{\mathbf{u_0} \times \mathbf{u_1}}{\|\mathbf{u_0} \times \mathbf{u_1}\|}$$

La convention d'orientation est telle que la propagation s'effectue dans le sens direct dans le plan $\mathbf{p_0} + \mathbf{n}^{\perp}$.

Exprimons $\mathbf{c}$ et $\mathbf{n}$ en fonction de R et des paramètres des deux cylindres. Si $\mathbf{p_1}$ est le point d'intersection entre les 2 cylindres sécants et $\alpha = \arccos(\mathbf{u_0}.\mathbf{u_1})$ l'angle entre leurs axes ($0 \leq \alpha < \pi$), on a :

$$\mathbf{c} = \mathbf{p_1} + \frac{R}{\cos(\alpha/2)} \frac{\mathbf{u_1} - \mathbf{u_0}}{\|\mathbf{u_1} - \mathbf{u_0}\|}$$

Paramètre libre $\mathbf{a} = (R)$ (dimension : 1)

4.3.2 Expression de la distance exacte

Les expressions des distances d'un point $\mathbf{x}$ à une primitive pour les différentes primitives introduites sont présentées dans le tableau 4.1. Le détail de construction de chaque distance est donné en annexe A.

Les distances des primitives contraintes se déduisent directement des distances des primitives libres (même si les paramètres sont différents).

TAB. 4.1 – Tableau récapitulatif des distances

Primitive P	Paramètres libres $\mathbf{a} \in \mathbb{R}^p$	dim p	Distance exacte(signée) $d(\mathbf{x}, P)$
Plan	$\mathbf{n}, p$	3	$(\mathbf{x}.\mathbf{n}) - p$
Droite 3D	a, b, θ, ϕ	4	$\sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2}$
Sphère	$\mathbf{c}, r$	4	$\|\mathbf{x} - \mathbf{c}\| - r$
Cylindre	a, b, θ, ϕ, r	5	$\sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2} - r$
Cône	$\theta, \phi, \mathbf{c}, \alpha$	6	$\cos\alpha \|(\mathbf{x} - \mathbf{c}) \times \mathbf{u}\| - \sin\alpha((\mathbf{x} - \mathbf{c}).\mathbf{u})$ (zone1) ; $\|\mathbf{x} - \mathbf{c}\|$(zone2)
Cercle 3D	$\mathbf{c}, \theta, \phi, R$	6	$\sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x} - \mathbf{c}) \times \mathbf{u}\|}$
Tore	$\theta, \phi, \mathbf{c}, R, r$	7	$\sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x} - \mathbf{c}) \times \mathbf{u}\|} - r$
Cylindre sécant	μ, θ, ϕ, r	4	$\|(\mathbf{x} - \mathbf{p_0} - \mu\mathbf{u_0}) \times \mathbf{u}\| - r$
Cylindre sécant de rayon fixé	μ, θ, ϕ	3	$\|(\mathbf{x} - \mathbf{p_0} - \mu\mathbf{u_0}) \times \mathbf{u}\| - r$
Cylindre-rotule	θ, ϕ	2	$\|(\mathbf{x} - \mathbf{a}) \times \mathbf{u}\| - r$
Cylindre après tore	α	1	$\|(\mathbf{x} - \mathbf{a}) \times [\frac{\mathbf{a} - \mathbf{c}}{R} \times \mathbf{n}]\| - r$
Tore après cylindre	$\mathbf{c}$	3	$\left(\|\mathbf{x} - \mathbf{c}\|^2 + \|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|^2 - 2\|(\mathbf{x} - \mathbf{c}) \times ((\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0})\|\right)^{1/2} - r$
Tore entre cylindres	R	1	$\sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x} - \mathbf{c}) \times \mathbf{n}\|} - r$

Notons que pour les surfaces introduites la distance exacte définie ici est signée : elle est négative ou positive suivant l'endroit où se trouve le point par rapport à la surface. En effet, toutes les surfaces que nous considérons sont orientables. Ceci implique naturellement des limites pour certains paramètres : par exemple, le rayon du cylindre ne doit pas être nul, et le grand rayon du tore doit être au moins égal au petit rayon (qui ne doit pas être nul non plus).

Ceci ne change rien lorsque l'on considère le carré de la distance. Par contre, le fait que la distance soit signée peut être utile (voir chapitre 5). La distance ne peut par contre pas être signée pour les

courbes.

4.3.3 Gradient des distances aux primitives

Equations pour chaque primitive

Pour chaque primitive, les dérivées premières de la métrique par rapport aux paramètres ont été calculés analytiquement. Les équations obtenues sont présentées dans l'annexe A.

Concernant les problèmes de singularités dus à la distance exacte

Pour les primitives introduites ici, on peut en général écrire la distance exacte sous la forme :

$$d(\mathbf{x},P) = \sqrt{g} + h \tag{4.13}$$

où g et h sont deux fonctions à valeurs dans $\mathbb{R}$, dépendant de $\mathbf{x}$ et des paramètres de P, de classe $\mathcal{C}1$ sur l'espace de paramètres $\mathbb{R}^p$. A titre d'exemple, dans le cas du cylindre, la distance se met sous cette forme avec $g = (\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2$ et $h = -r$. Formellement, le gradient d'une telle fonction s'écrit alors :

$$\mathbf{d}' = \frac{\mathbf{g}'}{2\sqrt{g}} + \mathbf{h}' \tag{4.14}$$

Etant donné que l'expression du gradient comporte un dénominateur (dû à la racine carrée), il convient de connaître son comportement lorsque ce dénominateur s'annule. En effet, considérons la fonction coût F correspondant à la méthode des moindres carrés :

$$F = \sum_i^N d_i^2$$

Le terme d_i^2 s'écrit $(\sqrt{g_i} + h_i)^2 = g_i + h_i^2 + 2\sqrt{g_i}h_i$, qui fait encore intervenir une racine carrée. Formellement, le gradient de la fonction coût s'écrit donc :

$$\mathbf{F}' = \sum_{i=1}^{N} [\mathbf{g_i'} + 2h_i\mathbf{h_i'} + 2\sqrt{g_i}\mathbf{h_i'} + h_i\frac{\mathbf{g_i'}}{\sqrt{g_i}}]$$

On voit que le dernier membre fait toujours intervenir le dénominateur issu de la racine carrée.

En réalité, on peut spécifier un peu plus précisément la forme de la distance exacte, du moins pour les primitives introduites section 4.1.1. En effet, dans tous les cas sauf celui du cône, la distance peut se mettre sous la forme :

$$d(\mathbf{x},P) = \sqrt{g_1^2 + g_2^2 + g_3^2} + h$$

où g_1, g_2, g_3 et h sont des fonctions $\mathcal{C}1$ sur l'espace de paramètres $\mathbb{R}^p$.

> Remarque:
> notons que ceci est plus restrictif que la forme de l'équation 4.13. En effet, si le paramètre est $t \in \mathbb{R}$ et $g(t) = t$, alors g est $\mathcal{C}1$, mais ne peut pas s'écrire sous la forme d'une somme des carrés $g_1^2 + g_2^2 + g_3^2$ de fonctions $\mathcal{C}1$, puisque $\sqrt{t}$ n'est pas dérivable au point $t = 0$.

Or, on montre la propriété suivante :

Propriété 1 *Soit $d(\mathbf{x},P) = \sqrt{g_1^2 + g_2^2 + g_3^2} + h$ où g_1, g_2, g_3 et h sont des fonctions $\mathcal{C}1$ sur $\mathbb{R}^p$. S'il existe un point $\mathbf{a_0}$ de l'espace de paramètres où le terme $g_1^2 + g_2^2 + g_3^2$ s'annule, alors le gradient de la distance d est en général indéterminé, mais borné.*

Preuve:
Formellement, on peut écrire

$$\mathbf{d}' = \frac{g_1\mathbf{g}'_\mathbf{1} + g_2\mathbf{g}'_\mathbf{2} + g_3\mathbf{g}'_\mathbf{3}}{\sqrt{g_1^2+g_2^2+g_3^2}} + \mathbf{h}'$$

Au voisinage du point $\mathbf{a_0}$, on peut poser $g_1 = \varepsilon_1$, $g_2 = \varepsilon_2$, $g_3 = \varepsilon_3$, où $\varepsilon_1, \varepsilon_2, \varepsilon_3$ sont des infiniment petits. Par hypothèse de continuité, les gradients $\mathbf{g}'_\mathbf{i}$ prennent une valeur bornée au point $\mathbf{a_0}$. Au voisinage de $\mathbf{a_0}$, on a donc :

$$\mathbf{d}' = \frac{\varepsilon_1\mathbf{g}'_\mathbf{1}(\mathbf{a_0}) + \varepsilon_2\mathbf{g}'_\mathbf{2}(\mathbf{a_0}) + \varepsilon_3\mathbf{g}'_\mathbf{3}(\mathbf{a_0})}{\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}} + \mathbf{h}'$$

D'où, en appliquant l'inégalité triangulaire et en posant $\gamma = \max(\|\mathbf{g}'_\mathbf{1}(\mathbf{a_0})\|, \|\mathbf{g}'_\mathbf{2}(\mathbf{a_0})\|, \|\mathbf{g}'_\mathbf{3}(\mathbf{a_0})\|)$, :

$$\|\mathbf{d}'\| \leq \gamma\frac{|\varepsilon_1| + |\varepsilon_2| + |\varepsilon_3|}{\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}} + \|\mathbf{h}'\|$$

où $\|.\|$ désigne ici la norme euclidienne de $\mathbb{R}^p$. Or, par propriété d'équivalence des normes N_1 ($|x|+|y|+|z|$) et N_2 ($\sqrt{x^2+y^2+z^2}$) sur l'espace de dimension finie $\mathbb{R}^3$, il existe un $k > 0$ tel que :

$$|\varepsilon_1| + |\varepsilon_2| + |\varepsilon_3| \leq k\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}$$

Ce qui montre que le gradient $\mathbf{d}'$ est borné.

Il est par contre indéterminé, puisque les termes ε_i peuvent varier indépendamment. Pour s'en convaincre, il suffit de choisir $\varepsilon_2 = \mu\varepsilon_1$ et $\varepsilon_3 = \nu\varepsilon_1$ où μ et ν sont des paramètres réels. On a alors (si $\varepsilon_1 > 0$, ce qui n'est pas restrictif) :

$$\mathbf{d}' = \frac{1}{\sqrt{1+\mu^2+\nu^2}}[\mathbf{g}'_\mathbf{1}(\mathbf{a_0}) + \mu\mathbf{g}'_\mathbf{2}(\mathbf{a_0}) + \nu\mathbf{g}'_\mathbf{3}(\mathbf{a_0})] + \mathbf{h}'$$

Ceci correspond au fait que les différentes dérivées directionnelles de d sont toutes définies mais ne sont en général pas égales en un tel point de l'espace des paramètres. En conséquence, le gradient n'est pas prolongeable par continuité au voisinage d'un tel point dans l'espace de paramètres. □

Remarque
On pourrait en fait montrer le même résultat pour toute fonction $d = \sqrt{g} + h$, où $g = \sum_i^r g_i^2$ est une somme finie de carrés de fonctions $\mathcal{C}1$ sur l'espace de paramètres. Ceci n'est pas utile pour nos primitives, aussi s'est-on limité à montrer cette propriété pour une somme de 3 carrés seulement.

Afin de visualiser cette propriété, prenons l'exemple d'un cylindre où seuls les paramètres (a,b) sont autorisés à varier (les autres étant fixés aux vraies valeurs). De manière équivalente, on peut considérer le cas d'un cercle en 2D, de rayon r fixé, et dont seul le centre (a,b) peut se déplacer. La distance exacte (signée) du point (x,y) au cercle de centre (a,b) et de rayon r fixé s'écrit :

$$d = \|(x,y) - (a,b)\| - r$$

Le graphe de $d(x,y,a,b)$ en fonction de (a,b) est dessiné figure 4.14. Le profil est un cône circulaire dont le sommet se trouve en (x,y). Comme on peut le voir sur la figure 4.14, le gradient de $d(x,y,a,b)$ par rapport à (a,b) est simplement la projection de la ligne de plus grande pente le long du cône au point $(a,b,d(x,y,a,b))^T$. On perçoit alors clairement la singularité au niveau du sommet du cône : au voisinage de ce point, les gradients sont distincts suivant le côté du cône. Les projections sur le plan sont donc également distinctes.

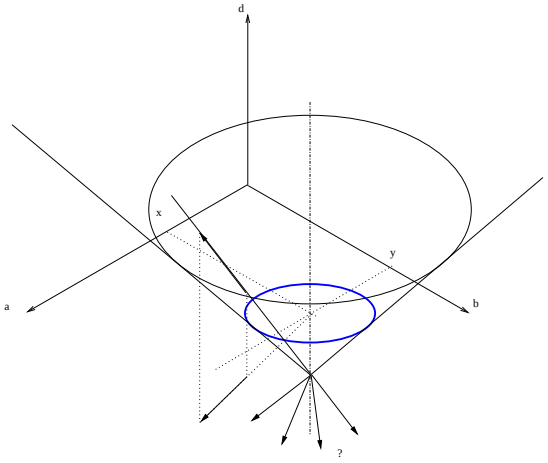


FIG. 4.14 – Graphe de $d(x,y,a,b)$ en fonction de (a,b) et visualisation de la singularité en $(a,b) = (x,y)$.

Il reste maintenant à montrer que la distance exacte pour les primitives introduites 4.1.1 peut bien s'écrire, dans chaque cas, sous la forme ci-dessus. C'est ce qui est reporté dans le tableau 4.2, où les termes g_1, g_2, g_3, h ont été explicités pour chaque primitive. Les primitives contraintes étant des cas particuliers des primitives libres, les lieux de singularités correspondent aux mêmes lieux de $\mathbb{R}^3$. Ils sont différents simplement en ce qu'ils se situent dans des espaces de paramètres différents.

> Remarque:
> Pour le cas du point, de la droite, du plan et du cercle en 3D, les singularités correspondent au lieu de l'espace des paramètres tel que le point **x** se trouve exactement sur la primitive elle-même (ensemble de niveau zéro). Dans les autres cas, les singularités sont éloignées des primitives. Il est d'ailleurs intéressant de noter que sphère et point, cylindre et droite, tore et cercle sont respectivement des ensembles de niveaux (*level-set*) des mêmes hypersurfaces.

En conclusion, pour les primitives introduites, le gradient de la distance exacte par rapport aux paramètres n'est pas continu sur tout l'espace de paramètres. En effet, il existe des singularités, c'est-à-dire des points de l'espace de paramètres où ce gradient n'est pas prolongeable par continuité, tout en restant fini au voisinage de ces points.

En pratique Dans notre implémentation, l'évaluation du gradient de $d(\mathbf{x},P)$ par rapport aux paramètres de P tient compte des éventuelles singularités de la manière suivante : lorsque l'on se trouve à une telle singularité (c'est-à-dire en un point de l'espace de paramètres où $g = 0$ dans l'équation 4.13), on attribue la valeur 0 aux composantes du gradient de d qui sont indéterminées en ce point (c'est-

TAB. 4.2 – Décomposition et singularités de la distance exacte pour chaque primitive

Primitive P	Param. $\mathbf{a} \in \mathbb{R}^p$	g_1	g_2	g_3	h	Lieu des singularités
Point 3D	$\mathbf{c}$	$(\mathbf{x}-\mathbf{c}).\mathbf{e_x}$	$(\mathbf{x}-\mathbf{c}).\mathbf{e_y}$	$(\mathbf{x}-\mathbf{c}).\mathbf{e_z}$	0	Ce même point ($\mathbf{c}$ tel que $\mathbf{c}=\mathbf{x}$)
Droite 3D	a, b, θ, ϕ	$(\mathbf{x}.\mathbf{v})-a$	$(\mathbf{x}.\mathbf{w})-b$	0	0	Cette même droite (a, b, θ, ϕ tel que $\mathbf{x} \in$ Droite)
Cercle 3D	$\mathbf{c}, \theta, \phi, R$	$d(\mathbf{x}, \text{cyl})$	$(\mathbf{x}-\mathbf{c}).\mathbf{u}$	0	0	Ce même cercle ($\mathbf{c}, \theta, \phi, R$ tel que $\mathbf{x} \in$ Cercle)
Sphère	$\mathbf{c}, r$	idem point			$-r$	Le centre ($\mathbf{c}, r$ tel que $\mathbf{c}=\mathbf{x}$)
Cylindre	a,b, θ,ϕ, r	idem droite			$-r$	L'axe (a, b, θ, ϕ, r tel que $\mathbf{x} \in$ Axe)
Cône	$\mathbf{c}$, θ, ϕ, α	Terme1 : $d(\mathbf{x}, \text{cyl})$ puis OK car produit et somme par fonctions $\mathcal{C}1$				Demi-axe ($\mathbf{c}, \theta, \phi, \alpha$ tel que $\mathbf{x} \in$ Axe)
Tore	$\mathbf{c}$, θ,ϕ, R,r	idem Cercle 3D			$-r$	Cercle directeur ($\mathbf{c}, \theta, \phi, R, r$ tel que $\mathbf{x} \in$ Cercle)

à-dire pour les paramètres qui apparaissent dans la fonction g). Les termes correspondants de la hessienne, qui n'est ici estimée qu'à partir des dérivées premières, sont également nuls en ces points.

Ceci ne semble pas poser problème en pratique, et ce pour deux raisons essentielles. Tout d'abord, comme remarqué précédemment, les lieux de singularités sont éloignés des primitives surfaciques. Aussi, lorsque des valeurs initiales correctes sont fournies, l'algorithme n'explore-t-il normalement pas les zones de singularités. D'autre part, si l'on se trouve près d'une discontinuité pour une fonction $d_i = d(\mathbf{x_i}, P)$, l'importance de cette discontinuité se trouve grandement atténuée par la somme de tous les autres termes correspondant aux autres $d_j = d(\mathbf{x_j}, P)$, qui eux correspondent peu probablement à une zone de singularité. Ce phénomène est d'autant plus vérifié que le nombre de points est grand (le nombre de points sur lequel on estime des primitives est souvent de l'ordre de 100).

Par contre, ceci peut éventuellement poser problème d'une part si l'on travaille avec peu de points, et d'autre part si l'on s'oriente vers des méthodes d'optimisation globale (par exemple : analyse par intervalles), auquel cas tout l'espace de paramètres est scruté.

Une alternative à l'approche proposée consiste à éliminer la racine carrée (responsable des dénominateurs dans le gradient) de la fonction distance, en considérant une approximation de la distance exacte dans le formalisme des moindres carrés. C'est l'approche suivie dans [LMM98, LMM97, MLM01], où l'on considère non plus la distance exacte $d = \sqrt{g} - h$ mais l'approximation

$$\tilde{d} = \frac{g - h^2}{2h}$$

Cette fonction distance $\tilde{d}$ a les même zéros que la distance exacte d, et *en ces points* son gradient par rapport aux paramètres est le même que celui de d. Ces propriétés sont illustrées sur la figure 4.15, qui

compare les graphes de d et de $\tilde{d}$ dans l'exemple du cercle vu précédemment. La figure 4.15 montre en outre un fait intéressant : la distance $\tilde{d}$ a tendance à donner plus d'importance aux valeurs extrêmes (pour lesquelles $d > 0$) que la distance exacte d. Ceci signifie que la fonction coût associée sera encore davantage sensible aux points aberrants que ne l'est la fonction coût associée à la distance exacte. La

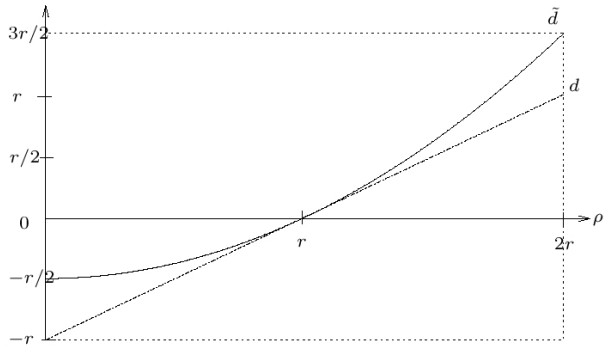


FIG. 4.15 – Distances exacte et distance approchée «Reccad» [LMM97] dans le cas du cercle (en 2D).

question est ensuite de savoir si la fonction coût construite à partir de la fonction $\tilde{d}$ a bien les mêmes minima globaux ou locaux que la fonction coût construite à partir de la distance exacte d. Ceci n'est pas évident a priori. En réalité, la justification majeure de l'utilisation de la distance $\tilde{d}$ est une question de simplicité et de temps calcul. Lorsque l'on souhaite obtenir un résultat précis, il est préférable de considérer la distance exacte.

4.3.4 Cas de séparabilité de l'espace de paramètres

On peut remarquer dans le tableau 4.1 que pour plusieurs des primitives introduites, la distance exacte d'un point $\mathbf{x}$ à la primitive prend la forme

$$d(\mathbf{x}; \text{Primitive}(\mathbf{a})) = f(\mathbf{x}; a_1, \ldots, a_{p-1}) - a_p \tag{4.15}$$

où l'on voit que l'un des paramètres, a_p, se trouve «séparé» des autres. Autrement dit, la surface est un ensemble de niveau d'une hyper-surface.

Ceci a pour conséquence que la fonction coût des moindres carrés s'écrit :

$$F = \sum_{i=1}^{N} d_i^2 = (\sum_{i=1}^{N} f(\mathbf{x}; a_1, \ldots, a_{p-1})^2) + N a_p^2 - (2 \sum_{i=1}^{N} f(\mathbf{x}; a_1, \ldots, a_{p-1})) a_p$$

Or, les paramètres étant libres, à l'optimum, on peut notamment écrire :

$$\frac{\partial F}{\partial a_p} = 0$$

ce qui donne l'équation :

$$2N a_p - (2 \sum_{i=1}^{N} f(\mathbf{x}; a_1, \ldots, a_{p-1})) = 0$$

soit

$$a_p = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}; a_1, \ldots, a_{p-1})) \tag{4.16}$$

On voit donc qu'une fois les paramètres $a_1, \ldots, a_{p-1}$ trouvés, le paramètre a_p peut s'exprimer analytiquement en fonction de ces derniers. Ceci signifie qu'il n'est pas utile d'effectuer l'algorithme d'optimisation avec tous les paramètres $a_1, \ldots, a_p$: on peut en effet, à chaque étape de l'algorithme, raisonner sur les paramètres $a_1, \ldots, a_{p-1}$ et mettre à jour a_p en fonction des paramètres $a_1, \ldots, a_{p-1}$ trouvés par la formule 4.16. La dimension de l'espace de paramètres dans lequel s'effectue la recherche peut ainsi être réduite de p à $p-1$. Ceci est intéressant à la fois du point de vue de la complexité algorithmique (la résolution du système de l'équation 4.11 est en $O(p^3)$), et donc en rapidité de calcul, mais aussi du point de vue de la qualité de la solution trouvée.

Une telle remarque a déjà été formulée sur ce sujet dans [Jia00] pour le cas du cylindre.

Dans notre contexte, les primitives libres concernées sont la sphère, le cylindre, et le tore. Ce n'est pas le cas du cône, en tout cas pas avec la paramétrisation utilisée, qui fait intervenir le sommet. Parmi les primitives contraintes, seul le «cylindre sécant» entre dans ce cas. Cette propriété n'a pour l'instant pas été utilisée au sein de notre implémentation, mais constitue une perspective intéressante.

4.4 Initialisation des paramètres

Un point crucial relatif à cette méthode est qu'elle nécessite une initialisation. La qualité, la précision et l'efficacité de l'ajustement dépendent fortement des valeurs initiales. En effet, la fonction coût peut présenter des minima locaux, comme l'illustre la figure 4.16. En outre, l'algorithme d'op-

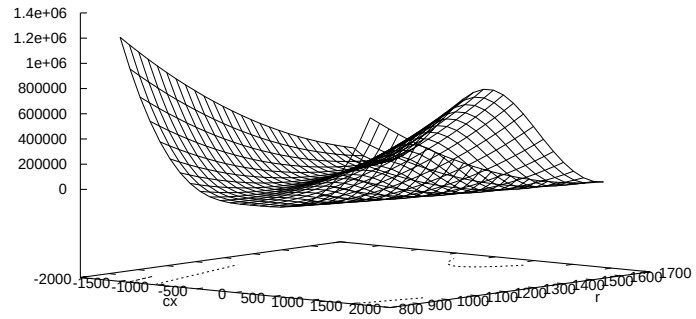


FIG. 4.16 – Profil de fonction coût. La fonction tracée dans cet exemple correspond à la fonction coût de la sphère en fonction des paramètres c_x et r, les deux autres paramètres (c_y, c_z) étant fixés à leur valeur finale obtenue après optimisation. On voit que le relief de la fonction coût forme des vallées distinctes, et que les fonctions coût sont susceptibles de contenir des minima locaux.

timisation de Levenberg-Marquardt ne garantit pas d'atteindre le minimum global. Les exemples des figures 4.3 et 4.4, donnés page 94, sont une illustration de cas où les valeurs initiales sont trop mauvaises pour que l'algorithme utilisé dans 3Dipsos parvienne à atteindre l'optimum global.

Ainsi, la philosophie de l'ajustement a été ici de produire une initialisation la meilleure et la plus cohérente possible, afin d'éviter les éventuels problèmes de minima locaux de la fonction coût.

D'autre part, étant donné que chaque primitive a une paramétrisation différente, il a été nécessaire de définir un procédé d'initialisation propre à chaque primitive. Dans cette section sont donc présentées successivement les méthodes de construction de valeurs initiales pour chacune des primitives introduites au 4.1.1. Un tableau récapitulatif des méthodes d'initialisation est donné à la fin de cette section, tableau 4.3.

Afin de produire des valeurs initiales correctes pour les paramètres des primitives, il est nécessaire d'utiliser des informations invariantes par mouvement solide (rotation et translation). C'est donc naturellement que l'on considère des informations locales de type géométrie différentielle : normales, sphère de Gauss, courbures, ou autres surfaces approximantes. En pratique les expérimentations utilisant les courbures n'ont pas donné de très bons résultats : résultats bruités, problème du choix de la surface approximante, forte sensibilité au choix de la taille de voisinage, qui est plus délicat que pour la normale. L'utilisation des normales (et de l'image de Gauss) s'est avérée plus fructueuse.

4.4.1 Primitives libres

De manière générale, plus la dimension de l'espace de paramètres est élevée, et plus la construction de valeurs initiales est délicate. Aussi, pour une primitive de type donnée, l'initialisation de la primitive libre est-elle plus difficile que la construction des valeurs initiales de la primitive contrainte.

Plan L'ajustement de plan par moindres carrés se fait de manière directe par un algorithme d'algèbre linéaire, et plus précisément d'extraction de valeurs-vecteurs propres (se reporter à l'annexe C).

Il n'est donc pas utile de réaliser l'ajustement par une méthode non-linéaire. On serait éventuellement amené à réaliser un ajustement de plan par une méthode non-linéaire si l'on utilisait une autre fonction coût que celles des moindres carrés, notamment une fonction coût plus robuste.

Sphère Deux possibilités ont été testées pour l'initialisation des paramètres de la sphère :
- ajustement par méthode linéaire à partir de la distance algébrique,
- utilisation du centre de gravité.

La première méthode consiste à utiliser une méthode directe faisant intervenir une approximation de la distance exacte d'un point à la sphère. Il est possible, en effet, en considérant la distance algébrique d'un point à la sphère (voir annexe C), de résoudre le problème par la même méthode que pour le plan. En pratique, ceci donne la plupart du temps une approximation très proche, voir identique à la solution finale après optimisation. L'optimisation n'est la plupart du temps pas cruciale ici. Par contre, lorsque les points sont très bruités, ou lorsqu'ils ne se trouvent pas réellement sur une sphère, il peut y avoir une grosse différence entre cette estimation initiale et la solution finale (après optimisation). Dans ces cas, il est important d'utiliser l'optimisation non-linéaire qui suit.

Nous avons testé une seconde méthode, qui consiste simplement à initialiser le centre de la sphère au centre de gravité des points, et le rayon à la moyenne des distances des points au centre de gravité. Cette initialisation est très souvent plus éloignée de la vraie solution que ne l'est l'initialisation issue de l'algorithme linéaire (Figure 4.17). En effet, le centre de gravité n'est proche du centre de la sphère que lorsque les points sont répartis tout autour de la sphère, ce qui est rarement le cas en

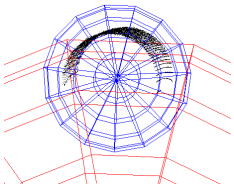


FIG. 4.17 – Valeurs initiales pour la sphère. La grande sphère correspond à l'initialisation par la méthode utilisant la distance algébrique, la petite sphère correspond à l'initialisation issue du centre de gravité

pratique. De ce fait, le rayon est en général sous-estimé car le centre de gravité se trouve plus proche des points que le centre de la sphère.

Cependant, nous n'avons pas trouvé de cas en pratique où cette initialisation mène à un minimum moins bon que l'initialisation précédente. Il semble même que les valeurs initiales obtenues mènent parfois à une meilleur minimum. Il semble donc dans certains cas que cette initialisation se situe dans une vallée plus intéressante de la fonction coût que l'initialisation issue du premier procédé. Ceci découle peut-être de propriétés géométriques (le centre de gravité se trouve dans l'enveloppe convexe des points, par exemple), mais la question n'a pas été étudiée plus avant.

Cylindre L'initialisation des paramètres du cylindre se déroule en deux étapes :

1. estimation de la direction du cylindre,
2. estimation de la position de l'axe et du rayon.

1. En ce qui concerne la direction du cylindre, le vecteur directeur unitaire **u** est estimé en introduisant l'image de Gauss. L'image de Gauss d'un cylindre, non nécessairement de révolution, est particulière : elle est inscrite sur un grand cercle de la sphère de Gauss (Figure 4.18). Ce grand

FIG. 4.18 – Image de Gauss d'un cylindre

cercle est l'intersection entre la sphère et un plan passant par son centre. De plus, la normale de ce plan correspond à la direction de l'axe du cylindre. Il suffit donc d'ajuster un plan de moindres carrés sur les points de l'image de Gauss, et de donner la normale unitaire à ce plan comme initialisation pour le vecteur **u**.

Lors de cet ajustement, le plan est contraint à passer par le centre de la sphère. Il s'agit donc d'un ajustement de plan contraint (voir annexe C), ce qui est important ici, car le plan de moindres carrés non-contraint risque d'être tangent à la sphère, et donc de ne pas passer du tout par le centre de la sphère. L'estimation est ainsi améliorée, en particulier dans les cas où la répartition angulaire des points sur le cylindre est faible.

Pour produire l'image de Gauss, la normale à la surface en chaque point est d'abord estimée. Il est important de noter que l'orientation des normales peut être quelconque ici, puisque que l'image de

Gauss d'un cylindre est symétrique par rapport à l'origine. C'est une bonne chose, car les normales estimées par le procédé utilisé, présenté en annexe B, n'ont pas d'orientation spécifique.

2. Une fois la direction du cylindre trouvée, l'estimation de la position de l'axe et du rayon est plus simple. En effet, si l'on projette les points sur un plan perpendiculaire à l'axe du cylindre, on obtient la section du cylindre, c'est-à-dire un cercle (pour un cylindre de révolution). Or, le cercle est une courbe algébrique du plan, et peut donc être estimé directement par une méthode d'algèbre linéaire utilisant une approximation de la distance exacte d'un point au cercle (semblable au cas de la sphère en 3D, voir annexe C). On obtient ainsi une estimation du centre du cercle et du rayon, qui donnent directement les valeurs pour a, b et r.

L'aspect intéressant de ce procédé d'estimation de valeurs initiales réside dans le fait qu'il ne dépend pas de la répartition des points sur la surface (à part pour l'estimation des normales unitaires). Ce n'est pas le cas de l'ajustement de cylindre sans contrainte [MEN01] actuellement implémenté dans le logiciel 3Dipsos : l'idée est schématiquement de considérer l'axe d'inertie des points comme axe initial du cylindre. Or on peut trouver des nuages de points se trouvant sur un cylindre et pour lequel l'axe d'inertie est très éloigné de l'axe du cylindre. Il peut en résulter un mauvais ajustement (même après optimisation), comme l'illustre la figure 4.3.

> Remarque
> En théorie, il est également possible de construire des valeurs initiales du cylindre à partir des centres de courbures, qui fournissent des points sur l'axe. Toutefois, l'information du centre de courbure cumule le bruit sur les normales et le bruit sur les courbures. Or, l'un des intérêts de la méthode présentée ici est de séparer, par le biais de l'image de Gauss, le bruit sur les normales du bruit radial. On obtient ainsi une meilleure estimation de la direction, ce qui est essentiel pour l'estimation des autres paramètres. Notons enfin qu'il est beaucoup plus lourd d'estimer les courbures que les normales (voir complexités algorithmiques dans l'annexe B).

Cône L'initialisation des paramètres du cône se déroule en deux étapes :

1. estimation de la direction de l'axe et du demi-angle au sommet,
2. estimation de la position du sommet.

De même que pour le cas du cylindre, l'image de Gauss d'un cône, non nécessairement de révolution, est particulière : elle est inscrite sur un petit cercle de la sphère de Gauss (Figure 4.19). Ce

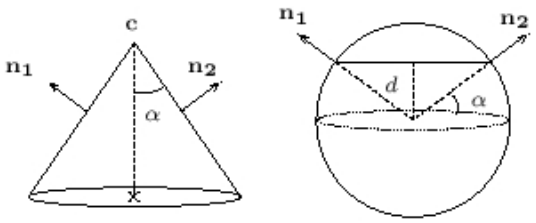


FIG. 4.19 – Image de Gauss du cône lorsque les normales sont orientées vers l'extérieur du cône

petit cercle est l'intersection entre la sphère et un plan ne passant pas nécessairement par son centre. La normale de ce plan correspond à la direction de l'axe du cône, et la distance du centre de la sphère au plan donne directement la valeur du demi-angle au sommet. Il suffit donc d'ajuster un plan de moindres carrés sur les points de l'image de Gauss, et de donner la normale unitaire à ce plan comme initialisation pour le vecteur $\mathbf{u}$. La distance d du centre de la sphère au plan fournit une estimation du demi-angle au sommet α :

$$\alpha = \arcsin(d)$$

A la différence du cas du cylindre, le plan n'est pas contraint à passer par le centre de la sphère. Il s'agit donc d'un ajustement de plan non-contraint (voir annexe C). Contrairement au cas du cylindre, il n'y a pas de garantie dans les cas où la répartition angulaire des points sur le cône est faible. On peut alors obtenir dans certains cas un plan pratiquement tangent à la sphère de Gauss. On peut toutefois envisager un critère visant à vérifier que la distance du centre de la sphère au plan est inférieure à 1.

D'autre part, il est important de noter que l'orientation des normales n'est plus quelconque ici, car l'image de Gauss n'est pas symétrique par rapport au centre de la sphère. En effet, si les normales sont quelconques, l'image de Gauss n'est plus inscrite sur un seul plan, mais sur deux plans parallèles symétriques par rapport à l'origine. Il est donc nécessaire de réorienter les normales pour obtenir une image telle que celle de la figure 4.19. En pratique, pour réorienter toutes les normales vers l'extérieur du cône, on procède à un ajustement de sphère de moindres carrés sur les points. Le centre $\mathbf{s}$ de la sphère fournit un point qui se trouve à l'intérieur du cône (il se situe plus au centre que le centre de gravité des points), et qui peut donc servir pour réorienter la normale $\mathbf{n_i}$ en chaque point $\mathbf{x_i}$:

$$\text{si } (\mathbf{n_i}.(\mathbf{x_i}-\mathbf{s})) < 0, \text{ on fait : } \mathbf{n_i} \leftarrow -\mathbf{n_i}$$

La sphère de moindres carrés a également l'avantage de fournir une estimation d'un point sur l'axe du cône, ce qui est utile pour positionner l'axe dans l'espace afin de déterminer le sommet du cône. En effet, une fois l'axe complètement spécifié (direction et position) et le demi-angle au sommet α estimé, il est aisé de produire une estimation du sommet du cône en projetant chaque point $\mathbf{x_i}$ sur l'axe avec l'angle α, comme suit : en introduisant (Figure 4.20) la projection $\mathbf{h_i}$ du point $\mathbf{x_i}$ sur l'axe $\mathbf{s}+\mathbb{R}\mathbf{u}$ (où $\mathbf{s}$ est le centre de la sphère), une estimation individuelle $\mathbf{c_i}$ du sommet du cône est fournie par

$$\mathbf{c_i} = \mathbf{h_i} - \frac{d(\mathbf{x_i}; \text{axe})}{\tan\alpha}\mathbf{u}$$

Comme on a $\mathbf{h_i} = \mathbf{s} + ((\mathbf{x_i}-\mathbf{s}).\mathbf{u})\mathbf{u}$, le point $\mathbf{c_i}$ s'écrit :

$$\mathbf{c_i} = \mathbf{s} + \left[((\mathbf{x_i}-\mathbf{s}).\mathbf{u}) - \frac{\|(\mathbf{x_i}-\mathbf{s})\times\mathbf{u}\|}{\tan\alpha}\right]\mathbf{u}$$

Le sommet $\mathbf{c}$ retenu est simplement la moyenne de ces projections $\mathbf{c_i}$.

> Remarque
> De même que pour le cas du cylindre, on peut construire des valeurs initiales du cône à partir des centres de courbures, qui fournissent des points sur l'axe. La remarque sur le bruit que nous avons faite pour le cylindre est encore valable dans ce cas. Mais se pose un problème supplémentaire ici, lié à la sensibilité du choix de la taille pour l'estimation de la courbure, plus critique dans le cas du cône (en pratique, on voit que les centres de courbures ne se trouvent pas trop sur l'axe...)

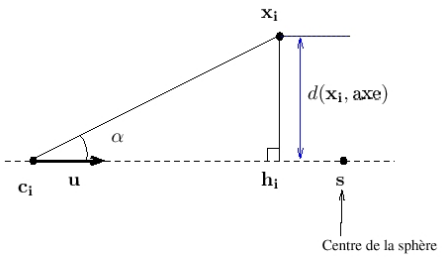


FIG. 4.20 – Projection des points pour l'estimation du sommet du cône

Tore L'estimation des paramètres du tore s'inspire du constat suivant. Sur un très petit voisinage, on peut approcher la surface du tore, comme toute autre surface lisse, par un plan (plan tangent). Lorsque l'on considère un voisinage un peu plus grand, on peut approcher la surface du tore par un cylindre (les tores traités ici n'ont pas d'auto-intersections). Le rayon d'un tel cylindre est égal au petit rayon du tore, et son axe est tangent au cercle directeur du tore.

On procède en deux étapes :

1. estimation du petit rayon r,
2. estimation du grand rayon R, du centre et de la normale au plan du cercle directeur.

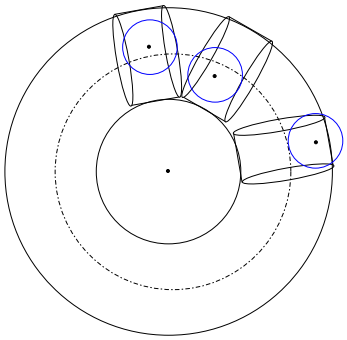


FIG. 4.21 – Principe d'initialisation des paramètres du tore par ajustement de cylindre sur des sous-nuages

1. Estimation du cercle directeur Dans la première étape, on construit des cylindres sur un certain nombre de voisinages dans le nuage de points. Ces cylindres donnent chacun un rayon, et la moyenne de ces rayons fournit une estimation du petit rayon du tore. Les voisinages sont déterminés en considérant les k plus proches voisins de points choisis aléatoirement (faute de mieux !) dans l'ensemble des N points.

2. Estimation du grand rayon Lors de la deuxième étape, on utilise la position et l'orientation des axes des cylindres pour déterminer plus globalement le tore.

Pour ce faire, on projette chacun des points aléatoires précédemment choisis sur l'axe du cylindre correspondant. Ce point se trouve approximativement sur le cercle directeur du tore.

Il suffit donc d'ajuster un plan de moindres carrés sur l'ensemble de ces projections pour obtenir le plan du cercle directeur, et sa normale **n**.

Ensuite, pour trouver le grand rayon R et le centre **c** du tore, on ajuste un cercle 2D de moindres carrés dans le plan trouvé (voir méthode annexe C). Ceci donne directement une estimation du centre du tore et du grand rayon.

Notons ici que deux paramètres interviennent : le nombre de voisinages et surtout la taille des voisinages (le nombre k de plus proches points). D'un côté, la taille des voisinages ne doit pas être trop petite pour que les cylindres ajustés soient corrects. D'autre part, cette taille ne doit pas être trop grande pour que le cercle directeur soit correct. Dans la pratique, cette taille a été laissée à une valeur assez haute : $N/2$. Le nombre de voisinages a quant à lui été fixé à la valeur minimum : 3.

4.4.2 Primitives contraintes

«Cylindre sécant» L'initialisation des paramètres du «cylindre sécant» se base sur la méthode d'initialisation du cylindre libre. On obtient ainsi un point sur l'axe **p**, un vecteur directeur **u**, et un rayon r.

Le rayon r fournit directement une initialisation pour le rayon du «cylindre sécant».

Le vecteur directeur **u** est également gardé comme initialisation.

D'autre part, en pratique, sauf cas exceptionnels (données synthétiques par exemple), l'axe du cylindre libre, $\mathbf{p}+\mathbb{R}\mathbf{u}$, n'est jamais exactement concourant avec l'axe du cylindre précédent, $\mathbf{p_0}+\mathbb{R}\mathbf{u_0}$. Par conséquent, pour estimer le paramètre μ, on cherche le point de l'axe du cylindre précédent dont la distance à l'axe $\mathbf{p}+\mathbb{R}\mathbf{u}$ est minimale. Ceci revient à chercher μ qui réalise le minimum de $d(\mathbf{p_0}+\mu\mathbf{u_0}, \text{Droite} : \mathbf{q}+\mathbb{R}\mathbf{u})$. On montre que ceci conduit à la valeur :

$$\mu = \frac{(\mathbf{q}-\mathbf{p_0}).[\mathbf{u_0}-(\mathbf{u_0}.\mathbf{u})\mathbf{u}]}{1-(\mathbf{u_0}.\mathbf{u})^2}$$

quand $\mathbf{u_0}$ et **u** ne sont pas colinéaires. Lorsque $\mathbf{u_0}$ et **u** sont colinéaires, on choisit $\mu = 0$ par convention.

«Cylindre sécant de rayon fixé» L'initialisation est la même que pour le «cylindre sécant» à ceci près que le rayon estimé n'est pas utilisé, puisqu'on le fixe à la valeur du rayon du cylindre précédent.

«Cylindre-rotule» L'initialisation de cette primitive a été volontairement réduite au plus simple. Le vecteur directeur de l'axe du «cylindre-rotule» est initialisé au vecteur directeur du cylindre précédent. Ceci fournit une initialisation valable dans les cas où l'angle que fait le «cylindre-rotule» avec le cylindre précédent n'est pas trop élevé.

«Tore après cylindre» L'estimation des paramètres suit ici le même principe que pour le tore libre.

Toutefois, les cylindres ajustés ne sont plus des cylindres libres, mais sont de type «cylindre sécant de rayon fixé», avec pour cylindre précédent le cylindre précédant le tore.

Ensuite, après l'estimation du cercle directeur, le grand rayon est calculé de sorte qu'en gardant le centre trouvé, le cercle directeur est tangent à l'axe du cylindre précédent.

«Tore entre cylindres» Cette primitive a été introduite pour améliorer l'ajustement d'une séquence «cylindre – tore après cylindre – cylindre après tore». Par définition, le deuxième cylindre est un «cylindre sécant de rayon fixé» par rapport au premier. Les paramètres initiaux sont donc ici simplement les paramètres du tore après cylindre.

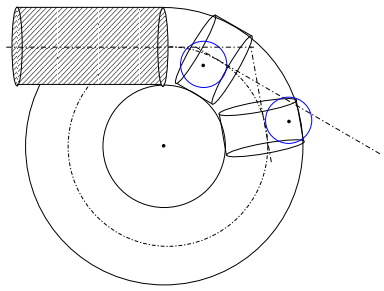


FIG. 4.22 – Principe d'initialisation des paramètres du «tore après cylindre».

«Cylindre après tore» De la même façon, le paramètre α est ici simplement déduit de l'extrémité (estimée en fonction du nuages points associé) du tore précédent.

Remarques

Notons qu'afin de trouver un meilleur optimum de la fonction coût, il est envisageable de fournir plusieurs valeurs initiales obtenues par des procédés différents et de réaliser successivement l'optimisation à partir de chaque valeur initiale, en retenant la meilleure solution obtenue. Ceci est particulièrement pertinent dans les cas où l'initialisation est délicate.

Parmi les procédés présentés, ceux qui font intervenir l'estimation d'informations locales telles que les normales ou les courbures sont coûteux en temps, ce qui peut devenir long lorsque le nombre de points N devient grand. Il est toutefois possible de procéder à un sous-échantillonnage des normales à estimer afin d'accélérer l'algorithme.

4.4.3 Tableau récapitulatif

Le tableau 4.3 montre un récapitulatif des différentes méthodes utilisées pour l'initialisation des paramètres de chaque primitive.

4.5 Discussion et perspectives

4.5.1 Aspect numérique et conditionnement

Les paramètres utilisés pour chaque primitive sont de natures très différentes, et par conséquent ont des ordres de grandeur différents. Par exemple, pour un cylindre, l'angle θ varie entre 0 et π, alors que le paramètre a peut varier entre 0 et 100 000 mm (100m), ou même davantage.

En conséquence, les termes de la hessienne peuvent être très différents. Il en résulte un mauvais conditionnement de cette matrice.

Or, en pratique, ce problème ne se pose pas vraiment car la matrice $\bar{\mathbf{H}}$ utilisée est diagonalement dominante. Ceci fait que la matrice n'est pas trop mal conditionnée, et que la résolution du système associée s'effectue sans problème.

TAB. 4.3 – Tableau récapitulatif des procédés d'initialisation

Primitive P	Paramètres libres	dim	Base pour initialisation
Sphère	c_x, c_y, c_z, r	4	Centre d'inertie et moyenne des distances
Cylindre	a, b, θ, ϕ, r	5	Normales. Ajust. Plan contraint, cercle 2D
Cône	$\theta, \phi, \mathbf{c}, \alpha$	6	Normales réorientées, ajust. Sphère, plan,
Tore	$\theta, \phi, \mathbf{c}, R, r$	7	Ajustement cylindres + plan + cercle 2D
Cylindre sécant	μ, θ, ϕ, r	4	similaire Cylindre libre
Cylindre sécant de rayon fixé	μ, θ, ϕ	3	similaire au cylindre libre
cylindre-rotule	θ, ϕ	2	axe du cylindre précédent
Cylindre après tore	α	1	Normales
Tore après cylindre	$\mathbf{c}$	3	Ajustement cylindres sécants de rayon fixé
Tore entre cylindres	R	1	(séquence cyl-tore-cyl)

4.5.2 En présence de points aberrants

Comment rendre cette méthode plus robuste par rapport aux points aberrants ? Un point aberrant est un point dont il est clair qu'il n'appartient pas au modèle qui correspond aux autres points (points avec un bruit inexpliqué, morceau d'une autre surface, points interpolés entre surfaces pour certains scanners (voir § 3.2.3, page 58)).

La méthode des moindres carrés est notoirement peu robuste par rapport aux points aberrants : un petit nombre de données aberrantes peuvent modifier complètement le résultat de l'ajustement, et donner un résultat qui ne correspond pas du tout à la réalité «visuelle».

Il faut considérer deux choses par rapport à la présence de points aberrants :

1. l'initialisation peut devenir mauvaise (car elle-même souvent basée sur des procédés d'ajustement de moindres carrés).
2. la fonction coût devient mauvaise également. En effet, pour observer ce phénomène, il suffit de considérer un cas qui ne nécessite pas d'initialisation : la droite de moindres carrés en 2D se calcule par exemple directement en utilisant la distance exacte (euclidienne). En présence de points aberrants, la droite de moindres carrés diverge assez facilement. Or dans ce cas on sait que l'on atteint le minimum global de la fonction coût car la solution est analytique. Ceci signifie que la fonction coût est inadaptée.

Pour réaliser un ajustement robuste par rapport aux points aberrants, il convient donc de jouer sur les deux étapes.

L'extension de la méthode présentée dans ce chapitre à d'autres fonction coût est possible. En particulier, on peut utiliser des fonctions coût correspondant aux M-estimateurs. On peut ainsi partir de l'initialisation des moindres carrés pour ensuite optimiser une autre fonction coût. Cependant, en

pratique, les essais avec d'autres fonctions coûts de type M-estimateurs et les mêmes initialisations ne semblent pas conduire à des résultats très différents. La raison est probablement la présence de vallées dans les fonctions coût. Ceci signifie qu'il faut produire également des initialisations robustes, et/ou introduire des fonction coût de nature différente, comme par exemple les fonction coût qui interviennent dans les méthodes d'extraction (voir chapitre 6). Ceci étant, comme il est dit au chapitre 6, l'utilisation locale des moindres carrés paraît être l'approche la plus efficace (notons que l'on peut toutefois envisager de compléter la méthode des moindres carrés par des méthodes robustes, telles que celles présentées au chapitre 6).

4.5.3 Résultats et remarque

Des résultats d'ajustement de plan, sphère, cylindre, cône et tore (primitives libres) sont présentés dans les chapitres 5 et 6. L'ajustement de primitives contraintes est utilisé dans les algorithmes d'extraction et de modélisation de lignes de tuyauterie du chapitre 3.

Dans le domaine académique, on peut noter l'absence de tests expérimentaux sur ce sujet (en partie due au fait qu'il y a encore très peu de travaux de recherche sur ce sujet à l'heure actuelle). Il semblerait intéressant de former des bancs de tests pour comparaisons expérimentales de tous les systèmes d'ajustement de telles primitives (banque de nuages de points 3D avec variations de bruit, de répartition des points, etc.). De telles scènes peuvent être produites par simulation, l'intérêt des scènes simulées étant que les vraies solutions de moindres carrés sont connues et que l'on peut contrôler tous les paramètres (bruit, répartition des points, etc.). Dans ce travail, nous avons notamment pu utiliser un simulateur de scanner, «simulaser», développé au Centre de Robotique. Ce type de bases de données de tests, comprenant des données simulées et réelles, existent dans le cas d'images 2.5D pour la comparaison d'algorithmes de segmentation d'image de profondeurs [HJBJ+96, RC88].

4.5.4 Nouvelles primitives

En ce qui concerne les applications du chapitre 3, il serait utile d'introduire des primitives supplémentaires, telles que la réduction concentrique (cône contraint joignant un cylindre de même axe), la réduction excentrique, le cône coaxial et/ou le cône sécant, le «conic stripe», le cylindre coaxial (Figure 4.23).

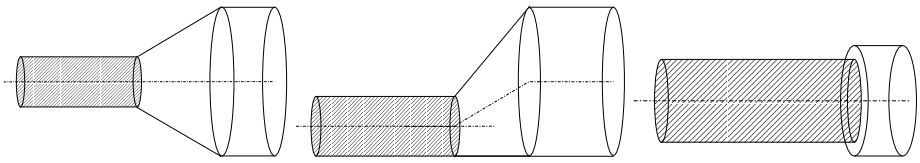


FIG. 4.23 – Autres primitives d'intérêt

4.5.5 Ajustement de primitives plus complexes

Il est possible d'étendre cette méthode à des primitives plus complexes, de type objets paramétrés : brides, ligne de tuyauterie (au moins pour une séquence fixée de primitives), poutres, etc. on peut en effet définir, à la manière du cône, la distance exacte pour une primitive plus complexe (en considérant des zones). On voit sur les exemples des figures 4.24, 4.25, 4.26, que l'on peut composer de nombreuses primitives nouvelles en utilisant seulement les surfaces traitées : la distance exacte à

une primitive composée est simplement définie par les différentes distances aux primitives simples introduites ici (par zones).

Le problème majeur réside alors dans l'étape d'initialisation des paramètres, qui elle peut s'avérer très complexe.

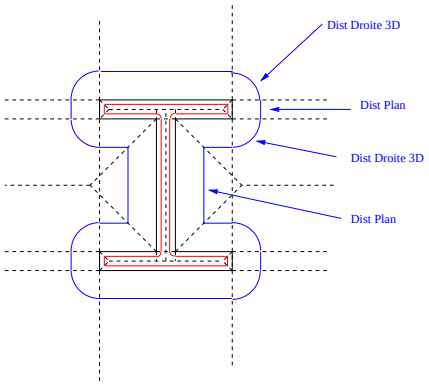


FIG. 4.24 – Section d'une poutre en I et lignes d'isodistances

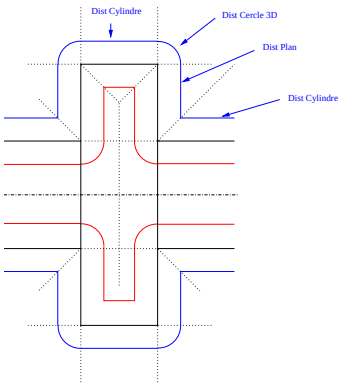


FIG. 4.25 – Modèle de bride et lignes d'isodistances

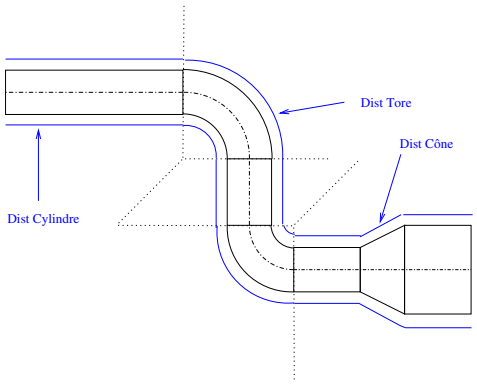


FIG. 4.26 – Modèle d'une ligne de tuyauterie et lignes d'isodistances

Chapitre 5

Sélection et validation de modèle

Ce chapitre traite de la reconnaissance de surface. Plus précisément, nous tentons de répondre aux questions suivantes :
Comment reconnaître la forme d'une surface formée par un nuage de points ?
Peut-on identifier le type de primitive qui est présente ?
Comment déterminer si le nuage de points suit bien la forme de l'une des primitives recherchées ?

Dans ce but, deux situations légèrement distinctes de reconnaissance de la surface formée par des points sont considérées :

1. La sélection de modèle : ayant ajusté différentes primitives sur un même nuage de points, l'on souhaite déterminer quelle est la meilleure des primitives parmi les différents types possibles (Figure 5.1). Cette question est traitée en section 5.1.
2. La validation de modèle : ayant ajusté une primitive, l'on souhaite examiner si cette primitive est acceptable ou non au vu des données (Figure 5.2). Cette question est traitée en section 5.2.

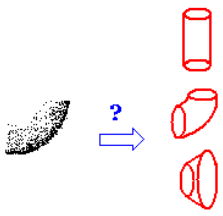


FIG. 5.1 – Sélection de modèle

Nous avons à ce jour très peu utilisé la sélection de modèle dans les applications du chapitre 3. Cependant, cette question a été étudiée en ce qu'elle jouera un rôle important au niveau de la modélisation finale de ligne de tuyauterie (section 3.3). La sélection de modèle intervient d'autre part dans l'extraction de primitives de types non-fixés, thème essentiel pour la segmentation en primitives géométriques, que nous avons commencé à aborder (section 6.4).

La validation d'un modèle est une question très générale qui se pose dans toutes les situations où l'on ajuste un modèle sur des données. Nous nous sommes en particulier intéressés à cette question pour résoudre le problème du critère d'arrêt de l'algorithme d'extraction de lignes de tuyauterie (section 3.2).

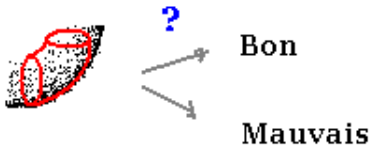


FIG. 5.2 – Validation de modèle

Remarque sur l'emploi de l'ajustement pour la reconnaissance

Le choix fait ici est d'étudier des méthodes de reconnaissance de surface qui mettent en œuvre directement l'ajustement des modèles recherchés (chapitre 4). Il pourrait sembler a priori naturel de s'intéresser aux méthodes qui ne font pas intervenir directement l'ajustement de modèle. En particulier, on peut considérer tout d'abord des informations dites «bas niveau», c'est-à-dire dont l'estimation ne dépend pas du modèle, telles que les normales, les courbures ou l'image de Gauss.

L'expression théorique de la normale unitaire en chaque point ainsi que des courbures est présentée pour chaque primitive en annexe B. On voit que les normales ont des valeurs bien déterminées, mais dépendent de la position du point sur la surface, ainsi que des paramètres de forme de la surface. Il n'est pas évident de déterminer un critère de reconnaissance à base de ces informations individuelles.

D'autre part, l'image de Gauss a déjà été utilisée pour l'initialisation des paramètres de certaines primitives au chapitre 4. En particulier, l'information fournie par l'image de Gauss est particulièrement adaptée aux surfaces développables (contenant une droite en tout point) puisque dans ce cas elle forme une courbe sur la sphère de Gauss. Parmi les primitives traitées ici, ceci concerne le plan, le cylindre et le cône. Les images de Gauss de la sphère et du tore paraissent plus difficiles à exploiter.

Enfin, il est également possible de classer les surfaces à partir des courbures (se reporter aux expressions des courbures en un point présentées pour chaque primitive en annexe B, voir également le graphe des courbures dans [Gou99]). Le problème en pratique est lié au caractère bruité des courbures estimées.

Il s'avère donc délicat de réaliser la reconnaissance de surfaces à partir de ces informations «bas niveau» seules. Ceci rejoint la conclusion du chapitre 2 concernant la nécessité d'introduire les primitives au sein même du procédé de segmentation. Aussi les algorithmes de reconnaissance de primitives consistent le plus souvent à 1. appliquer les méthodes d'ajustement des primitives recherchées, et 2. examiner ensuite les résultats de ces ajustements pour reconnaître la surface. C'est la situation qui est étudiée dans ce chapitre.

> Remarque
> Pour que la sélection et la validation de modèle puissent être cohérentes, il est nécessaire d'utiliser un procédé d'ajustement «sûr», au sens où pour chaque type, la primitive trouvée par l'algorithme d'ajustement doit effectivement réaliser le minimum de la fonction coût (des moindres carrés dans notre cas). D'où l'intérêt de définir avec soin les fonctions d'ajustement (chapitre 4), et en particulier les initialisations (section 4.4).

5.1 Sélection de modèle

Le cadre est ici de trouver, étant donné un ensemble de points, la primitive qui ajuste le mieux les données parmi un nombre fini de types de primitives possibles.

L'idée intuitive la plus couramment utilisée est de choisir le type de modèle dont l'erreur d'ajustement est la plus faible parmi les erreurs trouvées pour les différents types de modèles.

Le cas le plus fréquemment traité est celui de la régression linéaire. C'est pourquoi ce cas est d'abord présenté. On examine ensuite le cas des primitives géométriques.

5.1.1 En régression linéaire

On considère le modèle linéaire sous hypothèse gaussienne :

$$Y = Xb + u, \text{ où } u \sim \mathcal{N}(0, \sigma^2 I_N)$$

où Y et u sont des vecteurs de $\mathbb{R}^N$, X est une matrice $N \times p$, I_N est la matrice identité $N \times N$, et b est le vecteur de paramètres, à valeur dans $\mathbb{R}^p$.

> Remarque
> Notons dès à présent que l'on utilise le terme de modèle linéaire car le modèle est linéaire en les paramètres b. En particulier, le cas des surfaces $z = a^T q(x,y)$ entre dans ce cadre. Ceci inclut notamment les surfaces $z = f(x,y)$ polynomiales. On aurait alors $Y = (z_1, \ldots, z_N)^T$, $X = (q_i(x_j, y_j))_{i,j}$, et $b = a$.

Le problème des moindres carrés se formule :

$$\min_b \|Y - Xb\|^2$$

La solution $\hat{b}$ de ce problème s'écrit (annexe C) :

$$\hat{b} = (X^T X)^{-1} X^T Y$$

et l'erreur correspondante $\hat{u}$ est telle que :

$$\hat{\sigma}^2 = \frac{\|\hat{u}\|^2}{N - p}$$

est un estimateur sans biais de σ^2 ($\|.\|$ désigne ici la norme euclidienne de $\mathbb{R}^N$). Sans donner la démonstration, disons simplement que ceci découle du fait que :

$$(N - p)\frac{\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{N-p}$$

où χ^2_{N-p} désigne la loi du chi-deux à $N - p$ degrés de liberté.

Critère de sélection Le principe de la sélection est de comparer ces quantités $\hat{\sigma}$ d'un modèle à l'autre, en cherchant le modèle qui minimise cette quantité.

Autres critères

Le critère basé sur la quantité $\hat{\sigma}$ n'est pas le seul possible. En effet, de nombreux critères de sélection de modèles existent dans la littérature, ne serait-ce que pour le problème de la régression linéaire [BS98]. Les différents critères se distinguent par la manière dont ils pénalisent les modèles ayant de nombreux paramètres par rapport aux modèles ayant peu de paramètres. La plupart de ces critères sont issus de la théorie de l'information : critère d'information d'Akaike (*Akaike Information Criterion* ou AIC) [Aka74, BMG94, FFE97], critère d'information bayésien (*Bayesian Information Criterion* ou BIC) [JOM00, LMM98], longueur de description minimum (*Minimum Description Length* ou MDL) [PF96]. En outre, les critères peuvent être composites [LMM98]. Ces différents critères ne sont pas présentés plus en détail ici, car nous n'avons pas jugé nécessaire de les utiliser.

5.1.2 Pour des primitives géométriques 3D

Nous avons vu que l'ajustement de primitive géométrique tel que celui présenté au chapitre 4 ne s'exprime pas comme un problème de régression (§ 4.1.3, page 95). Ce cas n'entre a fortiori pas comme un problème de régression linéaire (notons que linéaire signifie ici linéarité en les paramètres b). En toute rigueur, la quantité $(N-p)\frac{\hat{\sigma}^2}{\sigma^2}$ ne suit donc plus exactement une loi χ^2_{N-p}, et $\hat{\sigma}^2$ n'est pas exactement un estimateur sans biais de σ^2.

En régression non-linéaire, il est commun, et en général «pas trop faux»[PTVF92b, p.660], de considérer le modèle de régression comme localement linéaire. Aussi suppose-t-on que la quantité $(N-p)\frac{\hat{\sigma}^2}{\sigma^2}$ suit une loi χ^2_{N-p}, et donc que e' est un estimateur sans biais de σ^2.

Qu'en est-il pour notre problème ? En fait, le théorème des fonctions implicites indique que l'on peut décrire une primitive géométrique par morceaux en utilisant des fonctions $z = f(x,y)$ (dans des repères locaux). Intuitivement, il suffit de découper la surface en zones sur lesquelles on peut projeter la surface sur un plan. Sur ces zones, en considérant une direction adéquate, la fonction implicite mène à une formulation $z = f(x,y)$. Ceci fournit une justification, intuitive plutôt que rigoureuse de l'utilisation de la quantité $\hat{\sigma}$ comme critère de sélection pour des primitives géométriques.

Indépendamment de la justification, ce critère peut quoiqu'il en soit être utilisé. C'est ce que nous avons fait. Nous avons donc considéré la quantité e', qui a la même définition que $\hat{\sigma}^2$ dans le cas de la régression linéaire, mais qui s'écrit plus généralement :

$$e'^2 = \frac{1}{N-p}\sum_{i=1}^{N} d(\mathbf{x_i}; M)^2 \tag{5.1}$$

où p est le nombre de paramètres du modèle M, et d est la mesure de l'écart entre le point $\mathbf{x_i}$ et le modèle. Notons que e' est simplement relié à l'erreur quadratique (ou résidu quadratique moyen) e

$$e = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d(\mathbf{x_i}; M)^2} \tag{5.2}$$

par :

$$e'^2 = \frac{N}{N-p}e^2$$

La quantité e' a donc tendance à avantager les modèles ayant peu de paramètres par rapport aux modèles ayant un grand nombre de paramètres.

> Remarque
> La distinction entre e' et e n'est réellement significative que lorsque le nombre de points N est petit. En effet, lorsque le nombre de points N est très grand, le rapport e'/e est très proche de 1. La normalisation par $N-p$ au lieu de N est donc superflue dans ce cas. D'ailleurs, dans plusieurs travaux, l'erreur e' est utilisée directement pour traiter le cas du cylindre, du cône, du plan et de la sphère [LMM98, NFJ93]).

Critères existants spécifiques au cas de primitives géométriques

Peu de travaux sortent du cadre de la régression. En particulier, on considère rarement le problème spécifique de la sélection de modèle parmi des modèles «géométriques», c'est-à-dire dont l'équation s'écrit :

$$f(\mathbf{x}; \mathbf{a}) = 0$$

où **x** est un point (données) et **a** est le vecteur de paramètres. Les travaux de K. Kanatani [Kan01, Kan00, Kan93] sont parmi les seuls à examiner ce problème du point de vue statistique. En particulier, ses travaux ont conduit à la définition d'autres critères : GAIC (*Geometric AIC*), GMDL (*Geometric MDL*). Ces critères sont construits spécifiquement pour comparer deux modèles entre eux. Sans présenter la méthode pour parvenir à ces critères, voici simplement l'expression du critère GAIC sur un exemple, dans le cas de la sélection entre un cylindre et un cône :

$$\begin{aligned} \text{GAIC(cylindre)} = &\quad Ne_{\text{cylindre}} + 2(2N+5)\frac{Ne_{\text{cône}}}{N-6} \\ \text{GAIC(cône)} = &\quad Ne_{\text{cône}} + 2(2N+6)\frac{Ne_{\text{cône}}}{N-6} \end{aligned}$$

Ces formules s'écrivent directement à partir des expressions de [Kan01]. La comparaison des deux quantités revient donc à évaluer :

$$\text{GAIC(cylindre)} - \text{GAIC(cône)} = N\left[(e_{\text{cylindre}} - e_{\text{cône}}) - 2\frac{e_{\text{cône}}}{N-6}\right]$$

et de choisir le cylindre si cette quantité est négative et le cône si cette quantité est positive. Au vu de cette formulation, le constat que nous faisons est le même que celui fait pour e' : lorsque N devient grand, ce critère est équivalent à compare les erreurs e. En outre, le critère GAIC est plus difficile à mettre en œuvre lorsque l'on a plus de deux modèles car il prévoit une comparaison des modèles deux à deux.

5.1.3 Cas dégénérés

Indépendamment du critère, le point crucial dans le domaine de la sélection de modèle concerne les cas dégénérés. Les termes «modèles emboîtés» (*nested models*), «sur-modèle», ou «sur-ajustement» (*over-fitting*) désignent la même notion. Pour se faire une idée, considérons un exemple de régression linéaire où l'on ajuste un modèle de type $y = ax + b$ et un modèle de type $y = ax^2 + bx + c$. Si l'on fait abstraction de l'aspect numérique, l'ajustement du deuxième modèle est toujours meilleur que l'ajustement du premier modèle. En effet, le deuxième modèle est supérieur au premier en ce qu'il a la possibilité de devenir comme le premier mais qu'il décrit également d'autres situations. Ceci veut dire que si l'on regarde simplement l'erreur quadratique, le deuxième modèle est toujours le meilleur, même si les données sont en réalité sur un modèle du premier type. En fait, si les données sont parfaites (sans bruit), les deux modèles donnent des résultats équivalents. Par contre, si les données sont bruitées, le second modèle ajuste mieux les données, même si celles-ci se trouvent — au bruit près — sur un modèle du premier type. L'idée intuitive est claire : pour le second modèle, il existe un paramètre en plus, donc un degré de liberté supplémentaire, c'est-à-dire plus de moyens de s'adapter aux formes des données. Le concept de cas *dégénéré* désigne alors la situation où le second modèle est pratiquement égal au premier modèle.

Parmi les primitives que nous avons introduites, il existe de tels cas (Figure 5.3). Les cas de dégénérescence entre primitives sont indiqués sur le schéma de la figure 5.4.

> Remarque
> Notons que les cas dégénérés dépendent de la paramétrisation des primitives. En effet, dans [MLM01], les paramètres choisis pour le tore sont tels qu'un cône est un cas dégénéré de tore, ce qui n'est pas le cas ici.

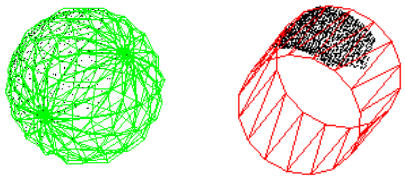


FIG. 5.3 – Cas dégénérés : tore dégénéré en sphère (à gauche), cône dégénéré en cylindre (à droite).

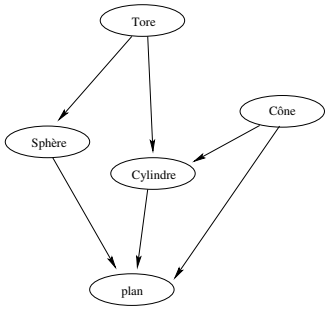


FIG. 5.4 – Hiérarchie des cas dégénérés parmi les primitives libres avec les paramétrisations décrites section 4.3.

D'autre part, le diagramme de la figure 5.4 est théorique. En pratique, cela dépend des paramétrisations choisies pour les primitives. Plus précisément, le comportement est différent si l'on choisit des paramètres tels que les cas dégénérés peuvent être atteints ou tels que les cas dégénérés sont seulement des cas limites. Par exemple, dans notre paramétrisation, pour qu'un cône devienne un cylindre, il faut que le demi-angle au sommet soit nul et que le sommet se trouve à l'infini. Le demi-angle au sommet peut fort bien atteindre la valeur 0. Par contre, le sommet ne peut pas atteindre l'infini, et la valeur retournée dépend de l'algorithme d'estimation. Par contre, si l'on choisit comme paramètres, au lieu du sommet du cône, la projection orthogonale de l'origine sur l'axe du cône et une valeur de rayon à ce point (comme c'est le cas dans [MLM01]), alors le cône peut atteindre son cas dégénéré cylindrique. En résumé, pour les paramètres qui ont été choisis ici, les cas dégénérés ne sont pas toujours tout à fait atteints. Ceci dépend également des procédés d'initialisation. Ceci signifie par exemple que sur un nuage de points représentant un cylindre, l'ajustement d'un cône sur ces points ne donnera pas systématiquement une meilleure erreur quadratique que l'ajustement du cylindre, alors que ceci devrait être le cas en théorie. Ceci ne constitue pas forcément un problème, et peut même être considéré comme intéressant, dans la mesure où notre approche est justement de trouver des primitives non-dégénérées, en essayant successivement tous les types de primitives.

Afin d'éliminer les cas dégénérés, la méthode employée a été d'imposer des contraintes sur la valeur de certains des paramètres des primitives. La figure 5.5 montre les contraintes utilisées pour chaque lien de dégénérescence.

Les primitives contraintes, telles que celles présentées au chapitre 4, sont également sujettes aux cas dégénérés. Un «tore après cylindre» peut par exemple dégénérer en «cylindre sécant de rayon fixe». Un réducteur (excentrique ou non) peut dégénérer en cylindre. Un «cylindre sécant» peut devenir un «cylindre sécant de rayon fixé», qui peut lui-même devenir un «cylindre-rotule» (mais la

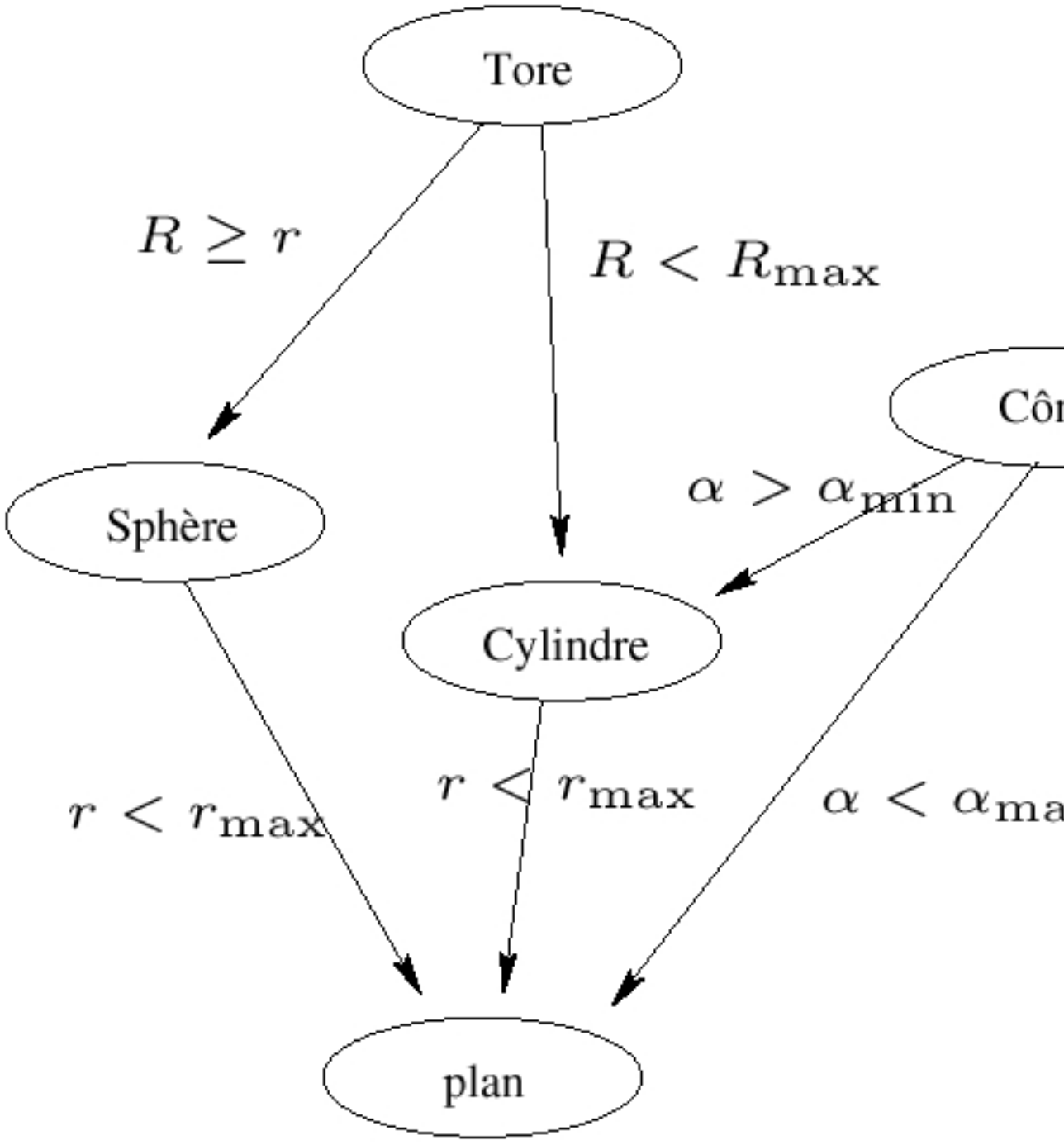


FIG. 5.5 – Contraintes imposées sur les paramètres pour éviter les primitives dégénérées

primitive sous-jacente est toujours un cylindre). Les contraintes appliquées ici sont essentiellement celles introduites pour les primitives libres. Les deux seuls cas de contraintes propres aux primitives contraintes qui ont été intégrées sont :

- une limite de l'angle formé par un «cylindre-rotule» avec le cylindre précédent, imposé inférieur à 45˚ en pratique,
- une limite sur le rapport entre grand et petit rayon pour le «tore après cylindre», imposé inférieur à 20, pour les applications du chapitre 3.

Ces valeurs limites dépendent des scènes à traiter : du type de scène, mais aussi de l'unité des points mesurés (mm, m, pieds,...). En pratique, elles sont spécifiées par l'utilisateur dans le logiciel (Figure 5.12, page 139).

Au final, la primitive choisie est celle rendant la quantité e' minimum parmi celles qui sont compatibles avec les contraintes. Comme il n'y a pas de contraintes sur le plan, l'algorithme rend toujours une primitive.

5.1.4 Résultats et conclusion

Résultats

Le tableau 5.1 donne les valeurs de la quantité e' relativement à l'ajustement de chaque primitive sur quelques scènes. Les points de chaque scène utilisée se trouvent (au bruit près) sur une primitive de type déterminé (qui définit le nom de la scène). Ici, les nombres de points sont tels qu'il n'y a pratiquement pas de différence entre les quantités e et e'. Ce tableau montre clairement deux situations :

TAB. 5.1 – Valeurs de la quantité e' pour l'ajustement de chaque primitive pour quelques scènes (en mm) :

Scène	Nb de pts	plan	sphère	cyl	cône	tore
Tore0	5407	116.22	76.60	35.35	35.10	4.66
Cône0	664	16.99	2.85	1.43	0.48	1.44
Cyl0	7946	80.00	76.00	2.30	2.29	5.49
Sphère0	388	7.83	0.59	5.28	2.10	0.58
Plan0	1169	2.27	2.22	2.44	2.27	179.28

- le cas où aucune primitive ne peut dégénérer en la «vraie» primitive,
- le cas où des primitives peuvent dégénérer en la primitive.

Lorsqu'il n'y a pas de cas de dégénérescence, l'erreur e'(ou e) minimale correspond bien à la primitive adéquate. C'est le cas pour les scènes «Tore0» et «Cône0» : les minima de e' désignent bien respectivement un tore et un cône. Ceci se comprend aisément puisque aucune autre primitive parmi celles disponibles ne peut dégénérer en tore ou cône (Figure 5.4). Etant donné que les contraintes n'interviennent pas dans ce type de cas, la sélection de modèle est ainsi réalisée sans problème (Figure 5.6).

Lorsqu'il y a théoriquement dégénérescence, celle-ci est atteinte la plupart du temps. C'est le cas pour les scènes «Cyl0», «Sphère0» et «Plan0». De plus, on constate que pour ces scènes, les cas

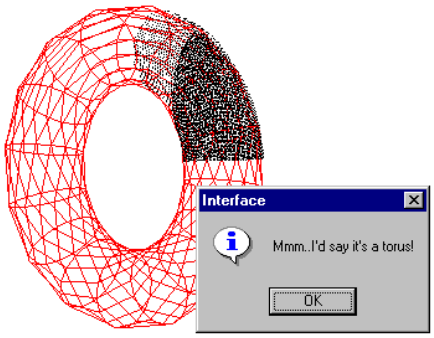


FIG. 5.6 – Résultat de la sélection de modèle sur la scène «tore0»

dégénérés sont ceux qui rendent minimum la quantité e'. Par exemple, pour la scène «Cyl0», le cône rend la quantité e' minimum. Sur ce même exemple, le résultat obtenu avec le critère GAIC défini précédemment est équivalent ; on trouve en effet :

$$\text{GAIC(cylindre)} - \text{GAIC(cône)} = 44,5 > 0$$

ce qui mène également à accepter le cône comme meilleur modèle.

Dans les exemples du tableau, les contraintes imposées sur les paramètres ont permis d'éliminer les cas dégénérés (Figure 5.7). En pratique, lorsque la scène est constituée d'une seule primitive, les

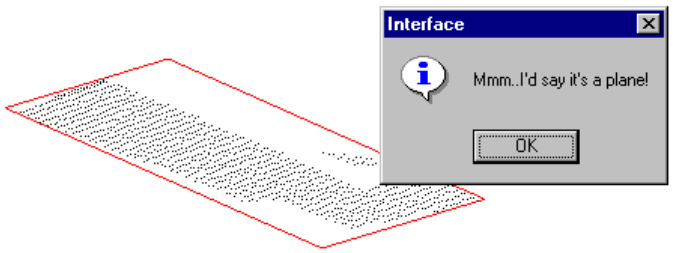


FIG. 5.7 – Sélection de modèle sur la scène «Plan0». On peut noter que les contraintes ont été utilisées ici, puisque la valeur de e' correspondant à l'ajustement de sphère est ici inférieure (voir tableau 5.1)

contraintes sur les paramètres permettent la plupart du temps d'éliminer les cas dégénérés[1].

Limites de la sélection de modèle

En pratique, dans un contexte où la définition du nuage de points est automatique (comme dans l'algorithme du §3.2), il existe des cas où aucun des modèles ne représente les données de manière satisfaisante. En effet, le nuage de points considéré peut contenir plusieurs surfaces. Dans ce cas, la sélection de modèle rend la primitive qui ajuste au mieux les données parmi celles possibles. Celle-ci n'est pas forcément satisfaisante, mais ceci n'est pas pour autant détectable par les contraintes sur les paramètres (Figure 5.8). La sélection de modèle que nous avons décrite dans cette section ne suffit

[1] Ceci bien entendu lorsque les valeurs extrémales utilisées dans ces contraintes sont raisonnables.

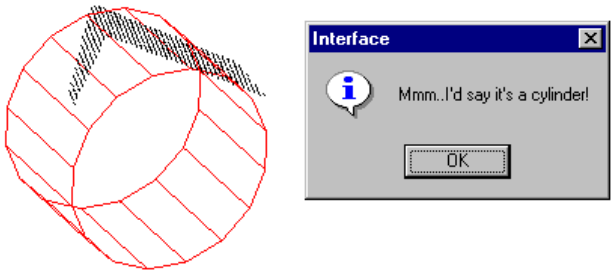


FIG. 5.8 – Exemple montrant les limites de la sélection de modèle sur une scène comportant deux plans.

donc pas pour reconnaître la ou les surfaces : il est nécessaire de définir des moyens de valider ou d'invalider chaque surface.

Conclusion

Il existe de nombreux critères statistiques de sélection de modèle, qui ont en général les même propriétés asymptotiques (lorsque le nombre de points N est grand) mais diffèrent pour les faibles valeurs de N. Aucun critère n'apporte de solution miracle, si les cas dégénérés ne sont pas clairement caractérisés.

Notre approche consiste donc à considérer un critère de sélection simple (e ou e'), et à invalider les cas de dégénérescence par des contraintes sur les paramètres. Ces contraintes requièrent une certaine connaissance de la scène. C'est pourquoi certaines de ces contraintes sont spécifiées par l'utilisateur (section 5.2.1).

D'autre part, dans le contexte de l'extraction automatique de primitives ou de lignes de tuyauterie, les sous-nuages des points considérés ne se trouvent pas nécessairement sur une seule primitive. Il existe également dans certaines zones des formes complexes qui ne sont pas modélisables par des primitives simples. Aussi convient-il d'envisager de compléter ce principe par une validation de modèle (section 5.2).

5.2 Validation de modèle

On a fait l'hypothèse que les points observés se trouvent sur une primitive géométrique d'un type donné et on a ajusté une telle primitive aux points (par exemple par une méthode de moindres carrés). Comment savoir si cette hypothèse est bonne ?

En particulier, cette question est importante dans un contexte de traitement d'une scène 3D où le niveau de bruit n'est ni connu, ni uniforme dans la scène. Cette question générale a été peu traitée en vision.

Le contrôle le plus simple consiste à vérifier si les paramètres du modèle sont corrects (indépendamment des points), c'est-à-dire compatibles avec l'information que l'on a sur l'environnement.

Ensuite, si jamais on connaît le niveau de bruit local (par exemple si on l'a estimé sur un voisinage proche), on procède à un simple contrôle de l'erreur d'ajustement.

Ce contrôle n'est pas toujours suffisant ni toujours possible. La figure 5.9 décrit une telle situation : la scène «SynthCyl2» contient des points bruités se trouvant sur un cylindre, et la scène «SynthCone1»

FIG. 5.9 – Scènes «SynthCone1» et «SynthCyl2»

FIG. 5.10 – Ajustement de cylindre sur les scènes «SynthCone1» et «SynthCyl2»

contient des points moins bruités se trouvant sur un cône [2]. Les deux scènes contiennent le même nombre de points (1332). Si on choisit d'ajuster un cylindre sur chacune des deux scènes (Figure 5.10), on obtient les valeurs suivantes pour la quantité e' :

	SynthCyl2	SynthCone1
N	1332	1332
e'	5.59	4.29

On voit donc que l'erreur d'ajustement e' n'est pas adaptée ici pour faire la distinction entre le cas où le modèle est satisfaisant (scène «SynthCyl2») et le cas où il ne l'est pas (scène «SynthCone1»). Aussi est-on amené à analyser plus finement les écarts entre les points et le modèle.

Afin de préciser le problème à résoudre, considérons un modèle m, et des données $\mathbf{x_i}(1 \leq i \leq N)$. Les résidus sont les écarts $d_i = d(\mathbf{x_i}; m)$ entre les points et le modèle. De plus, le modèle m étant une surface, on peut repérer tout point $\mathbf{x_i}$ par deux paramètres de position α_i et β_i. Par exemple, un cylindre est paramétré par les coordonnées cylindriques (z, ψ) qui lui sont associées.

Quand peut-on dire que le modèle m est valide au vu des données $\mathbf{x_i}$? L'idée intuitive que nous avons suivie est de considérer que si m est un modèle acceptable, alors les résidus ont l'aspect d'un bruit pur. Ceci se manifeste en particulier par le fait que les d_i ne doivent pas dépendre de la position (α_i, β_i) du point $\mathbf{x_i}$ sur la surface. Autrement dit, les résidus ne doivent pas indiquer une tendance. Cette situation est symbolisée figure 5.11. La question posée ici est donc de savoir si l'ensemble des

[2] Ces deux scènes ont été générées par un simulateur de scanner, «simulaser», développé au Centre de Robotique.

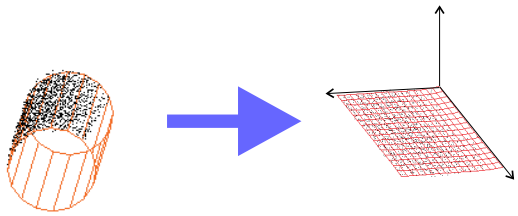


FIG. 5.11 – Résidus d_i tracés en fonction des paramètres de position (α_i, β_i)

résidus forme une tendance spatiale ou non (ceci au vu des données disponibles $\mathbf{x_i}(1 \leq i \leq N)$ bien entendu).

Cette partie est structurée comme suit. La section 5.2.1 présente la méthode que nous utilisons pour simplement invalider un modèle par rapport à la valeur de ses paramètres. La section 5.2.2 traite des procédés d'invalidation à partir de l'erreur d'ajustement lorsque le niveau de bruit sur les points est connu. Dans les sections suivantes, on suppose que le niveau de bruit n'est pas connu. La section 5.2.3 présente des techniques existantes pour examiner la loi statistique des résidus, et en particulier pour tester si les résidus suivent une loi normale. La section 5.2.4 décrit une classe de méthodes d'analyse des résidus à partir de séquences binaires existant dans les cas 2D et 2.5D, et comment ces méthodes peuvent s'adapter à notre cas. Les sections 5.2.5 et 5.2.6 présentent des voies que nous avons explorées pour résoudre le problème. La section 5.2.5 présente une approche originale qui fait intervenir une analyse de régression linéaire des résidus en fonction des paramètres de position et un test de sélection de modèle. Les limitations de cette méthode ont conduit à développer les techniques présentées au 5.2.6. L'approche présentée dans la section 5.2.6 est issue de développements récents visant à fournir une description non-paramétrique des résidus en fonction des paramètres de position (par lissage dépendant du modèle).

5.2.1 Vérification des paramètres

Ceci consiste simplement à vérifier si les paramètres de la primitive trouvée sont compatibles avec certaines limites, spécifiées par une certaine connaissance de l'environnement (voir connaissances métier dans le chapitre 3).

Certaines contraintes sont simples à spécifier. Par exemple, on impose pour le tore que son «grand rayon» R soit supérieur ou égal à son «petit rayon» r. Ceci est une contrainte naturelle car nos environnements ne contiennent que des tores en forme de «pneu» (pas d'auto-intersections). Cette contrainte est importante car elle supprime le lien de dégénérescence entre tore et sphère (lorsque le grand rayon est nul).

D'autres contraintes sont plus difficiles à définir, comme par exemple le rayon maximum d'un cylindre, ou d'une sphère dans ce type de scène. En pratique, ces informations sont demandées à l'utilisateur (Figure 5.12).

On peut ainsi invalider certains ajustements, en particulier de cas dégénérés (cône dont l'angle est très faible, etc.).

Notons cependant que ceci ne suffit pas à éliminer tous les cas de sur-ajustement. Ceci est cohérent, d'autant plus qu'il n'est pas possible d'imposer des limites très contraignantes sur certains paramètres. Par exemple, la gamme de diamètres possibles pour les lignes de tuyauterie est très vaste.

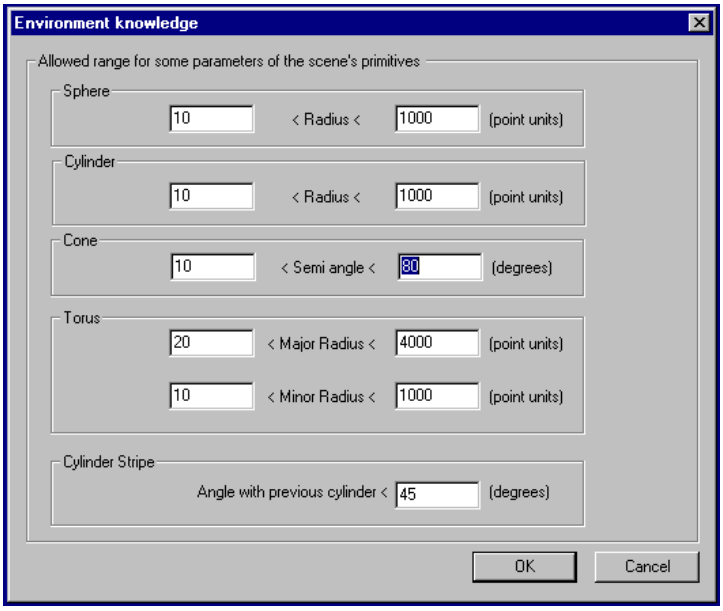


FIG. 5.12 – Dialogue de l'application pour spécifier les valeurs limites de certains paramètres

5.2.2 Test si le niveau de bruit est connu

L'idée est tout simplement de tester si l'écart-type empirique des d_i est inférieur à un certain σ_0, supposé connu (soit spécifié par l'utilisateur, soit estimé par ailleurs). Ceci revient simplement à comparer la quantité e' définie par l'équation 5.1 à la valeur σ_0. Etant donné que e'/σ_0 est censé suivre une loi du χ^2, il est même possible de rendre une probabilité d'accepter ou de rejeter le modèle (voir le concept de p-valeur au 5.2.5).

> Remarque
> En pratique, dans le meilleur des cas, si on connaît un niveau de bruit, c'est un bruit de saisie, et non un bruit par rapport aux modèles. Or ces deux notions sont différentes. Ceci étant dit, en connaissant la matrice de variance-covariance 3×3 du bruit sur chaque point, on peut en déduire une estimation du niveau de bruit σ_i au point considéré selon la normale à la primitive. Notons que même si le bruit de saisie est uniforme (et en faisant abstraction du bruit de recalage), le bruit des points par rapport au modèle ne l'est plus. On est donc amené à réaliser l'ajustement de primitive en considérant la fonction coût :
>
> $$\sum_i \frac{d_i^2}{\sigma_i^2}$$

Ceci dit, on peut voir que cette analyse ne suffit pas, en particulier dans notre application, où le bruit sur les points n'est ni connu, ni uniforme dans la scène. C'est justement le cas illustré figure 5.10.

5.2.3 Loi statistique des résidus

Principe

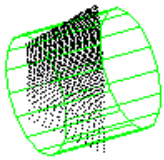
L'idée est ici de considérer que si la primitive est un modèle acceptable, la distribution des résidus suit à peu près la loi statistique du bruit sur les points. En particulier, si le modèle est complètement

faux, on peut s'attendre à ce que la distribution des écarts soit très différente de la loi du bruit sur les points.

On suppose généralement que le bruit sur les points est issu d'une variable aléatoire gaussienne, ce qui est fait implicitement lorsque l'on utilise une méthode de moindres carrés pour l'ajustement de la primitive [3] .

Dans notre contexte, il paraît réaliste de supposer que l'hypothèse de normalité des résidus est valable, au moins localement dans la scène complète. Ceci signifie que sur un voisinage, les résidus peuvent être considérés comme des réalisations d'une variable aléatoire gaussienne $\mathcal{N}(\mu, \sigma^2 I)$.

Sur les figures 5.13 et 5.14, on voit clairement une différence entre les scènes «SynthCyl2» et «SynthCone1» lorsque l'on compare l'histogramme des résidus avec la densité de la loi normale : pour la scène «SynthCone1», l'histogramme est éloigné de la forme de la loi normale correspondante ; pour la scène «SynthCyl2», l'histogramme est beaucoup plus proche de la densité de la loi normale. Notons que ceci concerne le profil de la loi, et non le niveau de bruit (les paramètres de la loi normale ont été estimés à partir des données dans les deux cas).



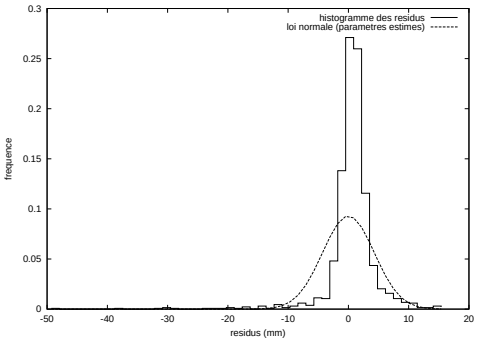


FIG. 5.13 – Histogramme des résidus et loi gaussienne (avec paramètres estimés) dans le cas d'un ajustement de cylindre sur la scène «SynthCone1».

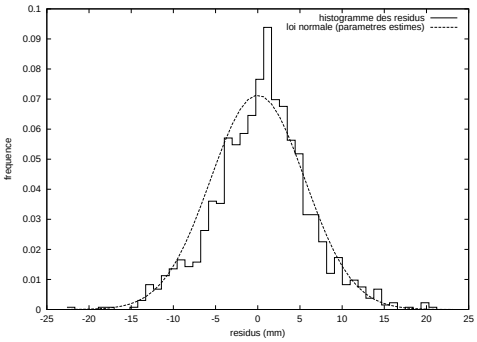


FIG. 5.14 – Histogramme des résidus et loi gaussienne (avec paramètres estimés) dans le cas d'un ajustement de cylindre sur la scène «SynthCyl2».

[3] Dans le cas d'erreurs gaussiennes, la méthode des moindres carrés correspond à l'estimateur du maximum de vraisemblance, ce qui fournit une certaine justification de l'utilisation de cette méthode.

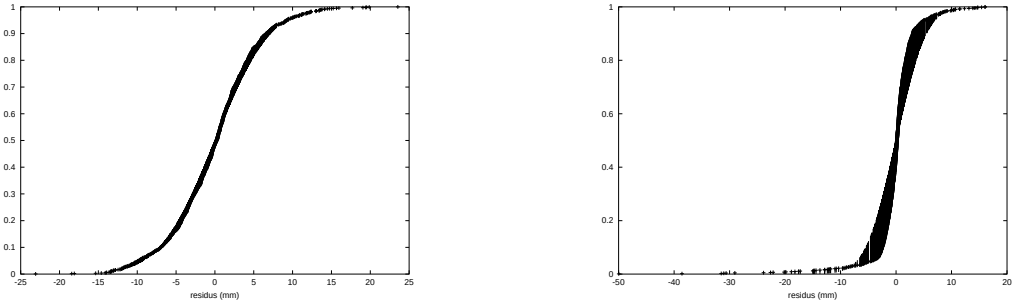


FIG. 5.15 – Ecarts entre fonctions de répartition empirique et de la loi normale (avec paramètres estimés) pour les scènes «SynthCyl2»(à gauche) et «SynthCone1» (à droite)

Pour tester la normalité des résidus, ou l'adéquation à un autre type de loi (exponentielle, lorentzienne, ...), il existe des tests basés sur la fonction de répartition («*edf (empirical distribution function) based tests*»), qui présentent l'avantage de ne pas nécessiter de création d'histogramme discrétisé. En effet, la fonction de répartition empirique est naturellement discrétisée par chaque intervalle $[d_i, d_{i+1}]$, où les d_i sont les résidus classés par ordre croissant (Figure 5.15).

Tests à partir de la fonction de répartition

Soit $X_1, \cdots, X_n$, n variables aléatoires indépendantes identiquement distribuées de loi μ. Etant donné une loi connue μ_0, un test d'adéquation consiste à examiner si :

$$\mu = \mu_0 \text{ (hypothèse } H_0)$$

Si on note F la fonction de répartition de la loi μ, et F_0 celle de μ_0, le test ci dessus revient à examiner si :

$$F = F_0 \text{ (hypothèse } H_0)$$

Ceci conduit aux tests basés sur la fonction de répartition empirique. Deux cas peuvent se présenter :

- La fonction de répartition F_0 est complètement déterminée. C'est un problème d'adéquation à une loi donnée.
- La fonction de répartition F_0 n'est pas complètement déterminée (i.e. ses paramètres ne sont pas connus). C'est un problème d'adéquation à une famille de lois.

Le premier cas est le plus simple, et il existe de nombreux tests possibles (chi-deux, Kolmogorov-Smirnov, etc.). On peut en particulier obtenir des tests qui ne dépendent pas du type de la loi μ_0. Le second cas est plus compliqué, et des tests spécifiques doivent être mis en œuvre pour chaque type de loi (gaussienne, exponentielle, etc.)

En pratique, pour estimer la fonction de répartition F, on construit la fonction de répartition empirique de l'échantillon $X_1, \cdots, X_n$:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I(X_i \leq x)$$

où $I(X \leq x) = 1_{]-\infty,x]}(X)$. L'une des manières de mesurer l'écart entre les deux fonctions de répartition est de considérer la statistique de Kolmogorov :

$$D_n = \sup_{x \in \mathbb{R}} |F_n(x) - F(x)|$$

En pratique, on considère :

$$D_n = \max_{1 \leq i \leq N} |F_n(X_i) - F(X_i)| \tag{5.3}$$

Un résultat intéressant est que $\sqrt{n}D_n$ tend, lorsque n tend vers $+\infty$, vers une variable aléatoire de loi Q déterminée (dite loi de Kolmogorov-Smirnov), qui est en outre indépendante de la loi μ, i.e. de F. La fonction de répartition de cette loi s'écrit [Die97] :

$$P(X > a) = 2 \sum_{k=1}^{\infty} (-1)^{k+1} e^{-2k^2 a^2} \tag{5.4}$$

Ce résultat permet notamment de former le test de Kolmogorov-Smirnov d'adéquation à une loi déterminée quelconque, présenté dans la suite. On utilise aussi la statistique D_n pour tester l'adéquation à une famille de lois, mais dans ce cas la loi limite de $\sqrt{n}D_n$ dépend de la loi μ. Ceci donne lieu à plusieurs tests (également nommés de Kolmogorov-Smirnov) d'adéquation à une loi de paramètres inconnus (estimés), spécifiques pour chaque type de loi. Le cas particulier que nous avons utilisé, à savoir celui de la loi gaussienne, est présenté ici.

a. Test de Kolmogorov-Smirnov d'adéquation à une loi quelconque complètement déterminée Asymptotiquement, on aurait :

$$P(\sqrt{n}D_n > k_\alpha) = \alpha$$

où k_α est issu d'une table de loi de Kolmogorov-Smirnov. En pratique, lorsque n est petit, des corrections sont apportées. Ces corrections, qui deviennent négligeables à partir de $n > 20$, sont issues de simulations de Monte-Carlo (cf. tabulations de Stephens [DS86]). La version modifiée s'écrit :

$$P((\sqrt{n} + 0.12 + 0.11/\sqrt{n})D_n > k_\alpha) = \alpha$$

Le tableau suivant donne les niveaux de signification asymptotiques :

α	.25	.15	.10	.05	.025	.01	.005	.001
k_α	1.019	1.138	1.224	1.358	1.480	1.628	1.731	1.950

Ces valeurs sont simplement issues de l'équation 5.4. Au lieu d'utiliser des tables, on peut ré-estimer ces valeurs directement à partir de la formule 5.4 (cf. fonction `probks` de [PTVF92b]).

b. Tests de Kolmogorov-Smirnov d'adéquation à une loi de paramètres inconnus Ces tests, encore appelés tests de Kolmogorov-Smirnov d'adéquation à une famille de lois, sont spécifiques à chaque loi. En particulier, les niveaux de signification mais aussi les corrections pour les faibles valeurs de n sont particuliers à chaque loi [DS86]. Nous donnons ici simplement le cas de la loi normale.

Loi normale $\mathcal{N}(m, \sigma^2)$(m et σ^2 inconnus). On estime m et σ^2 par :

$$m = \bar{X}$$

et

$$\sigma^2 = \frac{1}{n-1} \sum (X_i - \bar{X})^2$$

Ici encore, on effectue des corrections sur la statistique pour les faibles valeurs de n. La version modifiée s'écrit :

$$P((\sqrt{n}-0.01+0.85/\sqrt{n})D_n > k_\alpha) = \alpha$$

Le tableau suivant donne les niveaux de signification asymptotiques de la loi obtenue (qui n'est plus la loi de Kolmogorov-Smirnov) :

α	.15	.10	.05	.025	.01
k'_α	0.775	0.819	0.895	0.995	1.035

Ces valeurs sont utilisées pour analyser les résultats à la fin de cette partie.

Résultats et conclusion

Résultats Le tableau suivant montre la valeur de la statistique $\sqrt{n}D_n$ pour les résidus issus de l'ajustement de cylindre sur les scènes «SynthCyl2» et «SynthCone1» avec moyenne et variance estimées.

Scène	$\sqrt{n}D_n$
SynthCyl2	0,034
SynthCone1	0,184

On remarque tout d'abord que les deux valeurs sont cohérentes entre elles, dans la mesure où plus l'ajustement est bon, plus la valeur de la statistique $\sqrt{n}D_n$ est faible. Cependant, si l'on se reporte au tableau donnant les valeurs de k'_α, toutes les valeurs de seuil sont supérieures aux valeurs trouvées expérimentalement. Ceci signifie que, pour des valeurs de α usuelles, l'hypothèse de normalité des résidus n'est pas rejetée. Autrement dit, les résidus ne sont pas considérés comme incompatibles avec une hypothèse de loi normale.

En outre, cette méthode semble faire très peu la différence entre un modèle complètement mauvais et une erreur due au recalage, par exemple. En effet, sur la scène de la figure 5.16, les valeurs de la

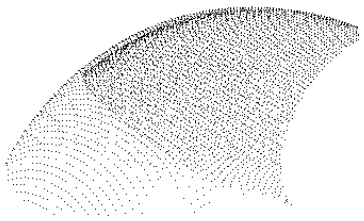


FIG. 5.16 – Scène présentant une (légère) erreur de recalage.

statistique $\sqrt{n}D_n$ pour l'ajustement des différentes primitives sont reportées dans le tableau suivant :

	plan	sphère	cyl	cône	tore
$\sqrt{n}D_n$	0.122	0.090	0.129	0.104	0.067

Tous les modèles à part le tore sont clairement mauvais (aucun modèle ne dégénère en tore !). On voit ici que les valeurs du tore sont peu séparées des autres valeurs, de celle de la sphère par exemple.

Le constat que nous faisons sur ces exemples a été également observé sur d'autres exemples, en particulier sur les séquences de cylindres de l'algorithme d'extraction de lignes de tuyauterie de la section 3.2. Dans la plupart des cas, le test n'est pas sélectif, au sens où l'hypothèse de loi normale n'est jamais rejetée (aux niveaux de signification usuels).

Conclusion Les résultats expérimentaux montrent que le test de normalité des résidus ne conduit pas à une validation véritablement sélective. Ceci peut s'expliquer en partie par le fait que la loi des résidus ne tient pas compte de la position des points sur la surface. Aussi une mauvais modèle peut-il conduire, en projection, à une loi qui n'est pas incompatible avec une loi normale. Pour cette raison, nous nous sommes orienté vers des méthodes visant à analyser le lien entre les résidus et les positions des points sur la surface.

5.2.4 Méthodes à partir de séquences binaires

Dans le domaine de la vision, les quelques travaux que nous connaissons où il est question de valider un modèle font intervenir des séquences binaires. Ces travaux concernent le cas de courbes 2D [Fit97] et des surfaces 2.5D [BJ88], et rejoignent certains thèmes de statistiques (tests sur les séquences). Il paraissait intéressant d'étudier ces méthodes et de voir comment celles-ci peuvent s'étendre à des nuages de points 3D.

Principe

L'approche consiste à remplacer les d_i par des valeurs binaires δ_i suivant que d_i est inférieur (-) ou supérieur(+) à une constante D. La constante D est généralement fixée à la valeur 0, mais peut également être fixée à la médiane des valeurs d_i. L'idée sous-jacente est de considérer qu'il est plus simple de repérer des tendances au sein des valeurs binaires qu'avec les valeurs de départ. Un autre argument est le fait que les valeurs binaires ne dépendent plus du niveau de bruit sur les d_i. Une «séquence»(*run*) désigne ici un ensemble de signes δ_i consécutifs égaux (Figure 5.17). Les méthodes

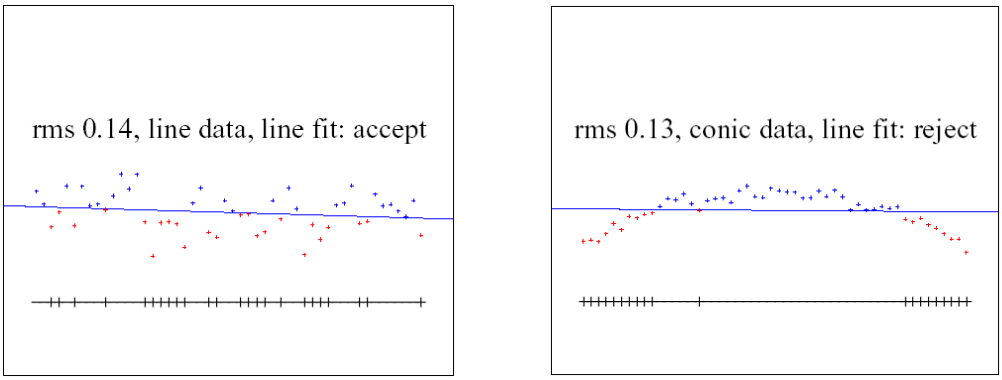


FIG. 5.17 – Méthode de validation à partir de séquences binaires en 2D [Fit97]

existantes traitent essentiellement du cas de courbes 2D. Ainsi, nous présentons d'abord ce cas, puis nous examinons comment ces méthodes peuvent être étendues au cas 3D.

Cas où la primitive est une courbe de $\mathbb{R}^2$

Ici un seul paramètre α_i est nécessaire pour définir la position du point $\mathbf{x_i}$ sur la primitive géométrique. On cherche alors à déterminer si les variables b et α sont stochastiquement indépendantes au vu de l'échantillon des N données (α_i, d_i).

On classe tout d'abord les δ_i suivant l'ordre des α_i (i.e. le long de la courbe).

Intuitivement, si les δ_i sont effectivement indépendants des α_i, il doit y avoir à peu près autant de + que de -, et les séquences de signes identiques consécutifs doivent être assez courtes.

$$+-+-++-+-+-+-\underbrace{++}-++-++-+-+-+$$

A l'inverse, lorsque les δ_i dépendent des α_i, on s'attend à ce que les séquences soient plus longues.

$$-------\underbrace{+++++++++++++}-------$$

Autrement dit, des séquences de signes consécutifs identiques très longues, qui traduisent une «tendance» plus que du bruit pur, doivent mettre en doute la validité du modèle géométrique. Les tests qui suivent permettent de quantifier cette idée.

Test sur la longueur maximum des séquences Il existe des tests portant sur la longueur de la séquence la plus longue de signes consécutifs identiques. Si cette longueur est plus grande qu'un certain seuil fois le nombre de points, alors l'hypothèse est rejetée [CLJ67, p.403], [BF81, BJ88]. Ce test semble assez sensible au bruit [Fit97].

Test sur le nombre de séquences Le nombre de séquences de signes consécutifs identiques semble être un bon indicateur pour déterminer si les δ_i sont indépendants des α_i.

Notons n_1 le nombre de '+', n_2 le nombre de '-', et s le nombre de séquences de signes identiques consécutifs dans la séquence globale. Etant donné n_1 et n_2 fixés, un inventaire des différents arrangements permet de déterminer les fréquences d'apparition des valeurs de s, qui donne la fonction de répartition de s pour n_1 et n_2 fixés. Pour $n_1 \leq 10$ et $n_2 \leq 10$, on trouve ces fréquences dans une table [DS66, p. 98]. Lorsque n_1 et n_2 dépassent la valeur de 10, on considère que leur loi est approchée convenablement par une loi normale, dont la moyenne μ et la variance σ^2 sont[4] :

$$\mu = \frac{2n_1n_2}{n_1+n_2} + 1$$

$$\sigma^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1+n_2)^2(n_1+n_2-1)}$$

Par conséquent, $S = \frac{s-\mu+\frac{1}{2}}{\sigma}$ suit approximativement une loi $\mathcal{N}(0,1)$ [DS66, pp.97]. Ainsi, le test final consiste à vérifier si la valeur empirique de s est comprise ou non dans l'intervalle de confiance $[-k_\alpha, +k_\alpha]$, où k_α est issu d'une table de loi normale $\mathcal{N}(0,1)$.

Test sur la loi de distribution des longueurs de séquences Le test construit dans [Fit97] s'intéresse aux longueurs de toutes les séquences considérées comme observations d'une variable aléatoire. L'histogramme des longueurs est ensuite comparé à la densité des longueurs dans le cas où l'ajustement est bon. La densité des longueurs est estimée par des simulations de Monte-Carlo, et ceci pour chaque type de primitive (droite, ellipse, cercle).

[4] On peut montrer que μ et σ^2 sont la moyenne et la variance de la véritable fonction de répartition discrète de s.

Extension au cas d'une surface de $\mathbb{R}^3$

Comment peut-on étendre ce type de méthode au cas d'une surface ? Comme on peut le voir à la partie précédente, les tests pour les courbes exploitent le fait que le paramètre de position est un scalaire, et donc qu'une relation d'ordre existe sur l'espace de ce paramètre. En ce qui concerne les surfaces, le problème est plus compliqué, car le vecteur paramètre possède deux coordonnées, et interdit donc l'usage de statistiques d'ordre. Que peut-on faire tout de même ?

Séquences de points projetés le long d'une courbe Afin d'utiliser les résultats existants sur les courbes, il semblait a priori intéressant de projeter les points sur une courbe. Cette courbe est alors utilisée pour définir l'ordre dans lequel classer les résidus. On peut alors utiliser tous les tests sur les courbes présentés ci dessus.

Les courbes les plus simples sont les droites correspondant à β=cste ou α=cste. Ceci revient à considérer les répartitions marginales des points (α_i, β_i, d_i). Autrement dit, on projette les points (α_i, β_i, d_i) successivement sur le plan (α_i, d_i) et sur le plan (β_i, d_i). Si l'on repère un caractère non-aléatoire dans l'une des directions u ou v, le modèle est à invalider. Bien entendu, il est également possible d'utiliser toute autre courbe.

Pour le cas du cylindre, on utilise z et ψ des coordonnées cylindriques dans un repère lié au cylindre. En introduisant par exemple la variable $w = z + a\psi$, on forme des séquences binaires le long d'une hélice enroulée autour du cylindre. Pour une hélice donnée (définie ici de façon à effectuer 4 tours autour du cylindre), on obtient les valeurs S suivantes :

Scène	S
SynthCyl2	0,5
SynthCone1	309
angle	123

Dans cet exemple, on obtient donc bien un résultat cohérent. Cependant, le résultat est très sensible au choix de la courbe ou de l'ordre selon lequel classer les résidus. Or, il est très délicat de choisir une courbe ou un ordre fixe qui s'adapte à toutes les situations.

Les résultats obtenus au sein de l'algorithme d'extraction de tuyauterie (section 3.2) ont montré que ces informations étaient peu pertinentes telles quelles. Les valeurs obtenues n'indiquent pas de grande différences entre les cas où l'ajustement de surface est bon des cas où l'ajustement est mauvais. La raison est la suivante : sauf dans certains cas, les répartitions marginales (sur les bords ou le long d'une courbe) sont en général assez faiblement représentatives de la tendance 3D de la surface, la projection ayant pour effet d'annuler les tendances spatiales. Une solution consisterait à essayer un grand nombre de courbes, ou encore d'ordres aléatoires et les combiner en faisant une moyenne.

Tests d'indépendance des d_i et des $(\alpha_i, \beta_i)^T$ L'extension naturelle des tests sur les séquences vus plus haut serait de construire des séquences 2D. Pour ce faire, on peut envisager de construire une triangulation de Delaunay 2D des points (α_i, β_i), puis de repérer sur la triangulation les composantes connexes composées de sommets ayant des signes identiques (équivalent 2D des séquences introduites dans le cas 1D).

Une alternative (plus simple) à la construction de maillage consisterait à considérer un quadrillage régulier, dans chaque case duquel serait calculée la moyenne des signes des points présents (s'il y en a). On raisonnerait alors sur les cases, en regardant les composantes connexes (4-ou 8-connexité). Le problème évident concerne la taille des cases, qui introduit un paramètre supplémentaire.

On peut alors évaluer la taille de la plus grande composante connexe, ou même le nombre de telles composantes, et leurs tailles respectives. Il resterait cependant à déterminer comment exploiter ces données, et notamment comment construire des tests similaires au cas 1D pour le cas 2D.

Dans [BJ88], une méthode similaire est utilisée pour étudier localement une surface du type $z = f(x,y)$ sur une image de profondeur. La grille de l'image 2.5D fournit dans ce cas un élément naturel pour définir les séquences. L'approche suivie consiste ensuite à détecter les tendances par des méthodes de morphologie mathématique sur les composantes connexes.

Limitation de ces approches

Dans les algorithmes mettant en œuvre les séquences binaires que nous venons de présenter, il n'y a pas de différence entre un petit écart et un gros écart. L'avantage est une grande robustesse aux points aberrants [FEF97]. L'inconvénient est que l'on ne fait pas toujours la différence entre des tendances et des écarts systématiques mais à peu près constants. Or dans notre cas, il peut y avoir des problèmes de recalage tels que les séquences de signes indiquent une tendance, alors qu'il n'y a pas de tendance réelle (voir l'exemple de la figure 1.12, page 14).

Conclusion

Nous n'avons pas implémenté cette approche dans le cas 3D, en partie à cause des limitations propres à ces méthodes, et à la difficulté de les adapter au cas de nuages de points 3D.

Il serait néanmoins intéressant d'étudier l'extension de la méthode pour les courbes utilisant le nombre de longueurs des séquences. Toutefois, le problème de trouver la loi du nombre de composantes connexes de signes identiques sur un maillage 3D ne semble pas avoir de solution évidente.

5.2.5 Approche par régression linéaire

Idée générale

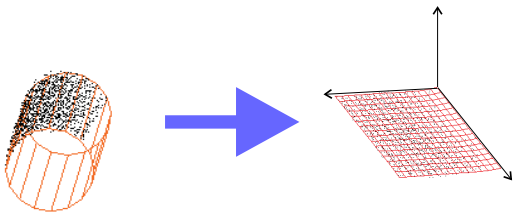


FIG. 5.18 – Principe de la méthode

Le principe de la méthode développée ici est de «déplier» la surface, c'est-à-dire de considérer les nouveaux points de $\mathbb{R}^3$ (Figure 5.18) :

$$\mathbf{y_i} \begin{pmatrix} \alpha_i \\ \beta_i \\ d_i \end{pmatrix}$$

et d'étudier la surface que forment les $\mathbf{y_i}$. Si les points $\mathbf{x_i}$ ne sont pas bruités et que l'ajustement du modèle m est parfait, les points $\mathbf{y_i}$ se trouvent exactement sur le plan $d = 0$.

En particulier, si l'on ajuste un modèle du type $d = P(\alpha, \beta)$ où P est un polynôme, les coefficients du polynôme P sont tous nuls (dans le cas idéal que nous venons de décrire). Dans le cas où les points de départ sont bruités, les coefficients trouvés ne sont plus tout à fait nuls, mais on peut tester l'hypothèse de cette nullité du point de vue statistique.

L'avantage est ici que l'ajustement d'une telle surface entre dans le cadre d'une régression linéaire (même si le polynôme P n'est pas linéaire), et donc que l'on peut utiliser les outils classiques de régression et d'analyse de variance pour le test de nullité des coefficients.

Etude de modèles emboîtés en régression linéaire

Le modèle de régression linéaire classique (non-contraint) s'écrit

$$Y = Xb + u, \text{ avec } u \sim \mathcal{N}(0, \sigma^2 I_N)$$

où $Y \in \mathbb{R}^N$, $X \in \mathcal{M}_{N,p}(\mathbb{R})$, $b \in \mathbb{R}^p$ et $u \in \mathbb{R}^N$. Comme cela a été dit au 5.1, la solution du problème de moindres carrés s'écrit :

$$\hat{b} = (X^T X)^{-1} X^T Y$$

On considère maintenant une contrainte affine sur les coefficients de b :

$$Cb = c$$

où $C \in \mathcal{M}_{r,p}(\mathbb{R})$, $b \in \mathbb{R}^p$, et $c \in \mathbb{R}^r$ avec $0 < r < p$, et où C est de plein rang lignes. Ce type de contrainte est intéressante en ce qu'elle fournit un cadre assez général pour définir des modèles emboîtés. La solution du problème de moindres carrés contraint par la condition affine sur b s'écrit [Del01, p.42] :

$$\hat{b_0} = \hat{b} + (X^T X)^{-1} C^T [C(X^T X)^{-1} C^T]^{-1} (c - C\hat{b})$$

Dans notre cas, on a :

$$X = \begin{pmatrix} \mathbf{f}(\alpha_1, \beta_1)^T \\ \cdots \\ \mathbf{f}(\alpha_N, \beta_N)^T \end{pmatrix}$$

et

$$Y = \begin{pmatrix} d_1 \\ \cdots \\ d_N \end{pmatrix}$$

La régression consiste ensuite à calculer la surface $d = f(\alpha, \beta)$ polynomiale de moindres carrés (annexe C).

Test sur les coefficients estimés

Le test que l'on souhaite effectuer sur les coefficients du polynôme obtenu est essentiellement un test de nullité de tout ou partie des coefficients. On souhaite donc tester l'hypothèse $H_0 : Cb = c$ contre l'hypothèse alternative $H_a : Cb \neq c$ [ER93, p.132,136]. Sous l'hypothèse H_0, la statistique F de Fisher suit une loi connue :

$$F = \frac{\|\hat{Y} - \hat{Y_0}\|^2}{r\hat{\sigma}^2} \sim F_{r,N-p}$$

où $\hat{Y} = X\hat{b}$ est la prédiction sous l'hypothèse H_a(modèle non-contraint), $\hat{Y_0} = X\hat{b_0}$ la prédiction sous l'hypothèse H_0 (modèle contraint), $\hat{\sigma}^2$ est l'estimation de la variance σ^2 sous H_a, $F_{.,.}$ est la loi de Fisher-Snedecor (cf. [ER93, p.134], [Del01, p.43]) et $\|.\|$ désigne la norme euclidienne de $\mathbb{R}^N$.

La statistique F de Fisher s'écrit également (par Pythagore) :

$$F = \frac{\|\hat{Y} - \hat{Y}_0\|^2/r}{\|\hat{u}\|^2/(N-p)} = \frac{(\|\hat{u_0}\|^2 - \|\hat{u}\|^2)/r}{\|\hat{u}\|^2/(N-p)} \tag{5.5}$$

où $\hat{u} = Y - \hat{Y}$ et $\hat{u_0} = Y - \hat{Y}_0$.

> Remarque
> La condition sur r, p, N s'écrit : $0 < r < p < N$. En effet, si $N = p$, on ne peut pas tester la qualité de l'ajustement puisque le modèle peut parfaitement ajuster les données.

Application au test de nullité de coefficients

Supposons que l'on souhaite tester la nullité de r coefficients b_i, i appartenant à un sous-ensemble I de $\{1, \ldots, p\}$. On prend alors comme matrice C la matrice obtenue en retranchant à la matrice identité I_p les lignes ne correspondant pas aux éléments i de I, et comme vecteur c le vecteur nul de $\mathbb{R}^r$.

Pour nos applications, si l'on approche Y par une fonction polynomiale, on veut tester si tous les coefficients sauf le terme constant sont nuls. Dans ce cas, $r = p - 1$.

Le concept de p-valeur

Au lieu de donner un niveau de tests α et de rendre une valeur booléenne, lorsque l'on connaît la loi de la statistique, on peut rendre la p-valeur : c'est le niveau à partir duquel la décision change. Plus précisément, la p-valeur du test est la probabilité, si H_0 est la bonne hypothèse, d'avoir observé une valeur pour X qui dépasse le x observé. Au niveau décisionnel, on rejette donc H_0 lorsque la p-valeur est faible [Car00, p.28]. Le calcul de la p-valeur est le suivant :

$$Q(x) = P(X \geq x) = 1 - F(x)$$

où F est la fonction de répartition de la variable X.

En l'occurrence, la fonction de répartition d'une loi de Fisher-Snedecor s'écrit :

$$F_{n_1,n_2}(x) = 1 - I_{\frac{n_2}{n_2+n_1 x}}(n_2/2, n_1/2)$$

définie pour $n_1, n_2 > 0$ et $x \geq 0$, où $I_.(.,.)$ est la fonction beta incomplète (cf. fonction `betai` dans [PTVF92b, §6.4.11, p.229]).

$$I_x(a,b) = \frac{B_x(a,b)}{B(a,b)} = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$$

(pour $a, b < 0$ et $0 \leq x \leq 1$) où

$$B(z,w) = \int_0^1 t^{z-1}(1-t)^{w-1} dt = \frac{\Gamma(z)\gamma(w)}{\Gamma(z+w)}$$

où

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$$

En pratique, les fonctions `betai`, `betacf`, `gammaln` de [PTVF92b] sont utilisées.

Retour à notre cas

Lorsque l'on veut tester s'il y a une «tendance», c'est-à-dire une dépendance entre d et (α,β), on prend pour hypothèses :

H_0 **:** surface $d = f(\alpha,\beta)$ polynomiale où tous les termes sont nuls sauf le terme constant. La surface des moindres carrés est donc dans ce cas simplement égale au polynôme constant égal à la moyenne des d_i.

H_a **:** surface $d = f(\alpha,\beta)$ polynomiale non contrainte. On ajuste un polynôme $z = P(x,y)$ sur les points $\mathbf{y_i} = (\alpha,\beta,d)^T$, par une méthode de moindres carrés linéaire (annexe C).

On évalue ensuite :

$$F = \frac{(N-p)}{(p-1)} \frac{\sum_{i=1}^{N}[P(\alpha_i,\beta_i) - P_0(\alpha_i,\beta_i)]^2}{\sum_{i=1}^{N}[d_i - P(\alpha_i,\beta_i)]^2} \tag{5.6}$$

et la p-valeur retournée par le test est égale à :

$$P = 1 - f_{p-1,N-p}(F)$$

où $f_{p-1,N-p}(.)$ est le quantile de la loi de Fisher $F_{p-1,N-p}$. Dans ce test, on choisit $P_0(\alpha_i,\beta_i) = \bar{d}$ (fonction constante égale à la moyenne des résidus).

On peut remarquer que le problème s'exprime ici comme un problème de sélection de modèle (parmi les surfaces de régression), telle que celui traité au § 5.1. Le test présenté ici fait intervenir les mêmes quantités que le test de sélection présenté au 5.1 (ceci est visible dans l'équation 5.5), mais est plus général et présente l'avantage de fournir une valeur de probabilité.

Une question cruciale dans ce test réside dans le choix du polynôme, et en particulier du nombre de paramètres p et de son degré. Or, ceci dépend a priori de la complexité du relief formé par les résidus. Un degré de polynôme donné peut s'avérer suffisant ou même trop élevé avec certaines situations et pas assez dans d'autres cas. La complexité de la surface formée par les résidus dépend à la fois du modèle et du nuage de points. Il est donc délicat de choisir un polynôme adéquat. Ce point est lié à la qualité d'approximation, problème abordé dans les limitations ci dessous.

Résultats

Reprenons l'exemple de la figure 5.10, page 137. Les figures 5.19 et 5.20 montrent respectivement pour la scène «SynthCone1» et la scène «SynthCyl2», les résidus issus de l'ajustement d'un cylindre, et la surface polynomiale ajustée sur les résidus en fonction des paramètres de position (avec un polynôme de degré 4×4).

Les valeurs de F et les P-valeurs correspondantes sont reportées dans le tableau suivant :

	SynthCyl2	SynthCone1
N	1332	1332
F	0.47	308.9
P-valeur	0.948	10^{-30}

Ainsi, sur cet exemple, on voit que le test distingue très clairement la situation où le modèle est satisfaisant de celle ou il ne l'est pas.

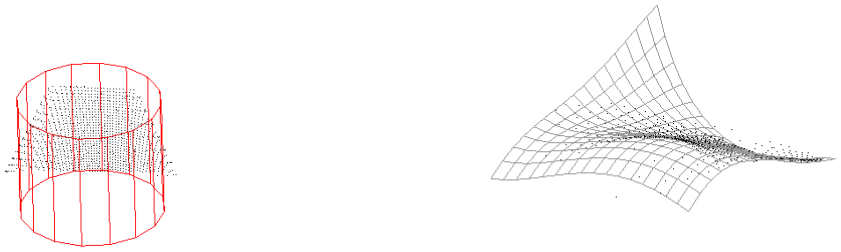


FIG. 5.19 – Ajustement de cylindre sur la scène «SynthCone1» et surface de régression des résidus.

FIG. 5.20 – Ajustement de cylindre sur la scène «SynthCyl2» et surface de régression des résidus.

Limitations

Il se pose un problème d'approximation de la surface de régression. Une surface $z = f(x,y)$ polynomiale ne s'adapte pas à tous les types de reliefs. Par exemple, sur l'exemple de la figure 5.21, le relief est tel que la surface de régression est restée plane. La conséquence est que le dénominateur du ratio est très grand, et donc que la p-valeur rendue est très proche de 1, ce qui indique que le modèle est satisfaisant (ce qui n'est pas le cas !). Il en est de même pour l'exemple des figures 5.22 et 5.23,

FIG. 5.21 – Ajustement de cône sur la scène «angle» et surface de régression des résidus.

présentant une partie de tuyauterie avec une bride (voir chapitre 3). Ces exemples illustrent des situations où le relief des résidus en fonction de la position est trop complexe pour le type de surface polynomiale ajusté et n'est donc pas bien approché. A l'inverse, il existe également des situations de sur-ajustement, où la surface n'approche pas bien les résidus car le degré du polynôme est trop élevé.

Une autre limitation est liée à des problèmes de topologie : la méthode présentée suppose implicitement que l'espace de paramètres construit permet de «déplier» la surface correctement. Or, ce n'est pas tout le temps le cas : par exemple, si le nuage est un ensemble de points répartis tout autour d'une sphère et si l'on ajuste un cylindre sur ce nuage, la projection des points dans l'espace de paramètres

FIG. 5.22 – Scène «bride1» comportant une portion de tuyau avec une bride, vue de face et de côté

FIG. 5.23 – Ajustement de cylindre sur la scène «bride1» et surface de régression des résidus.

ne peut pas être topologiquement correcte (Figure 5.24). En effet, on observe un repliement de surface. Ainsi, on n'obtient pas vraiment un relief puisque plusieurs altitudes sont présentes à la même

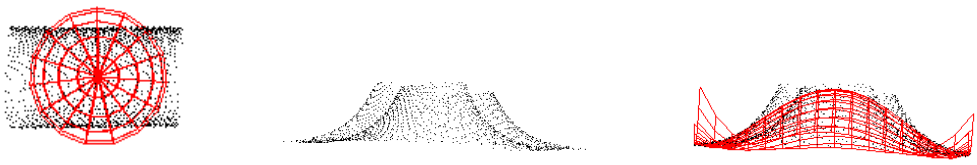


FIG. 5.24 – Problème de topologie. Sphère ajustée sur des points se trouvant sur un cylindre (à gauche), résidus en fonction des paramètres des deux angles des coordonnées sphériques (au centre), surface de régression ajustée sur les résidus dans cet espace (à droite).

position. Remarquons que les méthodes développées dans [BJ88] et [Fit97] ne s'affranchissent pas non plus de ce problème.

Aussi avons-nous tenté de développer une méthode qui s'affranchisse de ces deux limitations : c'est l'objet de la section suivante.

5.2.6 Approche par lissage des résidus à partir du modèle

Cette partie présente une méthode basée sur le même principe que celle du 5.2.5. Le cadre est d'étudier la surface formée par les points $\mathbf{y_i} = (\alpha_i, \beta_i, d_i)^T$ afin d'évaluer la relation entre les résidus d_i et la position (α_i, β_i) des points sur la surface. La méthode développée ici est inspirée de l'approche utilisant la régression linéaire (section 5.2.5), mais donne un résultat correct dans les cas où la première

approche ne fonctionne plus. Ceci mène à un procédé simple et efficace, qui semble prometteur pour les applications du chapitre 3 (Figure 3.33, page 80).

Principe

On a vu au 5.2.5 que les surfaces $d = f(\alpha, \beta)$ polynomiales ne fournissent pas toujours une description satisfaisante du relief des résidus dans l'espace de paramètres de position. Que peut-on faire pour mieux ajuster les résidus, ou plutôt mieux représenter la tendance des résidus dans l'espace des paramètres ?

Une idée simple est de réaliser un lissage des résidus : les résidus lissés suivent bien les formes des résidus de départ et décrivent également la tendance. C'est ce que nous faisons ici. Le principe consiste donc à remplacer le polynôme $P(\alpha_i, \beta_i)$ du 5.2.5 par la valeur :

$$d'_i = \frac{1}{k} \sum_{j/\mathbf{x_j} \in \mathcal{V}_k(\mathbf{x_i})} d_j \tag{5.7}$$

où $\mathcal{V}_k(x)$ est l'ensemble des k plus proches voisins du point $\mathbf{x}$ dans l'ensemble de points de départ $\mathcal{D}$ (Figure 5.25). On calcule alors le ratio ρ (inspiré de la méthode précédente) :

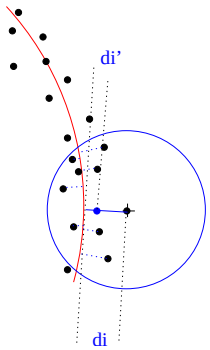


FIG. 5.25 – Principe du lissage à partir du modèle

$$\rho = \frac{\|d' - d_0\|^2}{\|d - d'\|^2}$$

où $\|.\|$ désigne la norme euclidienne de $\mathbb{R}^N$, $d = (d_1, \ldots, d_N)^T$ désigne le vecteur des résidus, $d' = (d'_1, \ldots, d'_N)^T$ désigne le vecteur des résidus lissés, et $d_0 = (0, \ldots, 0)^T$ ou $d_0 = (\bar{d}, \ldots, \bar{d})^T$. En reprenant les notations de la formule 5.6, ρ s'écrit :

$$\rho = \frac{\sum_{i=1}^{N} [d'_i - \bar{d}]^2}{\sum_{i=1}^{N} [d_i - d'_i]^2} \tag{5.8}$$

Il n'est pas possible ici de normaliser le ratio de la même façon que dans le cas de la régression linéaire, étant donné que d' n'a pas d'expression paramétrique en α, β. Cependant, le ratio ρ varie comme le ratio F de la régression linéaire : lorsque l'ajustement est bon, ρ est petit ; lorsque l'ajustement est mauvais, ρ est grand.

Notons que le numérateur et le dénominateur du ratio dépendent tous deux de k, le nombre de voisins. Il est bien sûr possible de remplacer les k plus proches voisins par une boule de taille fixe ; le

paramètre serait alors le rayon de la boule. Plus le nombre k est élevé, et plus les résidus d'_i sont lissés. On en déduit que :

- si k est élevé (proche de N), ρ est proche de 0,
- si k est petit (proche de 1), ρ tend vers l'infini.

Afin de ne pas valider le modèle trop facilement, le nombre de points k a été fixé à une petite valeur en pratique ($k = 10$).

Cette approche conduit à l'algorithme décrit par le pseudo-code suivant :

Pour i= 1 à N faire
  Chercher les k plus proches voisins $i_1, \ldots, i_k$ du point i
  $d'(i) \leftarrow f(d(i_1), \ldots, d(i_k))$
Fin Pour
(Calculer le ratio)

Remarque
L'espace de paramètres de position est finalement utilisé seulement pour la visualisation, et non plus pour la méthode elle-même, qui s'est ainsi affranchie des problèmes de projections et de topologie.

Remarque
Le critère actuel est sensible aux points aberrants. C'est sans doute le prix à payer pour avoir une fonction qui suit de près les données. Par contre, on peut envisager de remplacer la moyenne locale par la médiane locale (ou toute autre fonction des d_i voisins). Un tel choix fournit une surface proche des données mais moins sensible aux points isolés.

Résultats et conclusion

Résultats Le tableau suivant montre les valeurs du ratio ρ pour l'ajustement de cylindre sur les scènes «SynthCone1» et «SynthCyl2» :

	SynthCyl2	SynthCone1
N	1332	1332
ρ	0.35	8.44

Le ratio détecte clairement le cas où il y a un problème du cas où il n'y en a pas.

Pour l'ajustement de cône sur la scène «angle», le relief des résidus bruts d_i et des résidus lissés d'_i est montré figure 5.26 [5]. Pour cette scène, qui pose problème avec la méthode par régression linéaire, on obtient la valeur $\rho = 106,8$, ce qui indique clairement que le modèle n'est pas satisfaisant.

Ceci a été testé dans l'algorithme d'extraction de lignes de tuyauterie (Figure 3.33, page 80). Dans ce contexte, le ratio a semblé prometteur.

[5] les images montrent un maillage destiné à rendre la visualisation plus claire

FIG. 5.26 – Relief des résidus bruts (à gauche), et relief des résidus lissés (à droite) (issus d'un ajustement de cône sur la scène «angle»)

Discussion La question principale qui se pose concernant ce ratio est comment définir un seuil à partir duquel on peut dire que le modèle n'est plus satisfaisant. Il ne semble pas évident de trouver un tel seuil de manière théorique. On peut par contre envisager de trouver une valeur empirique, issue soit de simulations, soit de données réelles. Peut-on, de même que le test de Fisher, construire un test statistique sur ce ratio qui permettrait de fournir une valeur de probabilité ? Là encore, il est possible de réaliser des simulations en vue d'obtenir une loi empirique pour ρ.

D'autre part, si l'on préfère utiliser une quantité qui est bornée, le ratio :

$$\mu = \frac{\|d - d'\|^2}{\|d - d_0\|^2}$$

est compris entre 0 et 1. Ceci découle du fait que d' est toujours un meilleur modèle que d_0 pour approcher les données d.

On constate que la construction des résidus lissés mise en œuvre revient à peu près à construire un maillage surfacique avec pour surface de projection la primitive étudiée. Dans le cas où le maillage est déjà disponible, il est inutile de refaire le calcul des voisins.

Conclusion Nous avons construit ici un ratio obtenu par lissage à partir du modèle. Ce ratio est directement inspiré de la régression linéaire. L'approche par lissage conduit à un procédé simple qui n'est plus limité par des problèmes de topologie ni par les problèmes d'approximation. Cette méthode s'applique à toute surface pour laquelle on peut estimer les résidus (surfaces algébriques, splines, etc.).

En pratique, cette quantité semble permettre de quantifier la qualité d'un ajustement de manière satisfaisante. Il resterait à examiner comment on peut bâtir un test à partir de ce ratio.

5.3 Conclusion

Les deux problèmes de reconnaissance de surface qui sont présentés dans ce chapitre, la sélection de modèle et la validation d'un modèle, sont complémentaires et se sont avérés intimement imbriqués.

La question de la sélection de modèle est un thème très étudié en statistiques et en vision (bien que peu dans le cas de primitives géométriques). Ainsi, il existe un nombre important de critères différents. Le problème principal concerne les cas dégénérés. Mon point de vue sur cette question est que le critère de sélection est assez peu critique, mais qu'il est essentiel de définir précisément les cas que l'on considère comme dégénérés. En pratique, ceci implique de définir des valeurs de seuil. Lorsque les frontières entre les primitives sont claires, la sélection de modèle fonctionne bien, et ce même

avec des critères assez approximatifs. En particulier, la quantité e' (ou même le résidu quadratique moyen e) donne de bons résultats en général. Par contre, sans une telle définition des cas dégénérés, le problème devient mal posé. Ceci montre notamment l'importance de la validation de modèle au sein même du problème de sélection. En d'autres termes, nous proposons pour ce problème d'utiliser des critères de sélection classiques en faisant intervenir des procédés de validation de modèles.

La validation de modèle est un problème qui est paradoxalement très peu étudié. Dans ce chapitre, des techniques originales basées sur l'étude des résidus sont présentées. Les différentes voies explorées ont mené à définir un ratio issu d'un lissage à partir du modèle. Les résultats obtenus en pratique avec ce ratio sont prometteurs (Figure 3.33, page 80).

Chapitre 6

Extraction de primitives géométriques

Ce chapitre traite du problème de l'extraction de primitives géométriques à partir d'un nuage de points. La notion d'extraction diffère de celle de l'ajustement en ce que la scène contient maintenant des points qui n'appartiennent pas au modèle que l'on souhaite extraire. Autrement dit, la scène est composée de points situés sur plusieurs surfaces et/ou de points aberrants. Les méthodes présentées dans ce chapitre sont utilisées pour l'étape initiale de l'extraction de lignes de tuyauterie, section 3.2. De récents développements ont également visé à introduire ces méthodes lors de l'arrêt de l'extraction de lignes de tuyauterie en présence d'ombres (§ 3.2.5, page 72).

Dans ce chapitre, on utilise les termes d'extraction *locale* et d'extraction *globale*. L' extraction *locale* désigne la situation où l'on impose un point précis du nuage, noté $\mathbf{x_0}$ dans ce chapitre, autour duquel chercher une surface. Ce type d'extraction fait intervenir des voisinages autour de ce point. Dans tout ce chapitre, ce que l'on nomme la taille d'un voisinage est un nombre de points (plus proches voisins). Par opposition, l'extraction *globale* désigne le cas où l'on souhaite trouver une ou plusieurs surface(s) dans la scène sans cette donnée de point, c'est-à-dire sans spécifier au départ de points autour desquels ces surfaces se trouvent.

Les méthodes développées dans les sections 6.1, 6.2, 6.3, et 6.4 se fondent sur l'hypothèse suivante concernant la scène :

> H : en tout point où se trouve une surface (point non aberrant), on suppose qu'il existe un voisinage sphérique (non-réduit à un point) autour de ce point où la primitive est la seule présente (i.e. tous les points appartiennent à cette primitive au bruit près).

Une hypothèse à peu près équivalente à celle-ci consiste à dire :

> H' : en tout point non-aberrant, on suppose qu'il existe un voisinage sphérique contenant un nombre de points suffisants pour pouvoir approcher la surface par une primitive de moindres carrés.

En clair, ceci cherche à justifier l'emploi d'une méthode de moindres carrés localement, sur des voisinages plus ou moins petits. Cette hypothèse, illustrée sur la figure 6.1, est vérifiée pour la plupart des points de nos scènes.

Les sections 6.1, 6.2, 6.3, et 6.4 présentent les méthodes développées pour réaliser l'extraction d'une ou plusieurs primitives, localement ou globalement, respectivement de primitive(s) de type fixé ou non, basée sur l'hypothèse H'. Dans les sections 6.1, 6.2 et 6.3, nous considérons le cas le plus simple, qui est celui où les primitives que l'on souhaite extraire sont de type connu, fixé. En pratique, l'utilisateur spécifie explicitement le type de primitive à extraire parmi celles possibles (plan, sphère, cylindre, cône, tore). La section 6.4 présente les perspectives sur l'extraction d'une ou plusieurs primitives de type quelconque, et ouvre sur la segmentation en primitives géométriques.

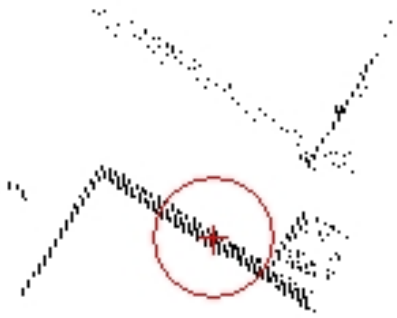


FIG. 6.1 – Hypothèse de validité de la méthode de moindres carrés sur des voisinages sphériques

Dans la section 6.5, nous présentons une méthode pour traiter les cas où l'on ne peut plus faire l'hypothèse H'. Cette méthode se base sur les échantillons minimaux de points. Un échantillon minimal de points est un ensemble de points dont le nombre d'éléments est le minimum nécessaire pour construire une primitive de type donné. Nous avons choisi de désigner un tel ensemble minimal par le terme de *quorum*, par analogie avec son sens commun : nombre de membres qu'une assemblée doit réunir pour pouvoir valablement délibérer (Larousse).

Recherche des k plus proches voisins d'un point dans un nuage

Le principe est le suivant : on considère tout d'abord une liste d'indices de k points initiaux. Cette liste de k indices est d'abord triée par ordre de distance de chaque point au point $\mathbf{x_0}$. Ensuite on procède à une lecture de la liste de points (tout le nuage) avec insertion éventuelle dans la liste triée des k points.

$L = (i_1, \ldots, i_k) \leftarrow$ Indices de k points initiaux
Trier L par ordre des distances des points à $\mathbf{x_0}$ (algorithme *quicksort*)
$\delta 2 \leftarrow \|\mathbf{x}(i_k) - \mathbf{x_0}\|^2$ (carré de la distance du point le plus éloigné)
Pour $i = 1$ à N faire
  Si $\|\mathbf{x}(i) - \mathbf{x_0}\|^2 < \delta 2$ faire
    Insérer i dans la liste triée L (dichotomie)
    $\delta 2 \leftarrow \|\mathbf{x}(i_k) - \mathbf{x_0}\|^2$
  Fin Si
Fin Pour
Retourner la liste des k points

Entrées : nuage de N de points, indice i_0 du point $\mathbf{x_0}$ dans le nuage, nombre k de voisins cherchés ($k \leq N$)
Sorties : Liste L, triée par ordre de distance à $\mathbf{x_0}$.
Paramètres : Choix des k points initiaux.

Il est important de noter que, sans post-traitement, la liste de points rendue en sortie de cet algorithme est triée par ordre de distance de chaque point à $\mathbf{x_0}$.

Comment choisir les k points initiaux ? Si le nuage suit l'ordre du scan, et si l'on connaît l'indice i_0 du point $\mathbf{x_0}$ dans le nuage (ce n'est pas vrai si le point ne fait pas partie du nuage !), on peut tirer parti de la structure en ligne. Il existe cependant des exceptions : fins de ligne, fins de point de vue (ou de sous-point de vue). D'autre part, ceci n'est pas tout le temps possible (par exemple, pour l'estimation des normales sur un voisinage, les points ont déjà pu être retriés). Bien-entendu, on peut dans ce cas former une liste contenant les mêmes indices et qui reproduit le même ordre que la liste du nuage de points. Cependant, ceci nécessite une lecture de toute la liste des N points, avec pour chaque indice un parcours de la liste des k indices.

Croissance de primitive dans un nuage de points

Nous distinguons deux types de croissance :

- Croissance bornée. Le voisinage sur lequel on a déterminé la primitive indique des limites dans l'espace. Par exemple, les points se trouvant sur un cylindre peuvent nous permettre d'estimer les extrémités de ce cylindre. Mais à cause des voisinages sphériques, on n'a pas forcément pris en compte tous les points qui se trouvent sur la primitive entre les limites trouvées. Dans l'exemple du cylindre, on n'a sans doute pas pris en compte les points qui se trouvent sur toute la circonférence de la section circulaire. On effectue donc un rajout de ces points. C'est ce que nous appelons croissance bornée. Les dimensions de la primitive ne sont pas changées par cette opération.
- Croissance non-bornée. Dans certains cas, on veut rajouter tous les points de la scène qui se trouvent sur la primitive (infinie) trouvée. Il s'agit là d'une croissance non-bornée.

Dans les deux cas, on doit définir une épaisseur pour rajouter les points. En pratique, on peut utiliser comme épaisseur une valeur proportionnelle à l'erreur quadratique trouvée lors de l'ajustement de la primitive.

La croissance bornée ne pose pas de problème particulier. Par contre, la croissance non-bornée pose le problème de l'arrêt. Il n'est pas réaliste de considérer que l'on peut toujours rajouter tous les points de la scène à la primitive (par exemple, lorsqu'il y a des murs). Tout dépend de ce que l'on veut obtenir :

- les limites de la primitive sont-elles importantes ?
- est-ce un problème de considérer trop de points dans la primitive ? (points qui ne sont plus considérés par la suite).

Il n'est pas aisé de stopper la croissance d'une primitive afin d'éviter les fausses intersections avec d'autres objets.

6.1 Extraction locale d'une primitive de type fixé

Comme cela a été dit en introduction, l'extraction locale concerne la recherche d'une surface autour d'un point connu $\mathbf{x_0}$. Or, puisque l'on se place dans l'hypothèse H, il existe un voisinage autour du point sur lequel la primitive recherchée est la seule présente.

Plusieurs idées sont alors possibles pour extraire la primitive :

- Si on connaît la taille du voisinage appropriée, ou si une grande précision n'est pas recherchée pour la primitive obtenue, on peut raisonner sur un petit voisinage, de taille fixée.
- Si on cherche la taille de voisinage la plus grande possible afin d'améliorer la précision de la primitive estimée, on est amené à essayer plusieurs tailles différentes. Ceci conduit à un procédé multi-résolution.

Ces deux idées sont développées dans la suite.

6.1.1 Extraction directe sur un voisinage de taille fixe

La taille est fixé à un certain nombre T de plus proches voisins du point $\mathbf{x_0}$. Pour trouver ce voisinage défini par T points, on procède à la recherche des T plus proches voisins du point $\mathbf{x_0}$, suivant l'algorithme de la page 158. Notons que ceci définit bien un voisinage sphérique autour du point. On réalise ensuite l'ajustement de la primitive sur le voisinage.

L'utilisation de ceci peut se faire avec un T faible (pour éviter de sortir de l'hypothèse H') et éventuellement une étape de croissance (là encore selon les besoins).

L'algorithme peut se résumer par le pseudo-code suivant :

Recherche des T plus proches voisins du point $\mathbf{x_0}$
Selon le type de primitive : pré-traitement pour ajustement (estimation normales)
Ajustement de la primitive sur l'ensemble des T points
Eventuellement : croissance de la primitive

Entrées : nuage de N points, indice i_0 du point $\mathbf{x_0}$, type de primitive
Sorties : primitive et sous-nuage de points associé
Paramètres : taille T(en nombre de points)

Croissance

Si on ne fait rien après l'algorithme ci dessus, on obtient toujours un ensemble de points qui se trouvent dans un voisinage sphérique. Par exemple, si l'on a extrait un plan, on obtient des points sur un disque circulaire. Or, les points de cette primitive ne suivent certainement pas une telle répartition. Ceci est illustré figure 6.2. On peut donc souhaiter effectuer une croissance de la primitive trouvée.

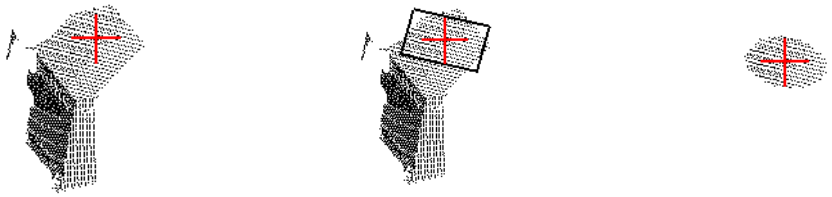


FIG. 6.2 – Extraction locale d'un plan. A gauche : point sélectionné ; au centre : le plan trouvé, à droite : le voisinage final.

Résultats

Le résultat de cet algorithme pour l'extraction de cylindre est montré figure 6.3. Ce procédé est bien entendu limité car il n'est pas toujours aisé de déterminer la taille de voisinage a priori.

FIG. 6.3 – Extraction locale d'un cylindre utilisant une taille fixée, sans croissance (gauche) et avec croissance (droite).

6.1.2 Extraction multi-résolution sur voisinages emboîtés

On souhaite ici extraire une primitive sans spécifier de taille de voisinage fixe. On souhaite également trouver la taille «optimale», c'est-à-dire la taille du plus gros voisinage autour du point $\mathbf{x_0}$ qui ne contient que la surface que l'on souhaite extraire.

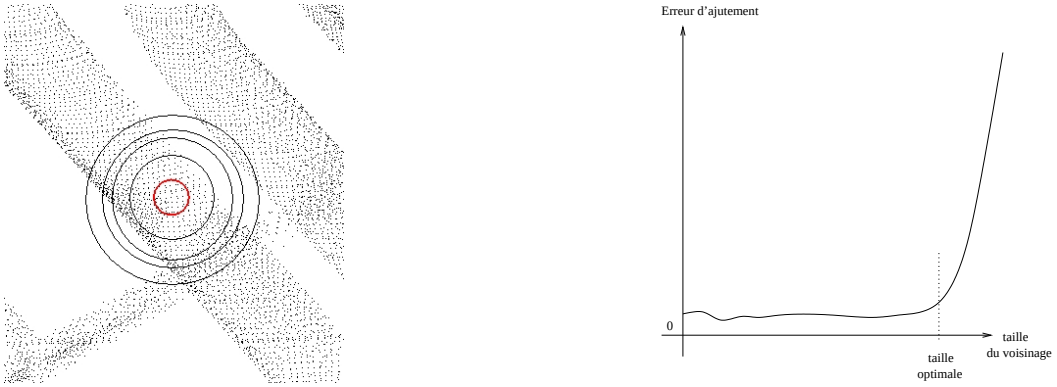


FIG. 6.4 – Voisinages emboîtés autour du point (schéma de gauche) et graphe de l'erreur en fonction de la taille (schéma de droite)

Principe

Le principe est ici un ajustement de primitive sur des voisinages emboîtés autour du point $\mathbf{x_0}$. Pour chaque taille de voisinage, on évalue l'erreur associée à la primitive correspondante. Logiquement, si l'on se place dans la situation où :

1. le point est sur une primitive du type recherché,
2. d'autres surfaces sont présentes dans la scène,

alors l'erreur d'ajustement doit être stable tant que le voisinage ne contient que la primitive, et augmenter rapidement lorsque le voisinage commence à englober des points appartenant à d'autres surfaces. Cette idée est illustrée sur la figure 6.4. La figure 6.5 montre cette notion sur un cas réel. Les graphes de la figure 6.6, qui correspondent à l'exemple de la figure 6.5, montrent également de manière claire qu'il existe une partie stable au niveau des paramètres estimés et donc au niveau de l'erreur d'ajustement.

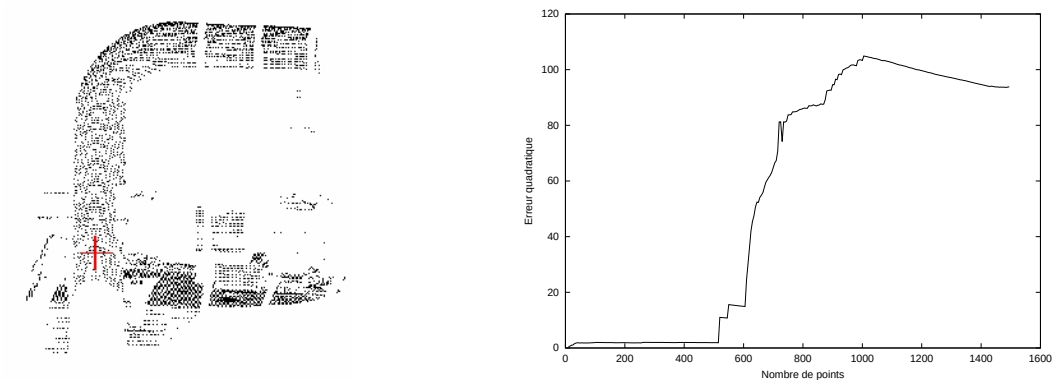


FIG. 6.5 – Présentation du principe sur une scène réelle.
A gauche : la scène «CGP» (issue du scanner Cyrax, société Cyra) et le point cliqué par l'utilisateur (indiqué par le «+»).
A droite : Graphe de l'erreur d'ajustement (erreur quadratique) des points par rapport au cylindre en fonction de la taille du voisinage (en nombre de points) sur la situation à gauche. Ici chaque taille a été testée. Dans l'algorithme multi-résolution développé, on ne teste en fait que quelques tailles (indiquées par des «+» sur le graphe).

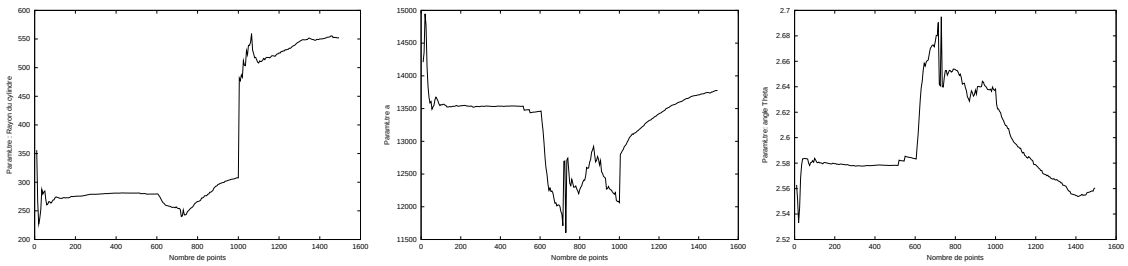


FIG. 6.6 – Graphes des paramètres du cylindre en fonction de la taille de voisinage sur l'exemple de la figure 6.5. le rayon (à gauche), le paramètre *a* (au centre), le paramètre θ (à droite).

Algorithme

Comment repérer le saut et trouver une taille «optimale» ? Ceci met en œuvre un procédé d'ajustement multi-résolution.

On procède en deux phases :

1. expansion jusqu'à détection du saut, puis
2. dichotomie.

La taille optimale est cherchée entre deux tailles (nombres de points) T_{min} et T_{max}. L'algorithme est présenté à travers le pseudo-code suivant :

Pseudo-code de l'algorithme d'extraction locale multi-résolution

Phase préliminaire
  $V_{T_{\max}} \leftarrow$ Recherche des $T_{\max}$ plus proches voisins du point $\mathbf{x_0}$ dans le nuage
  (et tri simultané de ces $T_{\max}$ points en fonction de la distance à $\mathbf{x_0}$)
  Pré-traitement (estimation des normales) de $V_{T_{\max}}$ si besoin
Initialisation
  $\bar{e} \leftarrow 0$ (erreur moyenne sur les tailles testées jusqu'à présent)
  Continuer$\leftarrow$ Vrai
  Phase1$\leftarrow$ Vrai (Si vrai : phase d'expansion, si faux : dichotomie)
  $i \leftarrow 0$
  $T \leftarrow T_{min}$
Tant que (Continuer = Vrai) faire
  $V_T \leftarrow$ T premiers points de $V_{T_{\max}}$
  Ajustement d'une primitive sur V_T (utilisation éventuelle des normales)
  Evaluation de l'erreur quadratique e (sur V_T)
  $\bar{e} \leftarrow (i\bar{e}+e)/(i+1)$
  Si (Phase1= Vrai) faire
    (Phase d'expansion)
    Si ($e < \alpha\bar{e}$) faire
      $[T_m, T_M] \leftarrow [T, 2T]$
      $T \leftarrow T_M$
      Si ($T > T_{max}$) faire : Continuer$\leftarrow$ faux
    Sinon
      (Fin de la phase d'expansion, début de la phase de dichotomie)
      Phase1 $\leftarrow$ faux
      $T_M \leftarrow T$
      $T \leftarrow (T_m + T_M)/2$
    Fin Si
  Sinon
    (Phase de dichotomie)
    Si ($e < \alpha\bar{e}$) faire :
      $T_m \leftarrow T$
    Sinon faire :
      $T_M \leftarrow T$
    Fin Si
    Si ($T_M - T_m < \Delta T_{\min}$) faire : Continuer $\leftarrow$ faux
    $T \leftarrow (T_m + T_M)/2$
  Fin Si
  $i \leftarrow i+1$
Fin Tant que
Retourner la taille optimale T et la primitive correspondante (paramètres, erreur d'ajustement, incertitude sur les paramètres, sous-nuage de points)

Entrées : nuage de N points, point $\mathbf{x_0}$, type de primitive.
Sorties : Primitive et sous-nuage associé
Paramètres : taille $T_{\min}$, taille $T_{\max}$, $\Delta T_{\min}$, α, Nb de pts pour l'estim. des normales.

Choix de $T_{\min}$ Le choix de la taille $T_{\min}$(en nombre de points) est important. Implicitement, on suppose que si le point se trouve bien sur la primitive de type recherché, il y a au moins $T_{\min}$ voisins du point qui se trouvent sur cette primitive.

D'autre part, T_{min} doit être suffisant pour éviter le caractère aléatoire lié aux petites tailles (se reporter à la discussion de la section 6.1.2, page 167).

Dans les tests de validation de l'algorithme d'extraction de ligne de tuyauterie présenté section 3.2, page 51, T_{min} a été fixé à la valeur 100.

Erreur d'ajustement Dans notre application, l'erreur d'ajustement utilisée est la moyenne quadratique des distances des points à la primitive, encore appelée *résidu quadratique moyen*, définie par :

$$e(t) = \sqrt{\frac{1}{t}\sum_{\mathbf{x}\in V_t} d^2(\mathbf{x}, \text{Primitive}(V_t))}$$

où V_t désigne le voisinage de t points, c'est-à-dire l'ensemble des t plus proches voisins du point $\mathbf{x_0}$. Nous désignons encore cette erreur d'ajustement par le terme : *erreur quadratique*. Cette quantité semble appropriée ici en ce qu'elle est très sensible à la présence de points aberrants. On peut remarquer que l'estimation de la primitive de moindres carrés utilise cette fonction comme fonction coût.

Voisinages Les voisinages utilisés ici sont définis comme l'ensemble des T plus proches voisins (au sens de la norme euclidienne de $\mathbb{R}^3$) du point sélectionné. La taille T (en nombre de points) d'un tel voisinage varie entre des valeurs minimum et maximum, $T_{\min}$ et $T_{\max}$. L'intérêt de cette définition est d'être indépendante des dimensions de la primitive (diamètre, longueur, ...), par opposition à un rayon de sphère de voisinage fixe. Par contre, ceci dépend de la densité de points locale.

En pratique, pour l'extraction de cylindre initial de l'algorithme de la section 3.2, page 51, les tailles $T_{\min}$ et $T_{\max}$ ont été fixées à 100 points et 2000 points.

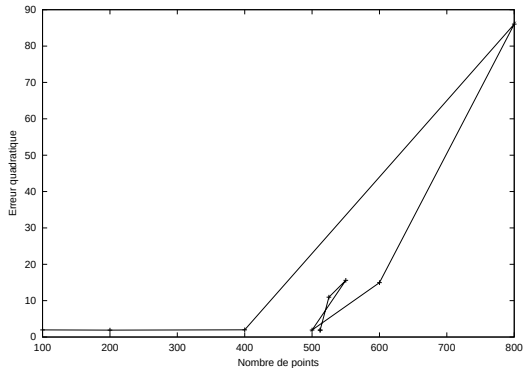
Procédé Il est coûteux de réaliser l'ajustement pour chaque taille entre $T_{\min}$ et $T_{\max}$. Aussi procède-t-on un peu plus grossièrement en ne considérant que quelques valeurs de tailles ($T_{\min}$ multiplié par des puissances de 2, le tout restant inférieur à $T_{\max}$). A l'aide d'un critère sur l'erreur, on repère la taille où l'erreur a augmenté de manière significative. On obtient ainsi un encadrement de la taille optimale. Ensuite, on souhaite être plus précis : une valeur de taille plus précise est évaluée par dichotomie dans l'intervalle obtenu. La figure 6.7 présente une illustration du procédé.

Construction des voisinages La construction des voisinage emboîtés se fait comme suit :

- Recherche des $T_{\max}$ plus proches voisins du point. Ceci fait intervenir un tri des $T_{\max}$ points par ordre croissant de la distance au point (algorithme page 158).
- Pour déterminer les T plus proches voisins (avec $T < T_{\max}$), il suffit de prendre les T premiers éléments de la liste triée des $T_{\max}$ points.

L'intérêt de ceci est que l'on effectue la recherche de voisins qu'une seule fois.

Pré-traitement Les procédés d'initialisation correspondant à l'ajustement de certaines primitives nécessitent l'estimation des normales estimées en chaque point (voir chapitre 4). Ceci est également fait une seule fois lors du pré-traitement, pour les $T_{\max}$ points (voir algorithme d'estimation en annexe B).



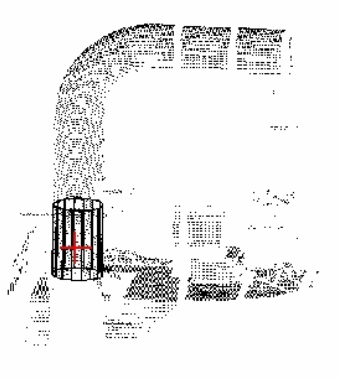


FIG. 6.7 – Résultat sur la scène «CGP». A gauche : graphe des erreurs pour les tailles testées lors du procédé multi-résolution. La ligne brisée suit la phase d'expansion puis la phase de dichotomie jusqu'à la taille optimale (les sommets de la ligne brisée sont les mêmes que les «+» apparaissant dans la figure 6.5). A droite : le cylindre extrait de la scène à l'issu du procédé multi-résolution.

Croissance De même que pour l'algorithme précédent, il est possible d'effectuer une croissance de la primitive obtenue. Ceci peut être souhaité dans certains cas.

Valeurs des paramètres Pour l'extraction de cylindre utilisée dans le chapitre 3, les valeurs suivantes ont été choisies pour les paramètres : T_{min}=100 points, T_{max}=2000 points, ΔT_{min}=25 points, α=1.5, Nombre de voisins pour l'estimation des normales : 10.

Résultats et discussion

Résultats Cette méthode a été implémentée pour chacune des primitives libres du chapitre 4. Les figures 6.8, 6.9 et 6.10 montrent des exemples pour le tore, le cône et la sphère. Dans la figure 6.10, il

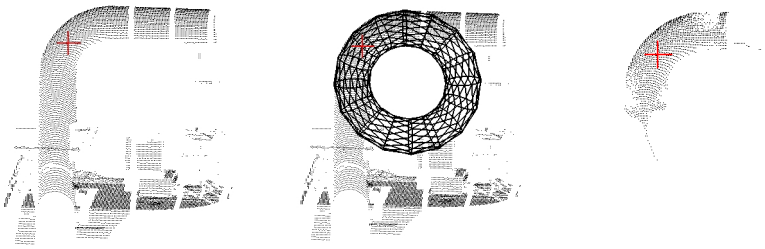


FIG. 6.8 – Extraction locale d'un tore dans la scène «CGP» (les extrémités du tore n'ont pas été calculées ici).

s'agit d'une sphère servant au recalage, que l'on place dans la scène. Notons que puisque l'on connaît précisément le rayon de ces sphères, on pourrait également réaliser une extraction ou un ajustement de sphère contrainte (de rayon fixe).

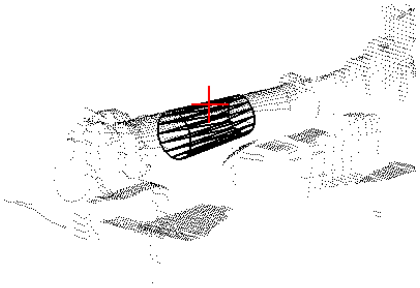


FIG. 6.9 – Extraction locale d'un cône.

FIG. 6.10 – Extraction locale d'une sphère.

Cette méthode est utilisée avec les cylindres pour l'étape du «cylindre initial» de l'algorithme d'extraction de lignes de tuyauterie du chapitre 3, section 3.2, page 51.

Discussion

Choix de $T_{\min}$ La primitive construite par ce procédé est correcte, tant que la valeur de $T_{\min}$ n'est pas trop grande. Plus précisément, tant que les $T_{\min}$ plus proches voisins du point $\mathbf{x_0}$ se trouvent sur la même primitive. Ceci est problématique dans des zones où la densité de points est faible et où il existe des surfaces proches. Dans ce cas, il se peut que le paramètre $T_{\min}$ ne permette pas de sélectionner des points appartenant seulement à la primitive recherchée.

En fait, deux types de cas à problèmes se produisent concernant cette taille :

– Il existe des cas pour lesquels une telle taille T_{min} est trop grande. La situation est ici le cas où la taille optimale est plus petite que T_{min}. Vu que l'on cherche les valeurs plus grande que T_{min}, on ne peut pas trouver la bonne valeur de T. La primitive peut ainsi être complètement fausse.
– Si T_{min} est trop petite (e.g., de l'ordre de 10 points), l'algorithme peut détecter un saut de l'erreur quadratique avant la taille optimale. La primitive peut là aussi être complètement fausse. En effet, prenons l'exemple d'une zone très dense, contenant un cylindre de gros diamètre. Dans ce cas, les voisinages les plus petits représentent un étalement angulaire très faible sur le cylindre, ce qui fait que l'on peut encore se trouver dans la zone aléatoire, et stopper de manière précoce.

Plusieurs méthodes sont envisageables pour traiter ce problème :

1. utiliser une valeur de $T_{\min}$ faible en changeant le critère de détection du saut. Il s'agit de dépasser la zone aléatoire, par exemple en faisant intervenir l'incertitude sur les paramètres du modèle

dans le critère d'arrêt.

2. utiliser des méthodes de δ-connexité, ou de maillage surfacique pour déterminer quels sont les vrais voisins au lieu de considérer des voisinages sphériques. Notons que ce procédé ne permet pas de traiter tous les cas !

3. faire appel à des méthodes d'extraction au sein même des voisinages sphériques (éventuellement avec des points aléatoires). L'idée est ici de considérer que l'hypothèse H' n'est peut-être pas vérifiée sur ce voisinage et de considérer des méthodes du 6.5.

A propos du critère de détection du saut Le critère pour stopper la phase montante est basé sur une heuristique et la valeur paramètre α arbitraire. La difficulté ici est que la forme du graphe de l'erreur en fonction de la taille n'est pas la même suivant la primitive et les surfaces qui se trouvent proches.

Cependant, on peut envisager d'utiliser d'autres informations que l'écart-type *e* des écarts pour valider ou invalider le modèle à chaque taille (section 5.2). Cette possibilité n'a pas encore été testée.

6.2 Extraction globale d'une primitive de type fixé

Principe

Ici, on ne sait pas où se trouve la primitive dans la scène. L'idée est d'appliquer l'une des méthodes d'extraction locales précédentes sur plusieurs points «germes» de la scène et de chercher le point où il est le plus probable qu'il existe une primitive du type recherché.

Comme précédemment, si on connaît une taille de voisinage sur laquelle on souhaite trouver la primitive, ou s'il n'est pas crucial d'obtenir une taille optimale, alors on peut utiliser la fonction d'extraction sur taille fixe de la section 6.1.1. Sinon, on peut utiliser la méthode multi-résolution de la section 6.1.2.

Il est en général coûteux, et inutile, de tester chaque point de la scène. Le principe suivi ici a été de tirer des points aléatoirement. Notons que ce choix, essentiellement motivé par une question de simplicité, n'est pas le seul possible. En effet, on peut également effectuer un sous-échantillonnage régulier dans l'espace, ou détecter des points d'intérêt par des procédés «bas niveau» (courbure faible, normales, image de Gauss, ...).

A chaque essai, on souhaite évaluer le point germe testé par rapport aux autres points (testés précédemment). Ceci se fait par le biais d'une fonction coût. Une fonction coût possible est l'opposé du nombre de points obtenu à l'issue de l'extraction locale.

Algorithme

La méthode est présentée à travers le pseudo-code suivant :

$n_{opt} \leftarrow 0$ (nombre de points optimal)
Pour k=1 à K faire
  Choisir un point $\mathbf{x}$ au hasard parmi les N points
  Extraire localement une primitive P
  sur un voisinage de taille T autour de $\mathbf{x}$ (§ 6.1)
  Evaluation de la primitive P :
  $n_\sigma \leftarrow 0$ (nombre de points courant)
  Pour i=1 à N faire
    Si $d(\mathbf{x_i}, P) < \sigma$ faire : $n_\sigma \leftarrow n_\sigma + 1$
  Fin Pour

  Si $n_{opt} < n_\sigma$ faire
    (garder cette primitive)
    $\mathbf{x_{opt}} \leftarrow \mathbf{x}$
    $n_{opt} \leftarrow n_\sigma$
  Fin Si
Fin Pour
Si $n_{opt} > n_{\min}$ faire
  Retourner $\mathbf{x_{opt}}$, la primitive P et le sous-nuage associés
Fin Si

Entrées : Nuage de N points, type de primitive.
Sorties : Primitive P (si $n_{opt} > n_{\min}$), sous-nuage associé
Paramètres :
nombre d'essais K, taille des voisinages T(en nombre de points),
épaisseur σ, nombre minimal d'*inliers* $n_{\min}$

Le pseudo-code montre l'utilisation de l'extraction locale avec une taille T fixe, mais on peut aussi utiliser l'extraction multi-résolution entre des tailles $T_{\min}$ et $T_{\max}$.

On remarque que l'étape qui consiste à compter le nombre d'*inliers* revient à faire une croissance de la primitive dans toute la scène et à compter le nombre de points obtenus.

Résultats et discussion

Cette fonction a été utilisée pour les cylindres (Figure 6.11) et les plans.

Le choix des paramètres qui interviennent dans l'algorithme est important. En particulier, le choix des valeurs de K, T et σ est crucial. Le paramètre K doit être pris le plus grand possible (mais est bien sûr limité par des questions de complexité algorithmique). Le paramètre σ est déduit d'une certaine connaissance de la scène et de la précision recherchée pour l'extraction. Il n'est pas aisé de choisir une valeur pour T. Un choix prudent peut consister à prendre une valeur petite, qui équivaut à la taille minimal $T_{\min}$ de l'algorithme du 6.1.2. Si, à la place de l'algorithme d'extraction locale à taille fixe, on utilise le procédé multi-résolution du 6.1.2, on peut sans doute se passer de l'étape de croissance et considérer directement la taille obtenue à l'issue de l'extraction locale pour la fonction coût.

Le fait de choisir des points au hasard présente l'inconvénient de ne pas garantir le même comportement pour deux exécutions du programme. Cet aspect est atténué si le nombre d'essais K est grand.

En pratique, on constate que ce type d'algorithme produit une solution dans la plupart des cas, y compris dans les cas où il n'y a aucune instance de primitive du type recherché. Nous nommons une

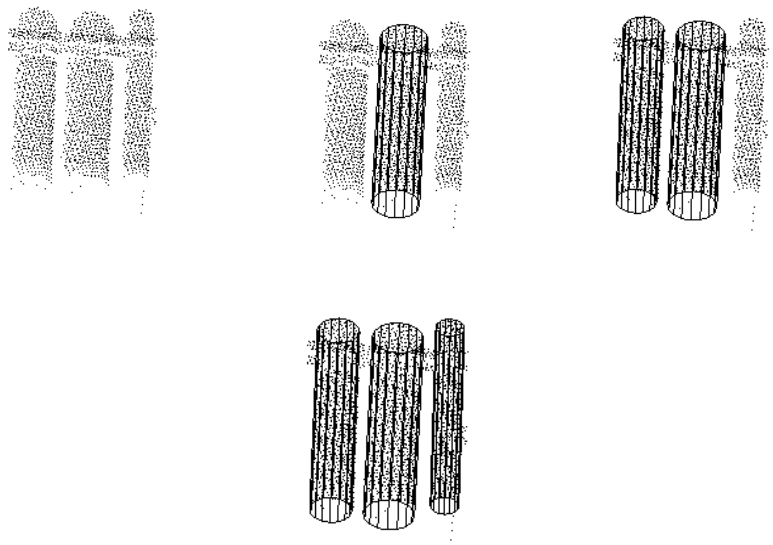


FIG. 6.11 – Extraction globale d'un cylindre (action répétée trois fois)

telle solution un «faux positif». Ceci dépend du paramètre $n_{\min}$, mais pas uniquement. A ce niveau, il peut être intéressant de faire intervenir des critères visant à valider ou invalider le modèle, comme ceux présentés au chapitre 5.2. Pour l'instant, on procède simplement à une vérification des paramètres estimés, présentée § 5.2.1, page 138.

6.3 Extraction globale d'un nombre inconnu de primitives de type fixé

Objectif

La méthode précédente vise à extraire globalement une primitive. Si plusieurs primitives du même type sont présentes, il paraît intéressant de les extraire toutes. C'est l'objectif de la méthode développée ici. La difficulté réside ici dans le fait que l'on ne connaît pas le nombre de primitives présentes.

Principe

Comment extraire plusieurs primitives ? Le principe qui a été suivi est d'extraire les primitives une par une, et ce tant qu'il y en a dans la scène. On procède donc à une itération de la méthode précédente, en enlevant à chaque fois les points correspondant à la primitive trouvée.

Algorithme

L'algorithme développé est présenté à travers le pseudo-code suivant :

NuageRestant ← Nuage global (N points)
$N_r \leftarrow N$ (nombre de points dans NuageRestant)
$k \leftarrow 1$
Continuer ← vrai
Tant que (Continuer = vrai) faire
  Extraire globalement un cylindre dans NuageRestant
  Paramètres :
    nombre d'essais : $N_r/5$, épaisseur σ,
    nb min d'inliers : $\max(10, N_r/20)$, Nb de voisins : 50
  Si un cylindre a été extrait faire
    Mettre à jour NuageRestant et N_r
    Si (($N_r < N_{\min}$) ou (k = K)) faire : Continuer ← faux
  Fin Si
  $k \leftarrow k+1$
Fin Tant que

Entrées : Nuage de N points, type de primitive.
Sorties : Primitives P, sous-nuages associés.
Paramètres : nombre maximum d'itérations K, épaisseur σ,
nombre minimum de points $N_{\min}$.

Résultats et discussion

Cette méthode a été implémentée dans le cas des cylindres. La figure 6.12) montre un exemple de résultat d'extraction de cylindres. On voit sur cet exemple que les trois cylindres présents dans la

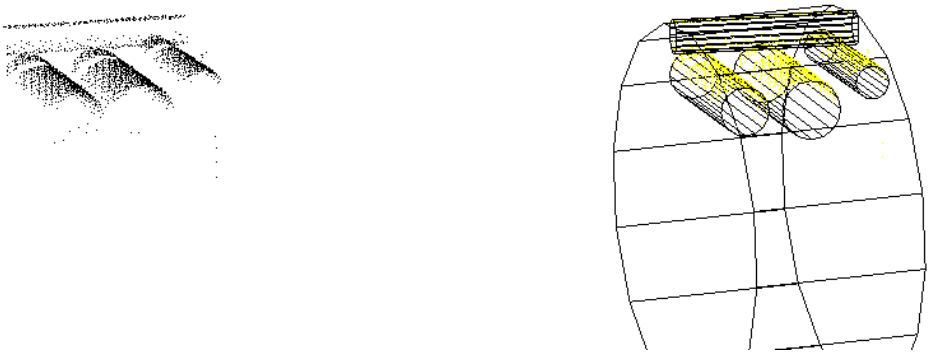


FIG. 6.12 – Résultat de l'algorithme d'extraction de cylindres libres

scène sont effectivement extraits. L'algorithme a d'autre part extrait deux faux cylindres sur les points correspondant à la poutre.

La présence de ces faux-positifs ne constitue pas toujours un problème. Cette méthode a par exemple été utilisée dans l'algorithme d'extraction de lignes de tuyauterie, lors des arrêts de l'algorithme en présence d'ombres. Ce procédé est présenté à la section 3.2.5, page 72. Or, dans ce contexte, on cherche des cylindres respectant certaines contraintes, en l'occurrence des cylindres qui se trouvent sur le même tuyau. L'idée était alors d'extraire des cylindres dans une zone telle que celle montrée figure 6.12, et d'examiner si l'un des cylindres est compatible avec les contraintes. Ce procédé permet-

trait ici d'éliminer les faux cylindres de la figure 6.12. Dans ce contexte, il est en effet peu probable que les faux-positifs respectent les contraintes imposées.

Si l'on souhaite cependant fournir une description satisfaisante de la scène, il convient d'éviter de tels faux-positifs. Il me semble que ceci ne peut se faire qu'en introduisant plusieurs primitives, simultanément au sein de l'extraction. Cette perspective est abordée en section 6.4.

On remarque que cet algorithme fait intervenir de nombreux paramètres. De plus, le choix de ces paramètres est délicat. En particulier, les paramètres choisis pour l'extraction d'une primitive sont empiriques. La condition d'arrêt peut notamment poser problème : comment fixer $N_{\min}$ sachant que dans les scènes traitées, il y a toujours un peu de points qui ne sont pas sur des primitives simples (fils électriques, formes complexes, points aberrants, etc.) ? Ceci met en lumière le besoin de valider ou invalider les modèles trouvés, soit au niveau de l'extraction d'une primitive, soit au niveau final. Bien que ceci n'ait pas encore été testé, il paraît pertinent de faire intervenir des critères tels que ceux développés à la section 5.2.

Remarquons enfin qu'avec l'algorithme présenté ici, les relations de voisinage changent à chaque itération (à chaque fois que l'on enlève des points) : si on cherche les k plus proches voisins d'un point parmi les points du nuage restant, on n'obtient pas la même chose suivant ce qu'il reste dans le nuage restant. Les relations de voisinages varient donc au cours du temps. Cet aspect peut paraître gênant, en ce que l'ordre d'extraction des primitives a une importance. Par contre, lorsque l'on se trouve effectivement sur une primitive, ceci peut être intéressant car la scène est plus «dégagée».

6.4 Perspectives : extraction de primitives de types non fixés

Dans le cas étudié ici, on souhaite extraire des primitives sans spécifier de quels types elles sont (plan, sphère, cylindre, cône, tore). Nous avons choisi de réaliser l'identification du type de primitive en faisant intervenir l'ajustement de chacune des primitives et la sélection du modèle par des techniques développées au chapitre 5, section 5.1, page 128.

Une telle méthode d'extraction a été implémentée en adaptant simplement l'extraction locale multi-résolution du 6.1.2 à plusieurs primitives. Les cas de l'extraction globale d'une ou de plusieurs primitives n'ont pas été implémentés et sont abordés ici en tant que perspectives.

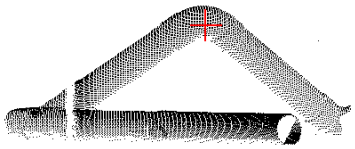
Extraction locale d'une primitive de type non fixé

On peut facilement étendre le principe de l'extraction locale avec taille fixe de la section 6.1.1 au cas de plusieurs primitives, mais se pose ici un problème d'échelle. En effet, lorsque l'on ne connaît pas le type de primitive, tout dépend de la taille du voisinage sur lequel on cherche une surface. Prenons l'exemple de points se trouvant sur un tore. Sur un très petit voisinage, le plan est une bonne approximation de la surface formée par les points. Sur un voisinage un peu plus grand, le cylindre est une approximation satisfaisante. Enfin, ce n'est qu'en prenant un voisinage encore plus grand que l'on s'aperçoit que le tore est la meilleure approximation. L'identification du type d'une primitive est donc liée à la taille du voisinage sur lequel on ajuste cette primitive. C'est pourquoi, dans un contexte multi-primitive, l'algorithme d'extraction locale multi-résolution de la section 6.1.2 est préférable.

Pour adapter cet algorithme au cas de plusieurs types de primitives possibles, on remplace simplement l'ajustement de la primitive de type connu par l'ajustement successif des primitives de chaque type : plan, sphère, cylindre, cône et tore. Un premier contrôle sur les primitives obtenues est effectué

à ce niveau. Celui-ci consiste tout simplement à vérifier la validité des paramètres estimés (voir section 5.2.1). La sélection du meilleur modèle parmi ceux-ci se fait par simple comparaison des erreurs d'ajustement parmi les modèles qui n'ont pas été invalidés par le premier contrôle (voir section 5.1). Tout ceci est fait pour chaque taille de voisinage (nombre de voisins de $\mathbf{x_0}$) testée. Ceci signifie que l'on peut obtenir des primitives de types différents selon la taille. L'erreur e qui est considérée à chaque étape est l'erreur d'ajustement de la primitive sélectionnée. Cette erreur correspond au minimum des erreurs de chaque primitive qui n'a pas été invalidée (celle-ci n'est donc pas toujours égale au minimum des erreurs). On peut alors procéder comme dans le cas d'une primitive de type fixé avec l'erreur obtenue à chaque étape.

La figure 6.13 donne une idée du procédé, à travers le graphe des erreurs d'ajustement de chacune des primitives : pour une taille donnée, le graphe de l'erreur considérée se trouve sous tous les graphes correspondants aux modèles qui n'ont pas été invalidés.



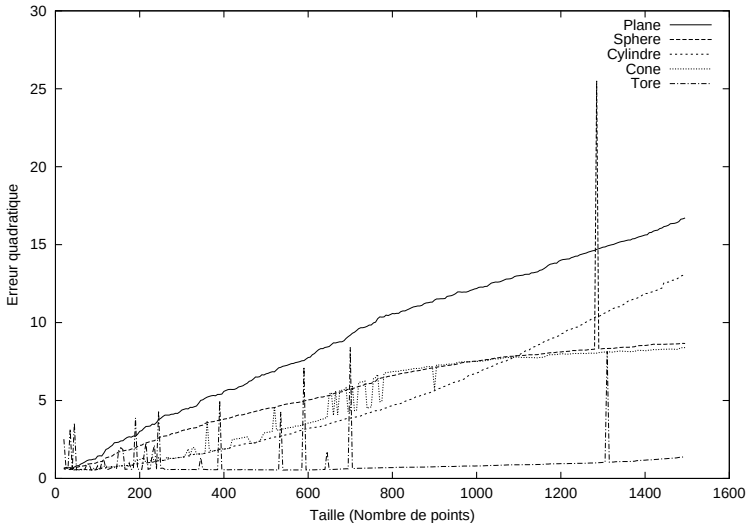


FIG. 6.13 – Graphe des erreurs d'ajustement du plan, de la sphère, du cylindre, du cône, et du tore en fonction de la taille des voisinages emboîtés autour du point $\mathbf{x_0}$.

La figure 6.14 montre quelques résultats où l'extraction avec sélection de modèle donne un résultat satisfaisant.

La version actuelle de cet algorithme ne trouve pas toujours la primitive adéquate. Certains des cas qui posent problèmes sont générés par les cas qui ne sont pas bien gérés par la sélection de modèle. C'est le cas par exemple lorsque plusieurs surfaces sont présentes dans le voisinage (Figure 5.8, page 136). Des procédés de validation de modèle tels que ceux développés section 5.2 semblent pouvoir fournir des critères pour traiter ces cas.

Ce type d'algorithme met d'autre part les procédés d'ajustement à rude épreuve. En effet, pour chaque taille, il faut que l'ajustement de chaque primitive produise la solution de moindres carrés. Or toutes les primitives ne peuvent pas être satisfaisantes sur le même nuage de points. Ceci implique notamment que l'initialisation et/ou la fonction coût de certaines primitives ne mènent pas forcément à une solution très claire. Il semble que les pics observés sur les graphes de la figure 6.13 soient dus à des instabilités de ce genre. Même s'il est apparemment peu probable de tomber sur l'un de ces pics lors du procédé multi-résolution, l'extraction fonctionnerait mieux si la stabilité des méthodes d'ajustement était améliorée. Cette question demanderait en tout cas un examen plus précis.

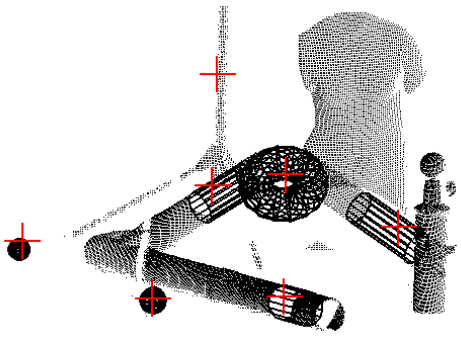


FIG. 6.14 – Quelques primitives extraites localement sans spécifier leurs types

Extraction globale d'une primitive de type non fixé

Pour réaliser l'extraction globale, on peut procéder de même que précédemment, en utilisant l'extraction locale et en introduisant une fonction coût globale. En théorie, les résultats obtenus devraient être meilleurs que ceux de la version pour primitive de type fixé, étant donné que le nombre de faux positifs devrait être inférieur.

Notons qu'il se pose encore le problème de ce qui n'est pas modélisable : points aberrants, formes complexes, etc. Ces situations génèreront encore des faux positifs.

Extraction globale de plusieurs primitives de types non fixés

L'objectif ultime ici est bien entendu de décrire de manière exhaustive la scène en terme de primitives géométriques. C'est la notion de segmentation en primitives géométriques que nous avons introduite aux chapitres 1 et 2.

Comme dans le cas de primitives de type fixé, l'idée la plus naturelle est d'extraire les primitives et d'enlever les points correspondants au fur et à mesure. Les questions clés sont alors de savoir comment adapter de manière cohérente les paramètres de l'algorithme lorsque la scène évolue à chaque étape, et comment définir la condition d'arrêt.

6.5 Méthode à partir de quorum de points

L'hypothèse H' présentée en début de chapitre n'est en général pas vérifiée en tout point d'une scène. En effet, il suffit de prendre des points se trouvant à la frontière entre deux surfaces, aux angles, ou dans des zones où il y a de nombreuses surfaces et une densité faible. La figure 6.15 montre quelques situations où l'hypothèse H' n'est plus valable. Un autre exemple est celui d'un point sur une poutre en I, qui serait scannée des deux côtés. Le point se trouve bien sur un plan, mais les deux plans sont très proches, et rien ne dit que la densité de points est partout telle que l'on puisse définir un tel voisinage sphérique. Il convient toutefois de formuler deux remarques par rapport à cet exemple :

1. il n'est pas rare de trouver de tels zones dans une scène, mais ces zones représentent en général peu de points par rapport à la scène totale (en particulier car on scanne beaucoup d'objets pleins),

FIG. 6.15 – Situations où la méthode à base de quorum de points peut s'avérer intéressante (surfaces confinées).

2. l'exemple donné précédemment se ramène à notre hypothèse si l'on introduit une primitive de plus haut niveau (ici ce serait une primitive «poutre»). Ce qui est possible (voir perspectives sur l'ajustement de primitives composées, section 4.5.5, page 123).

La méthode présentée dans cette partie est une adaptation de travaux existants [RL93]. Nous avons en particulier appliqué ce cadre aux cas du plan, du cylindre (avec les normales) et du cylindre-rotule (la définition de cette primitive est présentée au chapitre 4). Pour d'autres primitives, telles que le cylindre (à partir des points seuls), le cône ou le tore, la question s'avère plus difficile.

Ces méthodes sont a priori applicables à des scènes complètes [CGL01, CGL02, RL93]. Cependant, les résultats obtenus sur des scènes de taille importante ont fait apparaître la présence de nombreux faux-positifs. Il s'avère de plus délicat de régler les paramètres afin d'éliminer ces cas. Par contre, il existe des situations où cela devient intéressant : points aberrants, surfaces confinées, etc. (Figure 6.15). Ainsi, il semble que cette méthode soit réellement intéressante dans les cas où les méthodes développées précédemment dans ce chapitre ne sont plus valides, c'est-à-dire lorsque l'hypothèse H n'est pas valable.

Cette méthode fait intervenir la notion d'échantillon minimal ou quorum. Pour une primitive de type donné, un quorum désigne un ensemble de points dont le nombre d'éléments est le minimum nécessaire pour construire ce type de primitive.

Principe

Le principe de la méthode suggérée par Roth et Levine [RL93] peut se résumer par le pseudo-code suivant :

```
Pour k=1 à K faire
    Choisir un quorum de q points au hasard
    Construire la primitive P à partir du quorum de points
    Evaluer la primitive P à l'aide d'une fonction coût F
    Test d'optimalité :
    Si F(P) < F(P_opt) faire
        P_opt ← P (garder cette primitive)
    Fin Si
Fin Pour
Rendre la primitive P_opt
```

Ceci constitue un cadre très général, qui généralise des méthodes telles que la méthode «RANSAC» [BF81] ou la transformée de Hough.

Puisque le quorum de points est choisi aléatoirement, il n'y a aucune hypothèse sur le fait que l'on puisse dégager la surface sur un voisinage. On voit donc que cette méthode n'utilise pas l'hypothèse H'.

Algorithme

Dans [RL93], les auteurs suggèrent d'utiliser par exemple le nombre d'«inliers» à un seuil près comme fonction mérite, c'est-à-dire l'opposé du nombre d'inliers comme fonction coût. Ceci donne lieu a l'algorithme suivant :

$n_{opt} \leftarrow 0$
Pour k=1 à K faire
  Choisir un quorum de q points parmi les N points du nuage
  Construire la primitive P à partir du quorum de points
  $n_\sigma \leftarrow 0$
  Pour i=1 à N faire
    Si $|d(\mathbf{x}(\mathbf{i}); P)| < \sigma$ faire : $n_\sigma \leftarrow n_\sigma + 1$
  Fin Pour

  Si $n_{opt} < n_\sigma$ faire
    (garder cette primitive)
    $P_{opt} \leftarrow P$
    $n_{opt} \leftarrow n_\sigma$
  Fin Si
Fin Pour
Rendre la primitive P_{opt}

Entrées : Nuage de N points, type de primitive (quorum de q points).
Sorties : primitive P_{opt}.
Paramètres : Nombre d'essais K, épaisseur σ.

Nous venons de voir que l'algorithme comporte deux paramètres : σ et K. L'épaisseur σ peut éventuellement être spécifiée à partir d'une certaine connaissance (globale ou locale) du niveau de bruit de la scène.

Comment choisir le nombre d'essais K ? Dans [RL93], ce nombre K est relié à la probabilité qu'un point tiré au hasard dans la scène se trouve sur la primitive cherchée. Il est ainsi possible de choisir une valeur de K minimum. Notons toutefois que plus ce nombre K est grand, et plus il est probable que l'on trouve la primitive cherchée. Autrement dit, plus ce nombre est grand, moins l'aspect aléatoire lié à la sélection des points est important. Dans les tests que nous avons effectués en pratique, nous avons choisi des valeurs de K grandes pour que le caractère aléatoire intervienne le moins possible (de l'ordre de N pour le cas du plan).

Cet algorithme est très proche de l'algorithme d'extraction globale avec taille fixe présenté section 6.1.1, page 160, mais ici on ne réalise plus un ajustement de moindres carrés sur un voisinage. Notons cependant que l'on peut choisir le quorum de points dans un voisinage si l'on souhaite traiter une scène «complète». La différence se situe donc essentiellement dans la manière de construire la primitive à partir de points. Cependant, nous ne pensons pas que cette méthode, en tout cas avec la fonction coût utilisée ici, puisse être très intéressante pour l'extraction dans une scène globale. En effet, ce type de méthode présente l'inconvénient de produire des faux positifs, et ce d'autant plus que la scène est importante. Plusieurs raisons interviennent ici :

– Dans la méthode de la section 6.1.1, on choisit uniquement un point au hasard. Ici, on doit choisir q points au hasard. Ainsi, si la probabilité qu'un point appartienne à la primitive considérée est p, la probabilité que q points aléatoires appartiennent tous à la primitive considérée est p^q. Il est donc clair que pour donner des résultats équivalents, la méthode présentée ici nécessite beaucoup plus d'échantillons. Autrement dit, K doit être beaucoup plus élevé. Pour des valeurs de K semblables, la probabilité d'obtenir un faux-positif est plus élevée avec la présente méthode qu'avec la méthode du 6.1.1. On peut toutefois limiter cet aspect en choisissant les q points dans un voisinage.
– En outre, les voisinages sphériques présentent l'avantage de fournir un moyen d'éliminer les fausses primitives. En effet, ces voisinages contiennent les voisins du point choisi, alors que la primitive formée par les q points ne contient pas forcément les voisins des q points.

Par contre, ceci est intéressant dans certaines parties de la scène, en particulier dans des zones où des surfaces confinées rendent difficile l'hypothèse H ou H' exprimée au début de ce chapitre.

Résultats

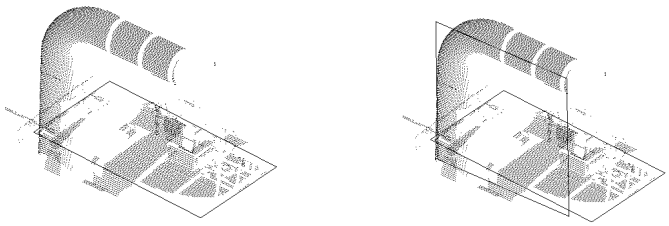


FIG. 6.16 – Extraction globale d'un plan. La première extraction a construit un plan effectivement présent dans la scène (à gauche). La deuxième extraction de plan, effectuée sur le nuage restant, a donné lieu à un faux positif (à droite).

La figure 6.16 montre l'extraction d'un plan dans une scène par cette méthode. Cette figure illustre également la présence de faux positifs.

Cette méthode a également été implémentée pour la primitive «cylindre-rotule», dans le contexte de l'extraction de ligne de tuyauterie (section 3.2).

Concernant la construction d'une primitive à partir d'un quorum de points

quorum pour nos primitives Le nombre de points nécessaire et suffisant pour construire une primitive donnée correspond au nombre de paramètres libres de cette primitive. En effet, le nombre de

points fournit un nombre d'équations, qui doit être égal au nombre d'inconnues. On obtient donc que le nombre de points nécessaire et suffisant (lorsque les points ne sont pas dégénérés) est 3 pour le plan, 4 pour la sphère, 5 pour le cylindre, 6 pour le cône, 7 pour le tore, 2 pour le cylindre-rotule, ...

Comment construire une primitive à partir d'un quorum de points ? Dans la méthode présentée, la construction de la primitive à partir des points est primordiale. Or, même si les primitives auxquelles on s'intéresse sont simples (le nombre de paramètres est en particulier faible), cette construction n'est pas simple pour toutes ces primitives.

Surface du type $a^T q(x,y,z) = 0$ Pour une surface du type $a^T q(x,y,z) = 0$, linéaire en ses paramètres a, la construction de la primitive en fonction du quorum est directe par un calcul de déterminant de Gram (voir annexe D). Ceci inclut notamment les cas du plan et de la sphère. En effet, prenons le cas du plan : tout plan peut se décrire par l'équation $a^T q(x,y,z) = 0$ avec $q(x,y,z) = (1,x,y,z)$, et toute équation du type $a^T(1,x,y,z) = 0$ avec $a \neq 0$ est celle d'un plan. De même pour la sphère : l'équation de toute sphère s'écrit $a^T(1,x,y,z,(x^2+y^2+z^2)) = 0$, et toute équation de ce type avec $a \neq 0$ est l'équation d'une sphère.

Les autres primitives libres : cylindre, cône et tore n'entrent pas dans ce cas. En effet, prenons l'exemple du cylindre. Il n'existe pas d'expression algébrique pour décrire spécifiquement la classe des cylindres, l'expression algébrique la plus simple étant celle des quadriques. On pourrait bien entendu chercher la quadrique à partir de 9 points et ensuite examiner si celle-ci est un cylindre. Le problème est que le cylindre correspond à un cas singulier de quadrique, qui est très peu probable en pratique (particulièrement en présence de bruit sur les points). On ne peut donc pas résoudre le cas du cylindre par ce formalisme. Il en est de même pour les cas du cône de révolution et du tore circulaire : le cône est également une quadrique dégénérée, et le tore est une tétrique dégénérée.

Cylindre de révolution Le cas du cylindre, qui est le plus simple des trois cas restants, a été étudié spécifiquement. Bien que d'apparence simple, le problème de construire le(s) cylindre(s) défini(s) par 5 points de $\mathbb{R}^3$ est un problème complexe et n'a semble-t-il été traité que très récemment [Tre00, OBPT01].

En particulier, ce problème n'a dans le cas général pas de solution analytique, et on se ramène donc à une résolution numérique.

Notre résolution du problème a consisté à partir des 5 équations exprimant la nullité de la distance de chacun des points du quorum au cylindre, et à simplifier ce système d'équations (voir annexe D). Cette simplification est nécessaire car le nombre d'équations initial est élevé et l'espace de recherche des paramètres de position et de taille est très vaste.

Nous avons ainsi obtenu un système d'équations polynomiales de 3 équations à 3 inconnues, qui sont les composantes du vecteur directeur de l'axe. Ce système d'équations est proche de celui qui est trouvé indépendamment dans [OBPT01]. Cependant, notre système (présenté en annexe D) est de degré inférieur à celui obtenu dans [OBPT01] (degrés (3,2,2) contre (3,3,2)). Ceci est intéressant pour la résolution (numérique) de ce système.

Le système obtenu est déjà riche en enseignement : 5 points donnés peuvent définir entre 0 et 6 cylindres (par le théorème de Bézout sur les équations algébriques). Ni l'existence ni l'unicité ne sont donc garanties pour 5 points quelconques.

Il semble que ce système ne puisse plus se simplifier. Afin de trouver les valeurs des paramètres, il faut donc recourir à des méthodes de résolution numériques.

Plusieurs méthodes sont possibles ici. Etant donné que les 3 inconnues restantes se trouvent dans un domaine borné simple, et que l'on souhaite a priori trouver toutes les solutions sur ce domaine, cette situation paraît adaptée à une méthode d'analyse par intervalles. Aussi avons-nous testé cette solution, avec le programme «*Interval_Solver*» de la bibliothèque ALIAS du projet SAGA de l'INRIA, et avec le programme «IA Solver» sur lequel travaille Isabelle Braems, doctorante à Orsay. La résolution a été testée sur quelques exemples réels et synthétiques. Les résultats ont confirmé la validité des équations, ainsi que le fait que la résolution par analyse par intervalles est possible pour ce cas. Ceci dit, les essais effectués ont donné des temps d'exécution assez longs (en général supérieurs à 5s sur un PC PentiumII et parfois beaucoup plus longs), ce qui s'est avéré à l'heure actuelle peu exploitable dans notre cadre.

On peut également envisager de résoudre ce système d'équations par des méthodes plus classiques de résolution numérique de systèmes (méthode de Newton par exemple). Or, de même que l'algorithme d'optimisation utilisé pour l'ajustement, ceux-ci nécessitent une valeur initiale des variables cherchées. De manière similaire à l'initialisation utilisée pour l'ajustement de cylindre par moindres carrés, il est envisageable d'utiliser les normales estimées en chaque point pour estimer le vecteur directeur **u**. Il resterait à examiner les temps obtenus par de telles méthodes. En particulier, il convient de connaître le comportement d'un tel algorithme lorsque les 5 points sont mal disposés (cas quasi dégénérés).

Enfin, ce système peut être résolu par des méthodes spécifiques aux systèmes polynomiaux, telles que celle présentée dans [OBPT01]. Ces méthodes ne nous ont pas (encore) été rendues accessibles. Elles paraissent cependant prometteuses, en ce qu'elles réduisent la résolution du système à un problème d'extraction de valeurs-vecteurs propres, ce qui conduit potentiellement à une résolution très rapide.

Cône et tore Il paraît clair que les cas du cône et du tore, pour lesquels le nombre d'inconnues est plus élevé, sont encore plus complexes que le cas du cylindre.

Alternatives Pour le cas du cylindre que nous venons de voir, une méthode alternative consiste à introduire les normales estimées en plus des points.

Une manière d'exploiter les normales serait de réaliser un ajustement de cylindre de moindres carrés sur un très petit nombre de points (5 par exemple). Les normales permettent de produire les valeurs initiales des paramètres (voir section 4.4). Le problème est que ces valeurs initiales peuvent être très mauvaises pour 5 points.

Une autre idée consiste à séparer l'estimation de l'orientation d'une part de l'estimation de la taille et de la position d'autre part. Pour cela, on réalise une extraction conjointement dans l'image de Gauss et dans le nuage de points. La première étape consiste à extraire un plan contraint dans l'image de Gauss (quorum de deux points). On en déduit une estimation de la direction du cylindre. La seconde étape consiste à extraire un cercle 2D, une fois les points projetés dans un plan perpendiculaire à la direction trouvée à la première étape (quorum de 3 points). Cette méthode est présentée en détail dans [CGL01, CGL02]. La figure 6.17 montre le résultat sur une scène comportant un té. Bien que ceci n'ait pas été fait, ce procédé s'étend assez facilement pour le cas du cône de révolution. Il paraît plus difficile d'adapter ce principe au cas du tore.

6.6 Conclusion

Dans ce chapitre, des méthodes pour réaliser l'extraction de primitives géométriques dans un nuage de points 3D ont été présentées.

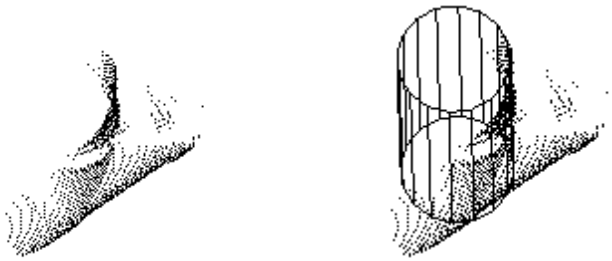


FIG. 6.17 – Extraction d'un cylindre par une méthode à base de quorum utilisant les normales

Dans le cadre de l'hypothèse H' présentée en début de chapitre, nous avons développé des méthodes d'extraction utilisant des méthodes d'ajustement de primitives par moindres carrés.

La méthode la plus simple consiste à extraire localement (autour d'un point) une primitive de type connu. Ceci a été réalisé par un procédé multi-résolution. Ce procédé donne de bons résultats en pratique, notamment dans le contexte de l'extraction de lignes de tuyauterie (section 3.2).

Des méthodes d'extraction globale d'une ou de plusieurs primitives ont été construites simplement à partir des procédés d'extraction locale développés. Ces méthodes ont également été utilisées dans les applications du chapitre 3.

Nous avons commencé à étendre ces approches au cas de primitives de type non fixé parmi : plan, sphère, cylindre, cône et tore. Ceci semble être une étape nécessaire vers la segmentation d'un nuage de points 3D en primitives géométriques. Les résultats obtenus pour l'extraction locale semblent prometteurs. Les essais effectués ont souligné l'importance de la sélection et de la validation de modèles, qui sont les thèmes abordés au chapitre 5.

Enfin, nous avons présenté une méthode qui permet de traiter le cas où H n'est pas valable. Cette méthode utilise le cadre de travaux existants [RL93]. Nous avons en particulier étendu ce formalisme à certaines de nos primitives. La construction de cette méthode pour les cas du cylindre, du cône et du tore soulève des questions difficiles.

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

Le cadre de ce travail était la segmentation d'un nuage de points 3D en primitives géométriques. Plus précisément, il s'agissait d'automatiser la segmentation-modélisation d'environnements industriels numérisés. Les lignes de tuyauterie constituent une grande partie de ce type d'environnement.

Dans ce contexte, nous avons développé des algorithmes d'extraction et de modélisation de lignes de tuyauterie à partir de nuages de points 3D. Etant donné la complexité des scènes à traiter et de la tâche à réaliser, nous avons opté pour des méthodes interactives, où l'intervention de l'utilisateur est néanmoins minime.

La méthode d'extraction de lignes de tuyauterie a été testée comparativement par des utilisateurs experts de 3Dipsos. Le résultat de ces tests a été positif, dans la mesure où les utilisateurs ont considéré que la nouvelle méthode constitue un outil pertinent pour la segmentation de lignes de tuyauterie, et moins pénible que les outils actuels. Le procédé d'extraction de lignes de tuyauterie a également donné des résultats satisfaisants sur des scènes provenant de différents scanners. Il est prévu d'intégrer cet algorithme dans le logiciel 3Dipsos ou RealWorks Survey de la société MENSI.

En ce qui concerne la segmentation-modélisation de lignes de tuyauterie, deux approches ont été développées. Ces deux approches donnent des résultats complémentaires qui nous semblent encourageants. La première approche, basée sur le même principe que la méthode d'extraction de lignes de tuyauteries, produit une description finale d'une ligne de tuyauterie dégagée. Des problèmes de stabilité nous ont conduit à développer la deuxième approche, qui tire directement parti du résultat de l'algorithme d'extraction.

L'ajustement de primitives joue un rôle central au niveau des méthodes développées. En particulier, l'une des originalités de notre approche est de faire intervenir l'ajustement de primitives géométriques au sein même des procédés d'extraction. Ceci se fait sans pré-segmentation à partir d'information bas-niveau, ce qui est rare en vision. De plus, en ce qui concerne les lignes de tuyauterie, la particularité est de faire intervenir des primitives liées et contraintes lors de l'extraction. Ces contraintes sont de simples relations géométriques issues d'informations «métier». Ceci s'avère particulièrement intéressant lorsque l'on souhaite construire des modèles CAO contraints, comme c'est le cas des modèles que l'on peut importer dans le logiciels dédiés aux lignes de tuyauterie (PDS, PDMS, ...).

En ce qui concerne les méthodes d'ajustement elles-mêmes, nous avons développé nos propres procédés d'ajustement par moindres carrés. De tels procédés ont été implémentés pour le cas de primitives libres (plan, sphère, cylindre, cône et tore) et pour le cas de primitives contraintes (cylindre-rotule, cylindre sécant, cylindre sécant de rayon fixe, tore après cylindre, cylindre après tore, tore entre cylindres). Ces méthodes d'ajustement se distinguent des méthodes existantes par la métrique utilisée et les procédés d'initialisation.

1. La métrique qui est utilisée est la distance exacte (distance euclidienne d'un point à une primitive).
2. Les procédés d'initialisation, propres à chaque primitive, utilisent des informations de type géométrie différentielle ainsi que l'ajustement d'autres primitives. Leur particularité est de produire des valeurs initiales correctes et précises dans un grand nombre de situations.

Ces caractéristiques font que les fonctions d'ajustement ont de bonnes performances en pratique, ce qui rend possible leur utilisation dans un procédé automatique d'extraction ou de segmentation.

A ce sujet, notons qu'il serait intéressant de constituer une banque de nuages de points 3D contenant chacun une primitive, avec des propriétés de bruit et de répartition des points diverses, afin d'évaluer expérimentalement les différents algorithmes d'ajustement de surfaces existants ou à venir.

D'autre part, les travaux effectués pour l'extraction de lignes de tuyauterie ont montré la nécessité de déterminer si une primitive donnée ajuste les données de manière satisfaisante ou non. Une particularité de ce travail est de s'être intéressé au problème de la validation de modèle, thème rarement traité. Ceci nous a conduit à définir des critères originaux, liés à la sélection de modèle et à la régression. Le dernier de ces critères fait intervenir un simple lissage des données à partir de la primitive. Ce critère paraît prometteur en pratique, et semble pouvoir s'appliquer à des problèmes plus généraux de reconnaissance de forme.

Enfin, l'un des points importants de cette thèse se situe au niveau des méthodes d'extraction de primitives géométriques d'un nuage de points 3D. La méthode la plus utilisée dans ce projet est un procédé multi-résolution, qui utilise l'ajustement de primitive par moindres carrés sur des voisinage emboîtés autour d'un point. Cette méthode conduit à de bons résultats lorsque le type de la primitive est connu, et est ainsi utilisée comme point de départ de l'extraction de lignes de tuyauterie. Les méthodes d'extraction globale qui ont été développées sont construites sur ce procédé d'extraction locale. De récents développements ont permis de commencer à étendre cette méthode au cas d'une primitive de type non fixé parmi : plan, sphère, cylindre, cône, et tore. Cette question semble être une étape essentielle pour réaliser la segmentation d'un nuage de points en primitives géométriques.

7.2 Perspectives

Méthode d'extraction de lignes de tuyauterie

Concernant l'algorithme actuel, plusieurs points peuvent être développés, tant du point de vue de l'interaction avec l'utilisateur que du point de vue des algorithmes.

Sélection du point initial Tout d'abord, le point initial qui doit actuellement être sélectionné par l'utilisateur peut sans doute être spécifié automatiquement, soit dans toute la scène, soit dans une zone désignée par l'utilisateur. L'idée est que cette zone n'a pas besoin d'être très précise. On peut

par exemple imaginer que l'utilisateur spécifie cette zone simplement par un rectangle en 2D. La construction du cylindre initial se fera non plus par extraction locale de cylindre autour du point sélectionné mais par une extraction de cylindre globale dans une zone raisonnable. L'utilisateur pourra ainsi sélectionner plusieurs cylindres initiaux en même temps.

Une autre idée consiste à extraire des cylindres germes dans la scène complète. Si les vues 2,5D qui constituent le nuage sont disponibles, la détection de points germes peut se faire par la recherche d'arcs de cercles sur des lignes de scan.

Optimisation des temps de calculs Les temps de calcul n'ont pas été optimisés jusqu'ici. En particulier, l'accès aux points voisins dans un nuage de points 3D est coûteux, puisqu'il nécessite une lecture de tout le nuage de points. Ceci explique en grande partie les temps observés lors du calcul du cylindre initial, en particulier pour les grandes scènes. Ces temps d'accès peuvent être franchement optimisés, ne serait-ce que par un simple partitionnement régulier de l'espace délimité par le nuage de points.

Changements de diamètre, éléments perturbateurs Les changements de diamètres ne sont pas intégrés dans l'algorithme actuel. De plus, les éléments perturbateurs rencontrés le long de la ligne (brides, poutre, etc.) ne sont pour l'instant pas identifiés. Il paraît important de pouvoir changer le diamètre de la tuyauterie, et de pouvoir mieux identifier le type des éléments perturbateurs. Ces deux thèmes reposent sur la reconnaissance de surface, et donc sur la validation de modèle.

Segmentation en primitives géométriques

Parallèlement, il paraît intéressant d'explorer les manières de segmenter la scène en terme de primitives géométriques. La méthode d'extraction locale de primitive de type non fixé développée constitue un premier pas.

L'algorithme d'extraction de lignes de tuyauterie existant pourrait à terme être intégré au sein d'un algorithme de segmentation global. Le principe pourrait être de lancer l'extraction de ligne de tuyauterie dès que l'extraction globale trouve un cylindre. Il y a dans ce cas un passage du global au local.

A l'inverse, lorsque l'extraction de ligne de tuyauterie rencontre une zone incertaine (ombre, éléments perturbateurs, ...), l'algorithme lancerait une méthode d'extraction sur une zone plus large. Il y a dans ce cas un passage du local à un niveau un peu plus global. En fin de tuyauterie, la segmentation globale pourrait reprendre sur le nuage restant.

Il peut être pertinent d'introduire des primitives «haut niveau» pour décrire les environnements industriels : poutre, bride, etc. Ces modèles peuvent être définis comme des primitives composées (ou objets paramétrés) à partir des primitives simples que nous avons considérées jusqu'ici. Il en est de même des fonctions d'ajustement de ces primitives.

A terme, si les procédés d'ajustement de ces primitives fonctionnent bien, il semble intéressant de proposer une segmentation de plus haut niveau à l'utilisateur, en demandant par exemple quels types d'éléments haut niveau sont présents dans la scène : tuyaux, poutres en I, murs, etc.

Modélisation de lignes de tuyauterie

Les deux méthodes développées dans le but de fournir un modèle final de ligne de tuyauterie semblent donner des résultats complémentaires. Il resterait à combiner les techniques de ces deux

approches afin d'obtenir une méthode stable fournissant le modèle final d'une ligne de tuyauterie.

D'autre part, pour annuler les effets de la propagation directionnelle, la modélisation d'une ligne de tuyauterie peut être complétée par un ajustement global de la séquence de primitives contraintes, considérée comme une seule primitive. En effet, il paraît intéressant d'affiner le résultat en cherchant par exemple la ligne de tuyauterie de moindres carrés. Ceci rejoint la remarque précédente concernant les primitives composées.

Remarque sur les méthodes d'ajustement à venir

Du point de vue théorique, l'algorithme de Levenberg-Marquardt présenté au chapitre 4 ne garantit pas de trouver le minimum global de la fonction coût. Comme on l'a vu, ceci souligne l'importance de produire de bonnes initialisations.

Or, les fonctions coût à optimiser dans l'ajustement de nos primitives (ou les systèmes à résoudre si l'on utilise les équations normales) sont très particulières, et s'expriment simplement en fonction des paramètres (carrés, racines carrées, etc.). La forme de ces fonctions coût est certes complexe à cause de la somme de nombreux termes, mais chaque terme est individuellement assez simple. Nous avons tiré parti de cette simplicité en utilisant un procédé d'estimation qui utilise les gradients (et une estimée de la hessienne) sous forme analytique.

A long terme, il me semble probable que des méthodes efficaces, dédiées à de telles fonctions simples, verront le jour. Déjà, des méthodes d'optimisation globale ou de résolution globale de systèmes existent, qui ne nécessitent pas d'initialisation. C'est le cas des méthodes d'analyse par intervalles, qui fournissent les optima globaux. Ces méthodes tirent parti du fait que la fonction coût s'exprime à partir de fonctions simples, donc connues. Toutefois, ces méthodes semblent encore coûteuses à l'heure actuelle, surtout pour les espaces de paramètres que nous avons à explorer ici. Cependant, les méthodes d'analyse par intervalles ne sont pas les seules méthodes qui s'intéressent à des fonction coût ou des systèmes simples. En effet, des méthodes à la fois symboliques et numériques pour la résolution de systèmes polynomiaux sont en train d'apparaître [OBPT01]. Nous verrons peut-être émerger des méthodes pour optimiser des fonctions rationnelles, puis faisant intervenir des fonctions de bases comme racine carrée, etc.

Ceci aura une incidence sur ce qui a été fait. En particulier, la partie sur les initialisations n'aura peut-être plus lieu d'être lorsque ces méthodes seront apparues (s'il n'y a plus besoin de fournir d'initialisation, ou si des moyens d'atteindre le minimum global existent). Cependant, à l'heure actuelle, il ne semble pas que l'on puisse tirer davantage parti de la forme particulière des fonctions coût. Il faut attendre des développements du côté des méthodes d'optimisation et/ou de résolution de systèmes dédiés à des fonctions simples.

Annexe A

Distance euclidienne et gradient

Notations et propriétés fréquemment utilisées dans cette annexe

Dans cette section, le gradient d'une fonction f par rapport à un vecteur paramètre $\mathbf{a} \in \mathbb{R}^p$ est noté à l'aide de l'opérateur ∇ :

$$\nabla_{\mathbf{a}} f = (\partial f/\partial a_1, \cdots, \partial f/\partial a_p)^T$$

Les vecteurs $\mathbf{u}, \mathbf{v}, \mathbf{w}$ liés aux coordonnées sphériques θ et ϕ qui sont introduits en début de cette thèse (Figure 1) sont utilisées dans cette annexe. On utilise les propriétés suivantes par rapport à la dérivation relativement aux angles θ et ϕ :

$$\begin{cases} \partial\mathbf{u}/\partial\theta &= \mathbf{v} \\ \partial\mathbf{u}/\partial\phi &= \sin\theta\mathbf{w} \\ \partial\mathbf{v}/\partial\theta &= -\mathbf{u} \\ \partial\mathbf{v}/\partial\phi &= \cos\theta\mathbf{w} \\ \partial\mathbf{w}/\partial\theta &= 0 \\ \partial\mathbf{w}/\partial\phi &= -\mathbf{w_1} \\ \partial\mathbf{w_1}/\partial\theta &= 0 \\ \partial\mathbf{w_1}/\partial\phi &= \mathbf{w} \end{cases}$$

D'autre part, les propriétés suivantes concernant le produit vectoriel et le produit scalaire ont également été fréquemment utilisées dans le calcul des distances :

$$\begin{cases} \mathbf{a}\times(\mathbf{b}\times\mathbf{c}) &= (\mathbf{a}.\mathbf{c})\mathbf{b} - (\mathbf{a}.\mathbf{b})\mathbf{c} \\ (\mathbf{a}\times\mathbf{b}).(\mathbf{c}\times\mathbf{d}) &= (\mathbf{a}.\mathbf{c})(\mathbf{b}.\mathbf{d}) - (\mathbf{a}.\mathbf{d})(\mathbf{b}.\mathbf{c}) \\ \|\mathbf{a}\times\mathbf{b}\|^2 &= \|\mathbf{a}\|^2\|\mathbf{b}\|^2 - (\mathbf{a}.\mathbf{b})^2 \end{cases}$$

Enfin, la propriété suivante est souvent utilisée pour les gradients de distance : lorsque $\mathbf{x}$ n'est pas égal au vecteur nul, on a

$$\nabla_{\mathbf{x}}(\|\mathbf{x}\|) = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Ceci découle simplement du fait que $\|\mathbf{x}\| = \sqrt{\mathbf{x}.\mathbf{x}}$ et que $\nabla_{\mathbf{x}}(\mathbf{x}.\mathbf{x}) = 2\mathbf{x}$.

A.1 Primitives usuelles

A.1.1 Droite 3D

L'ajustement de droite 3D par moindres carrés peut se faire par une méthode d'algèbre linéaire du type de celle présentée en annexe C. La description faite ici n'est donc pas utilisée pour l'ajustement de droite. Elle est cependant utile pour expliquer le cas du cylindre.

Définition Vecteur directeur **u**, et projection de **0** sur l'axe : $\mathbf{c} = a\mathbf{v} + b\mathbf{w}$ (Figure A.1).

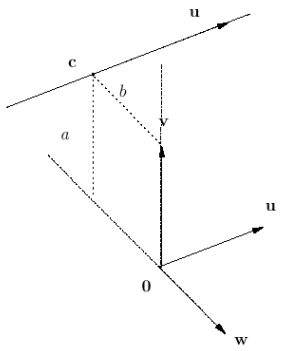


FIG. A.1 – Paramètres de la droite 3D.

Paramètres libres $\boxed{\theta, \phi, a, b}$
Dimension de l'espace de paramètres : 4

Distance euclidienne (non-signée)

$$d = \|(\mathbf{x} - \mathbf{c}) \times \mathbf{u}\|$$

soit

$$\begin{aligned} d &= \|\mathbf{x} \times \mathbf{u} + a\mathbf{w} - b\mathbf{v}\| \\ &= \sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2} \end{aligned}$$

A.1.2 Plan

De même que pour la droite, l'ajustement de plan par moindres carrés se fait de manière directe par un algorithme d'algèbre linéaire (voir annexe C). Il n'est donc pas utile de réaliser l'ajustement par une méthode non-linéaire. Cependant, la distance au plan intervient dans d'autres distances, aussi est-il utile de la faire apparaître ici.

Définition Vecteur normal **u**, et distance p de **0** au plan (Figure A.2).

Paramètres libres $\boxed{\theta, \phi, p}$
Dimension de l'espace de paramètres : 3

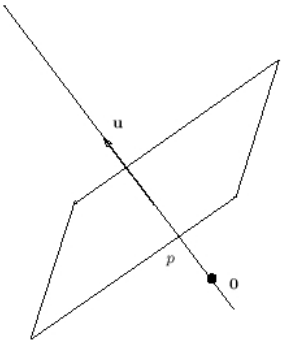


FIG. A.2 – Paramètres du plan.

Distance euclidienne signée

$$d = \mathbf{x}.\mathbf{n} - p$$

A.1.3 Sphère

Définition Centre **c** et rayon r (Figure A.3).

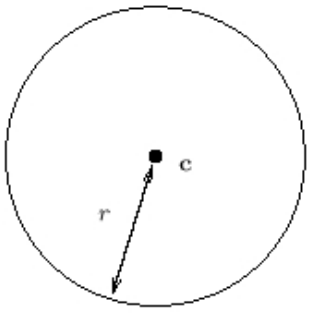


FIG. A.3 – Paramètres de la sphère.

Paramètres libres $\boxed{\mathbf{c}, r}$
Dimension de l'espace de paramètres : 4

Distance euclidienne signée

$$d = \|\mathbf{x} - \mathbf{c}\| - r$$

Dérivées premières Formellement, on a :

$$\nabla_{\mathbf{c}}(d) = -\frac{\mathbf{x} - \mathbf{c}}{\|\mathbf{x} - \mathbf{c}\|}$$

et

$$\frac{\partial d}{\partial r} = -1$$

A.1.4 Cylindre

Définition Vecteur directeur **u**, projection de **0** sur l'axe : $\mathbf{c} = a\mathbf{v} + b\mathbf{w}$, rayon r. (Figure A.4).

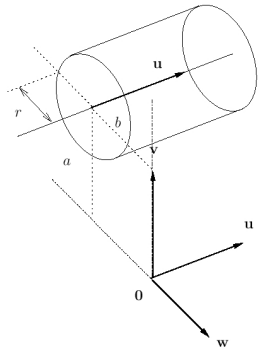


FIG. A.4 – Paramètres du cylindre.

Paramètres libres $\boxed{\theta, \phi, a, b, r}$
Dimension de l'espace de paramètres : 5

Distance euclidienne signée (se reporter au cas de la droite 3D pour plus de détails)

$$\begin{aligned} d &= d(\mathbf{x}, \text{Droite}(\mathbf{u}, \mathbf{c})) - r \\ &= \|(\mathbf{x} - \mathbf{c}) \times \mathbf{u}\| - r \end{aligned}$$

soit

$$\begin{aligned} d &= \|\mathbf{x} \times \mathbf{u} + a\mathbf{w} - b\mathbf{v}\| - r \\ &= \sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2} - r \end{aligned}$$

Dérivées premières Formellement, on a :

$$\begin{aligned} \frac{\partial d}{\partial \theta} &= \frac{-(\mathbf{x}.\mathbf{v} - a)(\mathbf{x}.\mathbf{u})}{\sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2}} \\ \frac{\partial d}{\partial \phi} &= \frac{\cos\theta(\mathbf{x}.\mathbf{v} - a)(\mathbf{x}.\mathbf{w}) - (\mathbf{x}.\mathbf{w} - b)(\mathbf{x}.\mathbf{w_1})}{\sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2}} \end{aligned}$$

$$\begin{aligned} \frac{\partial d}{\partial a} &= -\frac{\mathbf{x}.\mathbf{v} - a}{\|\mathbf{x} \times \mathbf{u} + a\mathbf{w} - b\mathbf{v}\|} \\ &= -\frac{\mathbf{x}.\mathbf{v} - a}{\sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2}} \end{aligned}$$

$$\begin{aligned} \frac{\partial d}{\partial b} &= -\frac{\mathbf{x}.\mathbf{w} - b}{\|\mathbf{x} \times \mathbf{u} + a\mathbf{w} - b\mathbf{v}\|} \\ &= -\frac{\mathbf{x}.\mathbf{w} - b}{\sqrt{(\mathbf{x}.\mathbf{v} - a)^2 + (\mathbf{x}.\mathbf{w} - b)^2}} \end{aligned}$$

$$\frac{\partial d}{\partial r} = -1$$

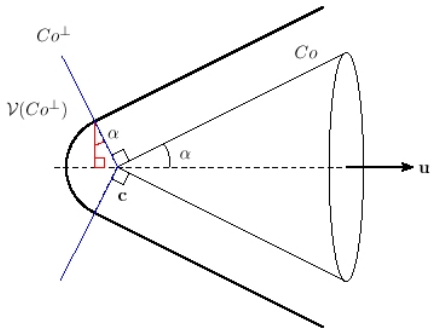


FIG. A.5 – Zone de la distance au demi-cône suivant la position du point par rapport au cône orthogonal au cône

A.1.5 Cône

Définition Le cône considéré ici est en fait un «demi-cône». On désigne ici par «demi-cône» un cône de révolution dont on ne considère qu'un seul côté en partant du sommet. Soit donc un demi-cône Co défini par un sommet $\mathbf{c}$, un vecteur unitaire $\mathbf{u}$ (pointant vers l'intérieur du cône depuis $\mathbf{c}$) et un demi-angle au sommet α ($0 \leq \alpha \leq \pi/2$).

Paramètres libres $\boxed{\theta, \phi, \mathbf{c}, \alpha}$
Dimension de l'espace de paramètres : 6

Distance euclidienne signée On introduit ici le demi-cône $Co^{\perp}$, représentant le demi-cône de sommet $\mathbf{c}$ et orthogonal à Co.

On a :

$$\mathbf{x} \in Co^{\perp} \text{ ssi } (\mathbf{x}-\mathbf{c}).\mathbf{u} = -\|\mathbf{x}-\mathbf{c}\| \cos(\pi/2-\alpha)$$

soit :

$$\mathbf{x} \in Co^{\perp} \text{ ssi } (\mathbf{x}-\mathbf{c}).\mathbf{u} = -\|\mathbf{x}-\mathbf{c}\| \sin\alpha$$

Si de plus on note $\mathcal{V}(Co^{\perp})$ le volume délimité par le cône $Co^{\perp}$, on a :

$$\mathbf{x} \in \mathcal{V}(Co^{\perp}) \text{ ssi } ((\mathbf{x}-\mathbf{c}).\mathbf{u} \leq -\|\mathbf{x}-\mathbf{c}\| \sin\alpha)$$

Ces zones sont illustrées figure A.5.

- Si $\mathbf{x} \in \mathcal{V}(Co^{\perp})$:

$$d = \|\mathbf{x}-\mathbf{c}\|$$

- Si $\mathbf{x} \notin \mathcal{V}(Co^{\perp})$:

$$\begin{aligned} d &= \cos\alpha \, (d(\mathbf{x}, \text{Axe}) - \tan\alpha((\mathbf{x}-\mathbf{c}).\mathbf{u})) \\ &= \cos\alpha \, d(\mathbf{x}, \text{Axe}) - \sin\alpha((\mathbf{x}-\mathbf{c}).\mathbf{u}) \\ &= \cos\alpha \|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| - \sin\alpha((\mathbf{x}-\mathbf{c}).\mathbf{u}) \end{aligned}$$

Voir dessins A.6 et A.5.

Remarque
Les formules utilisées ici ne sont pas symétriques en $\mathbf{u}$: on ne peut pas changer $\mathbf{u}$ en $-\mathbf{u}$.
Il est donc essentiel ici de s'assurer que $\mathbf{u}$ pointe vers l'intérieur du cône.

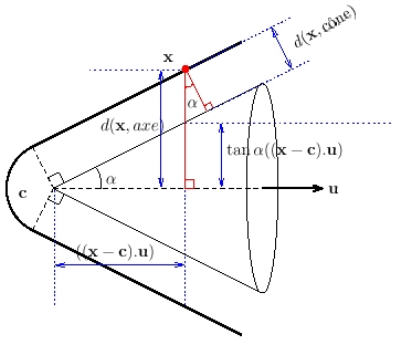


FIG. A.6 – Calcul de la distance d'un point au cône

Dérivées premières Formellement, on a :

– Si $\mathbf{x} \in \mathcal{V}(Co^{\perp})$:

$$\nabla_{\mathbf{c}}(d) = -\frac{\mathbf{x}-\mathbf{c}}{\|\mathbf{x}-\mathbf{c}\|}$$

$$\frac{\partial d}{\partial \theta} = \frac{\partial d}{\partial \phi} = \frac{\partial d}{\partial \alpha} = 0$$

– Si $\mathbf{x} \notin \mathcal{V}(Co^{\perp})$:

$$\frac{\partial d}{\partial \theta} = -\left[\frac{\cos\alpha(\mathbf{x}-\mathbf{c}).\mathbf{u}}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|} + \sin\alpha\right]((\mathbf{x}-\mathbf{c}).\mathbf{v})$$

$$\frac{\partial d}{\partial \phi} = -\sin\theta\left[\frac{\cos\alpha(\mathbf{x}-\mathbf{c}).\mathbf{u}}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|} + \sin\alpha\right]((\mathbf{x}-\mathbf{c}).\mathbf{w})$$

$$\begin{aligned}
\nabla_{\mathbf{c}}(d) &= \frac{\cos\alpha}{2\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}\nabla_{\mathbf{c}}[(\mathbf{x}-\mathbf{c}).(\mathbf{x}-\mathbf{c}) - ((\mathbf{x}-\mathbf{c}).\mathbf{u})^2] + \sin\alpha\mathbf{u} \\
&= \frac{\cos\alpha}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}[-(\mathbf{x}-\mathbf{c}) + ((\mathbf{x}-\mathbf{c}).\mathbf{u})\mathbf{u}] + \sin\alpha\mathbf{u} \\
&= -\frac{\cos\alpha}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}(\mathbf{x}-\mathbf{c}) + \left[\frac{\cos\alpha}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}((\mathbf{x}-\mathbf{c}).\mathbf{u}) + \sin\alpha\right]\mathbf{u}
\end{aligned}$$

$$\frac{\partial d}{\partial \alpha} = -\sin\alpha\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\| - \cos\alpha((\mathbf{x}-\mathbf{c}).\mathbf{u})$$

A.1.6 Cercle 3D

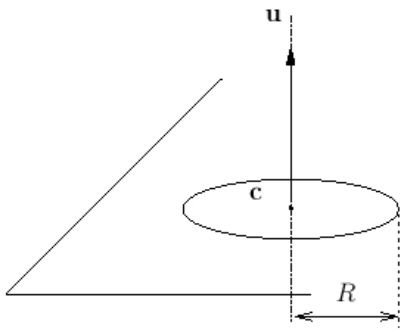


FIG. A.7 – Cercle 3D

Définition Cercle de centre **c**, de rayon R, dessiné dans le plan $\mathbf{c} + (\mathbf{u})^{\perp}$, le plan orthogonal à **u** passant par **c** (Figure A.7).

Paramètres libres $\boxed{\mathbf{c}, \theta, \phi, R}$
Dimension de l'espace de paramètres : 6

Distance euclidienne (non signée)

$$d = \sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}$$

Justification : d'après le théorème de Pythagore :

$$\begin{aligned} d^2 &= d^2(\mathbf{x}, \mathrm{Cyl}(\mathbf{c}, \mathbf{u}, R)) + ((\mathbf{x}-\mathbf{c}).\mathbf{u})^2 \\ &= (\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| - R)^2 + ((\mathbf{x}-\mathbf{c}).\mathbf{u})^2 \\ &= R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| + \|\mathbf{x}-\mathbf{c}\|^2 \\ &= R^2 - 2R\sqrt{\|\mathbf{x}-\mathbf{c}\|^2 - ((\mathbf{x}-\mathbf{c}).\mathbf{u})^2} + \|\mathbf{x}-\mathbf{c}\|^2 \end{aligned}$$

car $(\mathbf{a} \times \mathbf{b}).(\mathbf{a} \times \mathbf{b}) = (\mathbf{a}.\mathbf{a})(\mathbf{b}.\mathbf{b}) - (\mathbf{a}.\mathbf{b})^2$.

Dérivées premières

$$\begin{aligned} \nabla_{\mathbf{c}}(d) &= \frac{1}{d}\left[-(\mathbf{x}-\mathbf{c}) + R\frac{(\mathbf{x}-\mathbf{c}) - ((\mathbf{x}-\mathbf{c}).\mathbf{u})\mathbf{u}}{\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}\right] \\ &= \frac{-(\mathbf{x}-\mathbf{c}) + R\frac{(\mathbf{x}-\mathbf{c}) - ((\mathbf{x}-\mathbf{c}).\mathbf{u})\mathbf{u}}{\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}}{\sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}} \end{aligned}$$

$$\begin{aligned}\frac{\partial d}{\partial \theta} &= \frac{1}{d}\left[R\frac{((\mathbf{x}-\mathbf{c}).\mathbf{u})((\mathbf{x}-\mathbf{c}).\mathbf{v})}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}\right] \\ &= \frac{R((\mathbf{x}-\mathbf{c}).\mathbf{u})((\mathbf{x}-\mathbf{c}).\mathbf{v})}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|\sqrt{\|\mathbf{x}-\mathbf{c}\|^2+R^2-2R\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}}\end{aligned}$$

$$\begin{aligned}\frac{\partial d}{\partial \phi} &= \frac{1}{d}\left[R\sin\theta\frac{((\mathbf{x}-\mathbf{c}).\mathbf{u})((\mathbf{x}-\mathbf{c}).\mathbf{w})}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}\right] \\ &= \frac{R\sin\theta((\mathbf{x}-\mathbf{c}).\mathbf{u})((\mathbf{x}-\mathbf{c}).\mathbf{w})}{\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|\sqrt{\|\mathbf{x}-\mathbf{c}\|^2+R^2-2R\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}}\end{aligned}$$

$$\begin{aligned}\frac{\partial d}{\partial R} &= \frac{1}{d}\left[R-\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|\right] \\ &= \frac{R-\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}{\sqrt{\|\mathbf{x}-\mathbf{c}\|^2+R^2-2R\|(\mathbf{x}-\mathbf{c})\times\mathbf{u}\|}}\end{aligned}$$

A.1.7 Tore

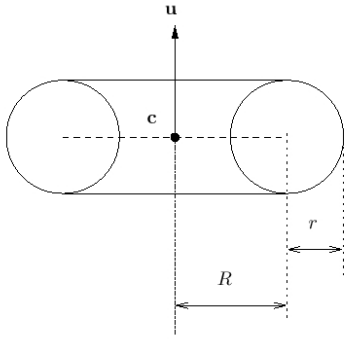


FIG. A.8 – Paramètres du tore

Définition Vecteur normal **u**, centre **c**, grand rayon R, petit rayon r (Figure A.8).

Paramètres libres $\boxed{\theta, \phi, \mathbf{c}, R, r}$
Dimension de l'espace de paramètres : 7

Distance euclidienne signée

$$\begin{aligned} d &= d(\mathbf{x}, \text{Cercle}) - r \\ &= \sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|} - r \end{aligned}$$

Dérivées premières (se reporter au cas du cercle 3D pour plus de détails)

$$\nabla_{\mathbf{c}}(d) = \frac{-(\mathbf{x}-\mathbf{c}) + R\frac{(\mathbf{x}-\mathbf{c}) - ((\mathbf{x}-\mathbf{c}).\mathbf{u})\mathbf{u}}{\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}}{\sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}}$$

$$\frac{\partial d}{\partial \theta} = \frac{R((\mathbf{x}-\mathbf{c}).\mathbf{u})((\mathbf{x}-\mathbf{c}).\mathbf{v})}{\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| \sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}}$$

$$\frac{\partial d}{\partial \phi} = \frac{R\sin\theta((\mathbf{x}-\mathbf{c}).\mathbf{u})((\mathbf{x}-\mathbf{c}).\mathbf{w})}{\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| \sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}}$$

$$\frac{\partial d}{\partial R} = \frac{R - \|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}{\sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\|}}$$

$$\frac{\partial d}{\partial r} = -1$$

A.2 Primitives contraintes

A.2.1 «Cylindre sécant»

Définition Un «cylindre sécant» est un cylindre de révolution dont l'axe est concourant avec un premier cylindre (Figure A.9).

Soit un premier cylindre $(\mathbf{p_0}, \mathbf{u_0}, r_0)$ de rayon r_0 et d'axe Δ_0, défini par un point $\mathbf{p_0}$ et un vecteur unitaire $\mathbf{u_0}$ ($\Delta_0 = \mathbf{p_0} + \mathbb{R}\mathbf{u_0}$). On définit ici le deuxième cylindre $(\mathbf{p}, \mathbf{u}, r)$ d'axe concourant avec le premier. On choisit donc un point $\mathbf{p}$ contraint à se trouver sur le premier axe. On définit donc $\mathbf{p}$ comme étant le point d'intersection des deux axes :

$$\mathbf{p} = \mathbf{p_0} + \mu\mathbf{u_0}$$

avec $\mu \in \mathbb{R}$.

Si on a $\mathbf{u} = \mathbf{u_0}$, on prendra $\mu = 0$, c'est-à-dire $\mathbf{p} = \mathbf{p_0}$ par convention.

On note $\mathbf{u}$ le vecteur directeur de l'axe du cylindre sécant.

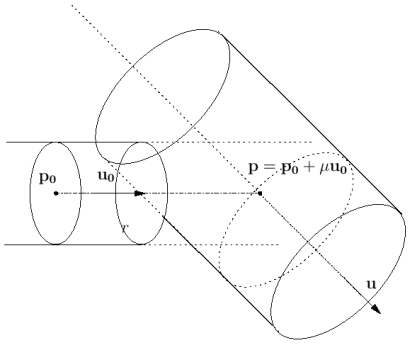


FIG. A.9 – «Cylindre sécant»

Paramètres libres $\boxed{\mu, \theta, \phi, r}$
Dimension de l'espace de paramètres : 4

Distance euclidienne signée

$$\begin{aligned} d &= d(\mathbf{x}, \Delta) - r \\ &= \|(\mathbf{x}-\mathbf{p}) \times \mathbf{u}\| - r \\ &= \|(\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}) \times \mathbf{u}\| - r \\ &= \sqrt{(\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}).(\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}) - ((\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}).\mathbf{u})^2} - r \end{aligned}$$

Dérivées premières

$$\begin{aligned} \frac{\partial d}{\partial \mu} &= \frac{1}{2\|(\mathbf{x}-\mathbf{p}) \times \mathbf{u}\|} \left[2(\mathbf{x}-\mathbf{p}).\frac{\partial(\mathbf{x}-\mathbf{p})}{\partial \mu} - 2((\mathbf{x}-\mathbf{p}).\mathbf{u})\frac{\partial((\mathbf{x}-\mathbf{p}).\mathbf{u})}{\partial \mu} \right] \\ &= \frac{1}{\|(\mathbf{x}-\mathbf{p}) \times \mathbf{u}\|} \left[(\mathbf{x}-\mathbf{p}).(-\mathbf{u_0}) - ((\mathbf{x}-\mathbf{p}).\mathbf{u})(-\mathbf{u_0}.\mathbf{u}) \right] \\ &= \frac{-(\mathbf{x}-\mathbf{p}).\mathbf{u_0} + ((\mathbf{x}-\mathbf{p}).\mathbf{u})(\mathbf{u_0}.\mathbf{u})}{\|(\mathbf{x}-\mathbf{p}) \times \mathbf{u}\|} \end{aligned}$$

$$\begin{aligned}\frac{\partial d}{\partial \phi} &= \frac{1}{2\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\left[-2((\mathbf{x}-\mathbf{p}).\mathbf{u})\frac{\partial((\mathbf{x}-\mathbf{p}).\mathbf{u})}{\partial\phi}\right] \\ &= -\frac{\sin\theta((\mathbf{x}-\mathbf{p}).\mathbf{u})((\mathbf{x}-\mathbf{p}).\mathbf{w})}{\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\end{aligned}$$

$$\begin{aligned}\frac{\partial d}{\partial \theta} &= \frac{1}{2\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\left[-2((\mathbf{x}-\mathbf{p}).\mathbf{u})\frac{\partial((\mathbf{x}-\mathbf{p}).\mathbf{u})}{\partial\theta}\right] \\ &= -\frac{((\mathbf{x}-\mathbf{p}).\mathbf{u})((\mathbf{x}-\mathbf{p}).\mathbf{v})}{\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\end{aligned}$$

$$\frac{\partial d}{\partial r} = -1$$

A.2.2 «Cylindre sécant de rayon fixé»

Définition Un «cylindre sécant de rayon fixé» est un «cylindre sécant» dont le rayon est fixé égal à celui du cylindre précédent (Figure A.10). Les paramètres sont les mêmes, sauf r en moins.

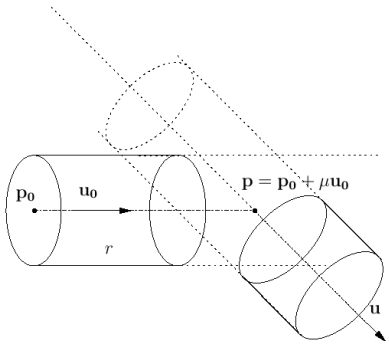


FIG. A.10 – «Cylindre sécant de rayon fixé»

Paramètres libres $\boxed{\mu, \theta, \phi}$
Dimension de l'espace de paramètres : 3

Distance euclidienne signée

$$\begin{aligned} d &= d(\mathbf{x},\Delta) - r \\ &= \|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\| - r \\ &= \|(\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0})\times\mathbf{u}\| - r \\ &= \sqrt{(\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}).(\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}) - ((\mathbf{x}-\mathbf{p_0}-\mu\mathbf{u_0}).\mathbf{u})^2} - r \end{aligned}$$

Dérivées premières

$$\begin{aligned} \frac{\partial d}{\partial \mu} &= \frac{1}{2\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\left[2(\mathbf{x}-\mathbf{p}).\frac{\partial(\mathbf{x}-\mathbf{p})}{\partial\mu} - 2((\mathbf{x}-\mathbf{p}).\mathbf{u})\frac{\partial((\mathbf{x}-\mathbf{p}).\mathbf{u})}{\partial\mu}\right] \\ &= \frac{1}{\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\left[(\mathbf{x}-\mathbf{p}).(-\mathbf{u_0}) - ((\mathbf{x}-\mathbf{p}).\mathbf{u})(-\mathbf{u_0}.\mathbf{u})\right] \\ &= \frac{-(\mathbf{x}-\mathbf{p}).\mathbf{u_0} + ((\mathbf{x}-\mathbf{p}).\mathbf{u})(\mathbf{u_0}.\mathbf{u})}{\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|} \end{aligned}$$

$$\begin{aligned} \frac{\partial d}{\partial \phi} &= \frac{1}{2\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\left[-2((\mathbf{x}-\mathbf{p}).\mathbf{u})\frac{\partial((\mathbf{x}-\mathbf{p}).\mathbf{u})}{\partial\phi}\right] \\ &= -\frac{\sin\theta((\mathbf{x}-\mathbf{p}).\mathbf{u})((\mathbf{x}-\mathbf{p}).\mathbf{w})}{\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|} \end{aligned}$$

$$\begin{aligned} \frac{\partial d}{\partial \theta} &= \frac{1}{2\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|}\left[-2((\mathbf{x}-\mathbf{p}).\mathbf{u})\frac{\partial((\mathbf{x}-\mathbf{p}).\mathbf{u})}{\partial\theta}\right] \\ &= -\frac{((\mathbf{x}-\mathbf{p}).\mathbf{u})((\mathbf{x}-\mathbf{p}).\mathbf{v})}{\|(\mathbf{x}-\mathbf{p})\times\mathbf{u}\|} \end{aligned}$$

A.2.3 «Cylindre-rotule»

Définition Etant donné un premier cylindre dont on connaît les extrémités, un «cylindre-rotule» est un cylindre sécant au premier dont l'axe passe par l'extrémité du premier cylindre (Figure A.11).

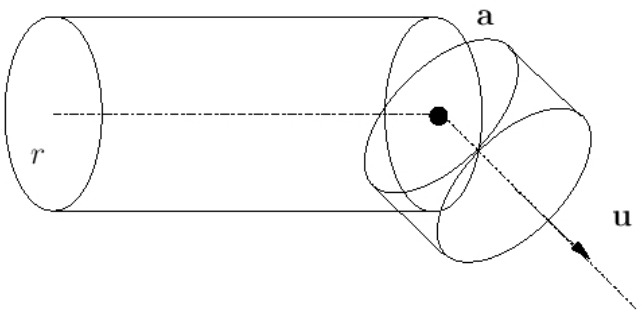


FIG. A.11 – Paramètres du «cylindre-rotule».

Paramètres libres $\boxed{\theta, \phi}$
Dimension de l'espace de paramètres : 2

Distance euclidienne signée

$$d = \|(\mathbf{x}-\mathbf{a}) \times \mathbf{u}\| - r$$

Dérivées premières

$$\frac{\partial d}{\partial \phi} = -\frac{\sin\theta((\mathbf{x}-\mathbf{a}).\mathbf{u})((\mathbf{x}-\mathbf{a}).\mathbf{w})}{\|(\mathbf{x}-\mathbf{a}) \times \mathbf{u}\|}$$

$$\frac{\partial d}{\partial \theta} = -\frac{((\mathbf{x}-\mathbf{a}).\mathbf{u})((\mathbf{x}-\mathbf{a}).\mathbf{v})}{\|(\mathbf{x}-\mathbf{a}) \times \mathbf{u}\|}$$

A.2.4 «Cylindre après tore»

Définition L'unique paramètre d'un «Cylindre après tore» est l'angle d'ouverture α du tore à partir de sa première base (Figure A.12).

on suppose que l'on a déjà déterminé les données caractérisant le tore, à savoir :

- un centre $\mathbf{c}$
- une première extrémité $\mathbf{a_0}$ (sur le cercle directeur)
- un vecteur directeur unitaire $\mathbf{u_0}$ en $\mathbf{a_0}$
- un vecteur normal unitaire $\mathbf{n}$. Par convention on suppose que $\mathbf{n}$ est orienté tel que :

$$(\mathbf{a_0}-\mathbf{c})\times\mathbf{u_0} = R\mathbf{n}$$

- Un grand rayon R et un petit rayon r

La deuxième extrémité $\mathbf{a}$ de la portion de tore se déduit de α par :

$$\mathbf{a} = \mathbf{c} + \mathbf{R}_{-\mathbf{n},\alpha}\cdot(\mathbf{a_0}-\mathbf{c})$$

où $\mathbf{R}_{\mathbf{n},-\alpha}$ désigne la rotation vectorielle d'angle $-\alpha$ autour du vecteur $\mathbf{n}$.

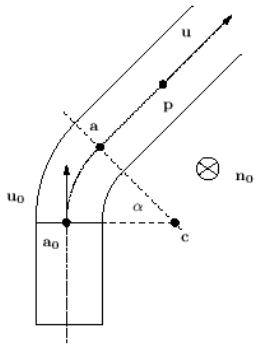


FIG. A.12 – «Cylindre après tore».

Paramètre libre $\boxed{\alpha}$

Dimension de l'espace de paramètres : 1

Distance euclidienne signée Le vecteur directeur du cylindre est

$$\mathbf{u} = \frac{\mathbf{a}-\mathbf{c}}{R}\times\mathbf{n}$$

Or

$$d = \|(\mathbf{x}-\mathbf{a})\times\mathbf{u}\| - r$$

donc

$$\begin{aligned} d &= \left\|(\mathbf{x}-\mathbf{a})\times\left(\frac{(\mathbf{a}-\mathbf{c})\times\mathbf{n}}{R}\right)\right\| - r \\ &= \frac{1}{R}\|((\mathbf{x}-\mathbf{a})\cdot\mathbf{n})(\mathbf{a}-\mathbf{c}) - ((\mathbf{x}-\mathbf{a})\cdot(\mathbf{a}-\mathbf{c}))\mathbf{n}\| - r \\ &= \sqrt{((\mathbf{x}-\mathbf{a})\cdot\mathbf{n})^2 + \frac{1}{R^2}((\mathbf{x}-\mathbf{a})\cdot(\mathbf{a}-\mathbf{c}))^2} - r \end{aligned}$$

car $(\mathbf{a}-\mathbf{c}) \perp \mathbf{n}$.

Dérivées premières

$$\frac{\partial d}{\partial \alpha} = \left(\frac{-((\mathbf{x}-\mathbf{a})\cdot\mathbf{n})\mathbf{n} + \frac{1}{R^2}((\mathbf{x}-\mathbf{a})\cdot(\mathbf{a}-\mathbf{c}))(\mathbf{x}+\mathbf{c}-\mathbf{2a})}{\sqrt{((\mathbf{x}-\mathbf{a})\cdot\mathbf{n})^2 + \frac{1}{R^2}\left((\mathbf{x}-\mathbf{a})\cdot(\mathbf{a}-\mathbf{c})\right)^2}} \right) \cdot \frac{\partial \mathbf{a}}{\partial \alpha}$$

avec

$$\frac{\partial \mathbf{a}}{\partial \alpha} = R_{-\mathbf{n},\alpha+\frac{\pi}{2}} \cdot (\mathbf{a_0}-\mathbf{c})$$

car

$$\frac{\partial}{\partial \alpha}\mathrm{Mat}_{(u,v,w)}(\mathbf{R}_{\mathbf{u},\alpha}) = \mathrm{Mat}_{(u,v,w)}(\mathbf{R}_{\mathbf{u},\alpha+\frac{\pi}{2}})$$

où $\mathbf{R}_{\mathbf{u},\alpha}$ désigne la rotation vectorielle autour du vecteur $\mathbf{u}$ et d'angle α, et où $\mathrm{Mat}_{(u,v,w)}(\mathbf{R})$ désigne la matrice de la rotation vectorielle $\mathbf{R}$ dans la base $(\mathbf{u},\mathbf{v},\mathbf{w})$ construite à partir du vecteur $\mathbf{u}$ (voir la partie notations au début de ce document).

A.2.5 «Tore après cylindre»

Définition Un «Tore après cylindre» est un tore circulaire joignant un cylindre de même (petit) rayon. Le cylindre est défini par un axe Δ_0, lui-même défini par un point $\mathbf{p}_0$ et un vecteur directeur unitaire $\mathbf{u}_0$ (dirigé vers le tore), et un rayon r (Figure A.13). R dénote le grand rayon du tore (on suppose que $R > r$), $\mathbf{n}$ le vecteur normal du tore et $\mathbf{x}$ un point courant.

Ici les coordonnées du centre du tore $\mathbf{c}$ sont prises comme seuls paramètres indépendants. En effet, les autres paramètres s'en déduisent :

$$R = \|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|$$

et

$$\mathbf{n} = \frac{(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}}{\|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|}$$

(la convention d'orientation de $\mathbf{n}$ est telle que : $((\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}).\mathbf{n} > 0$, ce qui défini le sens de propagation comme le sens direct dans le plan $\mathbf{c} + \mathbf{n}^{\perp}$).

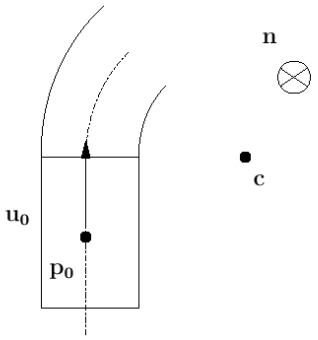


FIG. A.13 – «Tore après cylindre»

Paramètre libre $\boxed{\mathbf{c}}$
Dimension de l'espace de paramètres : 3

Distance euclidienne signée

$$d(\mathbf{x}, tore) = d(\mathbf{x}, \text{Cercle}) - r$$

et

$$\begin{aligned} d(\mathbf{x}, \text{Cercle}) &= \sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x} - \mathbf{c}) \times \mathbf{n}\|} \\ &= \sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + \|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|^2 - 2\|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|\|(\mathbf{x} - \mathbf{c}) \times \mathbf{n}\|} \\ &= \sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + \|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|^2 - 2\|(\mathbf{x} - \mathbf{c}) \times ((\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0})\|} \\ &= \sqrt{T1 + T2 + T3} \end{aligned}$$

avec

$$\begin{cases} T1 = \|\mathbf{x} - \mathbf{c}\|^2 = (\mathbf{x} - \mathbf{c}).(\mathbf{x} - \mathbf{c}) \\ T2 = R^2 = \|(\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0}\|^2 = (\mathbf{p_0} - \mathbf{c}).(\mathbf{p_0} - \mathbf{c}) - ((\mathbf{p_0} - \mathbf{c}).\mathbf{u_0})^2 \\ T3 = -2\|(\mathbf{x} - \mathbf{c}) \times ((\mathbf{p_0} - \mathbf{c}) \times \mathbf{u_0})\| \end{cases}$$

Dérivées premières

$$\nabla_{\mathbf{c}} T1 = -2(\mathbf{x}-\mathbf{c})$$
$$\nabla_{\mathbf{c}} T2 = -2((\mathbf{p_0}-\mathbf{c}) - ((\mathbf{p_0}-\mathbf{c}).\mathbf{u_0})\mathbf{u_0})$$

$$(\mathbf{x}-\mathbf{c}) \times ((\mathbf{p_0}-\mathbf{c}) \times \mathbf{u_0}) = ((\mathbf{x}-\mathbf{c}).\mathbf{u_0})(\mathbf{p_0}-\mathbf{c}) - ((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))\mathbf{u_0}$$

donc

$$\begin{aligned} \|(\mathbf{x}-\mathbf{c}) \times ((\mathbf{p_0}-\mathbf{c}) \times \mathbf{u_0})\|^2 &= ((\mathbf{x}-\mathbf{c}).\mathbf{u_0})^2 \|\mathbf{p_0}-\mathbf{c}\|^2 \\ &\quad +((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))^2 \\ &\quad -\mathbf{2}((\mathbf{x}-\mathbf{c}).\mathbf{u_0})((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))((\mathbf{p_0}-\mathbf{c}).\mathbf{u_0}) \\ &= T3a + T3b + T3c \end{aligned}$$

(car $\|u_0\|^2 = 1$) avec

$$T3a = ((\mathbf{x}-\mathbf{c}).\mathbf{u_0})^2 \|\mathbf{p_0}-\mathbf{c}\|^2$$
$$T3b = ((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))^2$$
$$T3c = -\mathbf{2}((\mathbf{x}-\mathbf{c}).\mathbf{u_0})((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))((\mathbf{p_0}-\mathbf{c}).\mathbf{u_0})$$

$$\nabla_{\mathbf{c}} T3a = -\mathbf{2}[((\mathbf{x}-\mathbf{c}).\mathbf{u_0})\|\mathbf{p_0}-\mathbf{c}\|^2\mathbf{u_0} + ((\mathbf{x}-\mathbf{c}).\mathbf{u_0})^2(\mathbf{p_0}-\mathbf{c})]$$
$$\frac{\partial T3b}{\partial c_x} = 2((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))\frac{\partial}{\partial c_x}[(x-c_x)(p_{0x}-c_x)]$$
$$\nabla_{\mathbf{c}} T3b = 2((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))[\mathbf{2c}-\mathbf{p_0}-\mathbf{x}]$$

$$\begin{aligned} \frac{\partial T3c}{\partial c_x} &= -2[(-u_{0x})((x-c).(p_0-c))((p_0-c).u_0) \\ &\quad +((x-c).u_0)(2c_x-x-p_x)((p_0-c).u_0) \\ &\quad +((x-c).u_0)((x-c).(p_0-c))(-u_{0x})] \end{aligned}$$

d'où

$$\begin{aligned} \nabla_{\mathbf{c}} T3c &= 2((\mathbf{x}-\mathbf{c}).(\mathbf{p_0}-\mathbf{c}))[((\mathbf{p_0}-\mathbf{c}).\mathbf{u_0}) + ((\mathbf{x}-\mathbf{c}).\mathbf{u_0})]\mathbf{u_0} \\ &\quad +((\mathbf{x}-\mathbf{c}).\mathbf{u_0})((\mathbf{p_0}-\mathbf{c}).\mathbf{u_0})(\mathbf{x}+\mathbf{p}-\mathbf{2c}) \end{aligned}$$

$$\nabla T3 = \tfrac{2}{T3}[\nabla T3a + \nabla T3b + \nabla T3c]$$

et enfin

$$\nabla_{\mathbf{c}} d(\mathbf{x}, \text{Cercle}) = \frac{\nabla T1 + \nabla T2 + \nabla T3}{2\sqrt{T1+T2+T3}}$$

A.2.6 «Tore entre cylindres»

Le but de cette primitive est d'améliorer la qualité du tore dans une séquence «cylindre-tore-cylindre» déjà existante, en laissant fixes les deux cylindre (l'ajustement de cylindre est moins délicat que l'ajustement de tore). Comme nous le verrons, dans le cadre de ces contraintes, le seul paramètre libre est le grand rayon du tore.

Définition Un «Tore entre cylindres» est un tore joignant un premier cylindre $(\mathbf{u_0}, \mathbf{p_0}, r)$, et un «cylindre sécant» avec le premier cylindre $(\mathbf{u_1}, \mathbf{p_1}, r)$ (Figure A.14). On suppose que $\mathbf{p_1}$ est le point de concours des deux axes et que $\mathbf{u_0}$ pointe vers $\mathbf{p_1}$ et $\mathbf{u_1}$ pointe dans le sens opposé à $\mathbf{p_1}$.

> Remarque
> Ici seul le cas d'axes sécants est traité. Comment traiter le cas d'axes parallèles (i.e. demi-tour) ?

La normale au plan du cercle directeur est complètement déterminée par les axes des deux cylindres :

$$\mathbf{n} = \frac{\mathbf{u_0} \times \mathbf{u_1}}{\|\mathbf{u_0} \times \mathbf{u_1}\|}$$

(convention d'orientation telle que la propagation s'effectue dans le sens direct dans le plan $\mathbf{p_0} + \mathbf{n}^{\perp}$).

Exprimons $\mathbf{c}$ et $\mathbf{n}$ en fonction de R et des paramètres des 2 cylindres. Si $\mathbf{p_1}$ est le point d'intersection entre les 2 cylindres sécants et $\alpha = \arccos(\mathbf{u_0}.\mathbf{u_1})$ l'angle entre leurs axes ($0 \leq \alpha < \pi$), on a :

$$\mathbf{c} = \mathbf{p_1} + \frac{R}{\cos(\alpha/2)} \frac{\mathbf{u_1} - \mathbf{u_0}}{\|\mathbf{u_1} - \mathbf{u_0}\|}$$

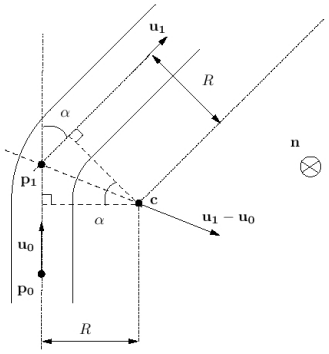


FIG. A.14 – Tore entre cylindres

Paramètre libre $\boxed{R}$
Dimension de l'espace de paramètres : 1

Distance euclidienne signée La distance d'un point au tore s'écrit :

$$d = \sqrt{\|\mathbf{x} - \mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x} - \mathbf{c}) \times \mathbf{n}\|} - r$$

où $\mathbf{c}$, $\mathbf{n}$ se déduisent des formules ci avant.

Dérivées premières Seul le centre **c** dépend de la variable R ici. D'où

$$\frac{\partial d}{\partial R} = \frac{\frac{\partial}{\partial R}\|\mathbf{x}-\mathbf{c}\|^2 + 2R - 2\|(\mathbf{x}-\mathbf{c})\times\mathbf{n}\| - 2R\frac{\partial}{\partial R}\|(\mathbf{x}-\mathbf{c})\times\mathbf{n}\|}{2\sqrt{\|\mathbf{x}-\mathbf{c}\|^2 + R^2 - 2R\|(\mathbf{x}-\mathbf{c})\times\mathbf{n}\|}}$$

Or

$$\frac{\partial \mathbf{c}}{\partial R} = \frac{1}{\cos(\alpha/2)}\frac{\mathbf{u_1}-\mathbf{u_0}}{\|\mathbf{u_1}-\mathbf{u_0}\|}$$

$$\begin{aligned}\frac{\partial\|\mathbf{x}-\mathbf{c}\|^2}{\partial R} &= \frac{\partial}{\partial R}((\mathbf{x}-\mathbf{c})\cdot(\mathbf{x}-\mathbf{c})) \\ &= -2\frac{\partial \mathbf{c}}{\partial R}\cdot(\mathbf{x}-\mathbf{c}) \\ &= -\frac{(\mathbf{x}-\mathbf{c}).(\mathbf{u_1}-\mathbf{u_0})}{\cos(\alpha/2)\|\mathbf{u_1}-\mathbf{u_0}\|}\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial R}\|(\mathbf{x}-\mathbf{c})\times\mathbf{n}\| &= \frac{\frac{\partial}{\partial R}[((\mathbf{x}-\mathbf{c})\times\mathbf{n})\cdot((\mathbf{x}-\mathbf{c})\times\mathbf{n})]}{2\|(\mathbf{x}-\mathbf{c})\times\mathbf{n}\|} \\ &= -\frac{((\mathbf{x}-\mathbf{c})\times\mathbf{n})\cdot((\mathbf{u_1}-\mathbf{u_0})\times\mathbf{n})}{\cos(\alpha/2)\|(\mathbf{x}-\mathbf{c})\times\mathbf{n}\|\|\mathbf{u_1}-\mathbf{u_0}\|}\end{aligned}$$

Annexe B

Notions de normale et de courbures d'une surface, estimation sur un nuage de points

Soit une surface définie par deux paramètres u et v :

$$\mathbf{x}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$$

On note :

$$\begin{cases} \mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u} \\ \mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v} \end{cases}$$

ainsi que

$$\begin{cases} \mathbf{x}_{uu} = \frac{\partial^2 \mathbf{x}}{\partial^2 u} \\ \mathbf{x}_{vv} = \frac{\partial^2 \mathbf{x}}{\partial^2 v} \\ \mathbf{x}_{uv} = \frac{\partial^2 \mathbf{x}}{\partial u \partial v} \end{cases}$$

B.1 Normale

Définition

La normale au point (u,v) de la surface est définie au signe près par l'expression (le signe peut par exemple être choisi en fonction de celui de la courbure maximale. Voir plus loin) :

$$\mathbf{n}(u,v) = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$$

Remarque
L'aire de la surface est donnée par

$$\int_U \|\mathbf{x_u} \times \mathbf{x_v}\| du dv$$

Valeurs pour quelques primitives géométriques

Primitive	Point $\mathbf{x}$	Normale $\mathbf{n}$
Plan	$\mathbf{x}$	constante
Sphère	$\mathbf{x}$	$\mathbf{n} = \frac{\mathbf{x}-\mathbf{c}}{R}$
Cylindre	$\mathbf{x}(z,\phi) = \begin{pmatrix} r\cos\phi \\ r\sin\phi \\ z \end{pmatrix}$ $(\star)$	$\mathbf{n}(z,\phi) = \begin{pmatrix} \cos\phi \\ \sin\phi \\ 0 \end{pmatrix}$
Cône	$\mathbf{x}(z,\phi) = \begin{pmatrix} z\tan\alpha\cos\phi \\ z\tan\alpha\sin\phi \\ z \end{pmatrix}$ $(\star)$	$\mathbf{n}(z,\phi) = \begin{pmatrix} \cos\alpha\cos\phi \\ \cos\alpha\sin\phi \\ -\sin\alpha \end{pmatrix}$
Tore	$\mathbf{x}(u,v) = \begin{pmatrix} (R+r\cos v)\cos u \\ (R+r\cos v)\sin u \\ r\sin v \end{pmatrix}$ $(\star)$	$\mathbf{n}(u,v) = \begin{pmatrix} \cos v\cos u \\ \cos v\sin u \\ \sin v \end{pmatrix}$

($(\star)$: dans un repère propre)

B.2 Courbures

Définition

On introduit

$$\begin{cases} E = \|\mathbf{x}_u\|^2 \\ F = \mathbf{x}_u.\mathbf{x}_v \\ G = \|\mathbf{x}_v\|^2 \end{cases}$$

et

$$\begin{cases} e = \mathbf{n}.\mathbf{x}_{uu} \\ f = \mathbf{n}.\mathbf{x}_{uv} \\ g = \mathbf{n}.\mathbf{x}_{vv} \end{cases}$$

La courbure de Gauss est définie par

$$K = \frac{eg - f^2}{EG - F^2}$$

et la courbure moyenne par

$$H = \frac{eG + Eg - 2fF}{2(EG - F^2)}$$

Le lien avec les courbures min et max est donné par

$$\begin{cases} K = K_{min}K_{max} \\ H = \frac{1}{2}(K_{min} + K_{max}) \end{cases}$$

Ceci revient à dire que les courbures principales K_{min} et K_{max} sont les deux solutions de l'équation $X^2 - 2HX + K = 0$, c'est-à-dire que

$$K_{min}, K_{max} = H \pm \sqrt{H^2 - K}$$

où K_{min} est la solution de valeur absolue minimum, et K_{max} est la solution de valeur absolue maximum.

Remarque
K est appelée courbure «totale» ou courbure de Gauss, et peut être interprétée comme la limite du rapport de l'aire sur la surface de Gauss et l'aire de la surface lorsque cette dernière tend vers 0 [Die78]. Voir aussi [Bar94] pour une définition des courbures d'une surface et plusieurs interprétations de la courbure de Gauss.

Remarque
Il y a un choix dans l'orientation du vecteur $\mathbf{n}$. L'influence sur les courbures est la suivante :

$$\begin{cases} K(-\mathbf{n}) & = K(\mathbf{n}) \\ H(-\mathbf{n}) & = -H(\mathbf{n}) \\ K_{min}(-\mathbf{n}) & = -K_{min}(\mathbf{n}) \\ K_{max}(-\mathbf{n}) & = -K_{max}(\mathbf{n}) \end{cases}$$

Remarque
La première forme fondamentale d'une surface est la forme quadratique suivante (liée à l'abscisse curviligne) :

$$I = \mathbf{dx}.\mathbf{dx} = ds^2 = Edu^2 + 2Fdudv + Gdv^2 = \begin{pmatrix} du & dv \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix}$$

(ceci permet notamment de calculer la longueur d'une courbe tracée sur la surface).

Remarque
La deuxième forme fondamentale d'une surface est la forme quadratique suivante :

$$II = -\mathbf{dx}.\mathbf{dn} = edu^2 + 2fdudv + gdv^2 = \begin{pmatrix} du & dv \end{pmatrix} \begin{pmatrix} e & f \\ f & g \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix}$$

Justification :

$$II = -\mathbf{dx}.\mathbf{dn} = -\begin{pmatrix} du & dv \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{pmatrix} \begin{pmatrix} \frac{\partial n_x}{\partial u} & \frac{\partial n_x}{\partial v} \\ \frac{\partial n_y}{\partial u} & \frac{\partial n_y}{\partial v} \\ \frac{\partial n_z}{\partial u} & \frac{\partial n_z}{\partial v} \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix}$$

soit

$$II = -\begin{pmatrix} du & dv \end{pmatrix} \begin{pmatrix} \mathbf{x_u}.\mathbf{n_u} & \mathbf{x_u}.\mathbf{n_v} \\ \mathbf{x_v}.\mathbf{n_u} & \mathbf{x_v}.\mathbf{n_v} \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix}$$

Or $\mathbf{a}.(\mathbf{b} \times \mathbf{c}) = det(a,b,c) = -det(b,a,c) = -\mathbf{b}.(\mathbf{a} \times \mathbf{c})$ d'où (après calculs) :

$$\mathbf{x_u}.\mathbf{n_u} = -\mathbf{x_{uu}}.\mathbf{n}$$

(idem pour les autres termes).

Remarque
Les directions correspondant à K_{max} et K_{min} sont perpendiculaires. Plus précisément, si les courbures des courbes dessinées sur la surface et passant par le point considéré ne sont pas toutes égales, alors il existe deux directions orthogonales qui correspondent aux minimum et maximum de courbure.

Valeurs pour quelques primitives géométriques

Le tableau suivant donne les valeurs des courbures des primitives géométriques introduites au chapitre 1. Les courbures sont ici calculées en fonction des coordonnées dans un repère propre (lié à la primitive).

Primitive	K	H	$K_{\min}, K_{\max}$
Plan	0	0	0,0
Sphère	$\frac{1}{R^2}$	$\frac{1}{R}$	$\frac{1}{R}, \frac{1}{R}$
Cylindre	0	$\frac{1}{2r}$	$0, \frac{1}{r}$
Cône	0	$\frac{\cos\alpha}{2z\tan\alpha}$	$0, \frac{\cos\alpha}{z\tan\alpha}$
Tore ($R \geq r$)	$\frac{\cos v}{r(R+r\cos v)}$	$\frac{R+2r\cos v}{2r(R+r\cos v)}$	$\frac{1}{r}, \frac{\cos v}{R+r\cos v}$

B.3 Estimation sur un nuage de points

B.3.1 Normale

La méthode classique consiste à estimer le plan d'inertie des points sur un voisinage. Ceci ramène donc à l'ajustement d'un plan de moindres carrés, qui se résout par simple calcul de valeurs-vecteurs propres sur une matrice symétrique réelle (méthode de réduction de Householder d'une matrice symétrique réelle en matrice tri-diagonale puis décomposition QL).

On peut également envisager des méthodes simples pour effectuer une estimation plus robuste du plan tangent, notamment une méthode de Monte-Carlo avec des triplets de points aléatoires.

Il est possible d'associer une incertitude sur l'estimation de la normale unitaire. Ceci peut se faire par un rapport des valeurs propres maximales et minimales de la matrice d'inertie. Une estimation sans doute plus précise consiste à considérer, comme pour le cas de surfaces ajustées par une méthode non-linéaire, la covariance en les paramètres libres, ici les angles θ, ϕ, en utilisant l'équation 4.12, page 101.

Enfin, la question du voisinage, qui est importante ici, est discutée au B.3.3.

B.3.2 Courbures

Si les normales sont toujours estimées à peu près de la même façon, il existe plusieurs variantes pour l'estimation des courbures [MV97][BC94a, BC94b].

En voici quelques-unes :

– Par une surface bi-quadratique $z = f(x,y)$ locale
  En 3D, certains auteurs estiment la courbure en ajustant simultanément et de manière itérative une surface bi-quadratique et la normale locale à la surface (correspondant à l'axe Oz pour la surface bi-quadratique) [MV97, Gou97, FF93]. Remarquons que cette méthode peut être lente, dans la mesure où il y a itération jusqu'à un certain critère de convergence (l'application de cette méthode pour dégager les centres de courbures dans [Gou97] a pris 3h30 heures pour un fichier de 30000 points, 50h pour un fichier de 400000 points sur SGI Indigo2 [Gou99, p. 149–152]).
– Par des matrices de variance-covariance
  On forme la matrice d'inertie des normales des points voisins projetées sur le plan tangent, ce qui donne une estimation de la courbure par une méthode intégrale [BC94a, LT90].
– Par un vote local
  Méthodes proches de celles par les matrices de covariance, mais avec un vote local (tensoriel) pondéré (profil Gaussien directionnel) [TM99].

– Dans [YT94], on dégage une «région locale» autour d'un point : ensemble des voisins dont la normale est assez proche angulairement de celle du point considéré. Les éléments propres de la matrice d'inertie de cette région donnent une estimation des courbures.

Les courbures ne dépendent en principe pas du repère et de la paramétrisation utilisés. On raisonne donc dans un repère lié à la normale locale **n** (estimée ci avant).

Nous présentons ici la méthode que nous avons utilisée.

Approximation par une surface locale

Supposons que l'on puisse décrire localement la surface , dans ce repère, par l'équation

$$z = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 = \mathbf{a^T q}(\mathbf{x},\mathbf{y})$$

Remarque
les coefficients a_0, a_1 et a_2 sont-ils nécessaires ici ? En effet, un autre modèle possible serait $z = a_0 x^2 + a_1 xy + a_2 y^2$, qui en théorie suffit... En pratique, il semble meilleur de considérer le modèle gardant les termes linéaires (par rapport aux erreurs d'estimation de la normale) [MV97].

Classiquement, on cherche alors la valeur de **a** qui minimise le résidu

$$R = \sum_i (z_i - \mathbf{a^T q}(x_i,y_i))^2$$

(approximation de la distance euclidienne fréquemment utilisée). Ceci conduit à minimiser

$$R = \sum_i z_i^2 - 2\mathbf{a}^T \sum_i z_i \mathbf{q}(x_i,y_i) + \mathbf{a}^T \sum_i \mathbf{q}(x_i,y_i)\mathbf{q}(x_i,y_i)^T \mathbf{a}$$

Puisqu'il n'y a pas de contrainte sur **a**, on a la condition :

$$\nabla_{\mathbf{a}} R = -2\sum_i z_i \mathbf{q}(x_i,y_i) + 2\sum_i \mathbf{q}(x_i,y_i)\mathbf{q}(x_i,y_i)^T \mathbf{a} = 0$$

donc

$$\mathbf{a}^T \left[\frac{1}{N} \sum_i \mathbf{q}(x_i,y_i)\mathbf{q}(x_i,y_i)^T \right] = \frac{1}{N} \sum_i z_i \mathbf{q}(x_i,y_i)$$

Ce qui se résout en **a** par inversion d'une matrice symétrique réelle (méthode de décomposition de Cholesky en un produit d'une matrice triangulaire inférieure et de sa transposée [PTVF92b]).

Remarque
De même que pour la normale, une estimation robuste de la fonction biquadratique ci-dessus est possible en effectuant une méthode de Monte-Carlo avec des échantillons aléatoires de 6 points, dont l'origine **0** (on est alors ramené à un problème d'inversion de matrice).

Expression des courbures en fonction des coefficients

On est dans le cas

$$\mathbf{x}(x,y) = \begin{pmatrix} x \\ y \\ z = f(x,y) \end{pmatrix}$$

On a alors

$$\mathbf{x}_x = \begin{pmatrix} 1 \\ 0 \\ f_x \end{pmatrix}, \mathbf{x}_y = \begin{pmatrix} 0 \\ 1 \\ f_y \end{pmatrix}, \mathbf{n} = \frac{1}{\sqrt{1+f_x^2+f_y^2}} \begin{pmatrix} -f_x \\ -f_y \\ 1 \end{pmatrix}$$

et

$$\mathbf{x}_{xx} = \begin{pmatrix} 0 \\ 0 \\ f_{xx} \end{pmatrix}, \mathbf{x}_{yy} = \begin{pmatrix} 0 \\ 0 \\ f_{yy} \end{pmatrix}, \mathbf{x}_{xy} = \begin{pmatrix} 0 \\ 0 \\ f_{xy} \end{pmatrix},$$

Ce qui donne

$$\begin{cases} E = 1+f_x^2 \\ F = f_x f_y \\ G = 1+f_y^2 \end{cases}$$

et

$$\begin{cases} e = \frac{f_{xx}}{\sqrt{1+f_x^2+f_y^2}} \\ f = \frac{f_{xy}}{\sqrt{1+f_x^2+f_y^2}} \\ g = \frac{f_{yy}}{\sqrt{1+f_x^2+f_y^2}} \end{cases}$$

D'où

$$K = \frac{f_{xx}f_{yy} - f_{xy}^2}{(1+f_x^2+f_y^2)^2}$$

et

$$H = \frac{f_{xx}(1+f_y^2) + f_{yy}(1+f_x^2) - 2f_x f_y f_{xy}}{2(1+f_x^2+f_y^2)^{3/2}}$$

Dans le cas de la surface bi-quadratique introduite au-dessus, ceci donne :

$$K = \frac{4a_3a_5 - a_4^2}{(1+a_1^2+a_2^2)^2}$$

et

$$H = \frac{a_3(1+a_2^2) + a_5(1+a_1^2) - a_1a_2a_4}{(1+a_1^2+a_2^2)^{3/2}}$$

Il serait intéressant d'évaluer, de même que pour la normale, l'incertitude sur les courbures estimées. On peut estimer la variance sur les valeurs des coefficients a_i, et en déduire un ordre de grandeur sur l'incertitude pour H et K. Cependant, en pratique, d'autres paramètres entrent en jeu, tels que le repère local (estimée lui aussi), et le type de surface utilisé.

B.3.3 Question sur la taille du voisinage

On a vu plus haut que l'estimation des informations locales telles que normales et courbures nécessite de définir un voisinage, c'est-à-dire un sous-nuage de points, autour du point considéré. Se pose alors la question de savoir quelle taille donner à un tel voisinage.

En effet, si cette taille est trop petite, le bruit associé à chaque point se répercute sur la précision de l'estimation.

A l'inverse, lorsque la taille est trop grande, par exemple pour le cas de la normale, le plan d'inertie devient une mauvaise approximation du plan tangent. Il en est de même pour l'estimation de la surface biquadratique utilisée pour la courbure, qui est également une approximation uniquement locale. Or, le problème est plus délicat avec la courbure, car des tailles de voisinages différentes peuvent donner des résultats complètement différents.

Définir la taille de voisinage, ou le nombre de points voisins, est un problème important et peu traité à ce jour.

Annexe C

Ajustement par méthodes d'algèbre linéaire

C.1 Modèles du type $a^T q(x,y,z) = 0$

C.1.1 Plan libre

$$F = \sum_{i=1}^{N} d(\mathbf{x_i}; \text{Plan}(\mathbf{n}, p))^2 = \sum_{i=1}^{N} [(\mathbf{x_i}.\mathbf{n}) - p]^2$$

Le paramètre p étant libre, à l'optimum, on a :

$$\frac{\partial F}{\partial p} = 0$$

ce qui donne l'expression analytique :

$$p = (\mathbf{x_m}.\mathbf{n})$$

où $\mathbf{x_m} = \sum_{i=1}^{N} \mathbf{x_i}$ est le centre de gravité des points $\mathbf{x_i}$. Le paramètre p est ainsi éliminé, et la fonction coût à minimiser devient :

$$F = \frac{1}{N} \sum_{i=1}^{N} [(\mathbf{x_i} - \mathbf{x_m}).\mathbf{n}]^2$$

soit en notation matricielle :

$$F = \frac{1}{N} \sum_{i=1}^{N} \mathbf{n^T}(\mathbf{x_i} - \mathbf{x_m})(\mathbf{x_i} - \mathbf{x_m})^\mathbf{T}\mathbf{n} = \mathbf{n^T}[\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x_i} - \mathbf{x_m})(\mathbf{x_i} - \mathbf{x_m})^\mathbf{T}]\mathbf{n}$$

En définissant la matrice symétrique réelle positive suivante (matrice d'inertie)

$$M = [\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x_i} - \mathbf{x_m})(\mathbf{x_i} - \mathbf{x_m})^\mathbf{T}]$$

le vecteur unitaire $\mathbf{n}$ cherché est le vecteur propre unitaire associé à la plus petite valeur propre de la matrice M. Or, la matrice M est définie positive : M est clairement positive ; elle est singulière seulement dans le cas où les points se situent exactement sur un plan, ce qui n'est jamais le cas en pratique à partir de 4 points. Dans ce cas, le calcul pratique peut se faire via une décomposition de Householder [PTVF92b].

C.1.2 Plan contraint à passer par un point fixe

On note $\mathbf{x_m}$ le point fixe par lequel le plan doit passer, et on utilise la méthode du plan sans contrainte ci-avant, à la seule différence que $\mathbf{x_m}$ ne désigne plus le (véritable) centre d'inertie.

En réalité, on raisonne inversement : l'ajustement de plan libre est un cas particulier de plan contraint à passer par un point fixe, avec pour point fixe le centre de gravité des points.

C.1.3 Extension à d'autres modèles

Le mécanisme qui a été présenté pour le plan est très général, et peut être étendu à d'autres types de modèles. En effet, supposons que nous voulions ajuster une fonction de la forme $a^T q(x,y,z) = 0$ où

$$q(x,y,z) = (q_1(x,y,z), \ldots, q_r(x,y,z))^T$$

est un vecteur dont les composantes sont des fonctions quelconques (non-paramétrées) de (x,y,z) et a est un vecteur de coefficients constants. On suppose simplement qu'il n'y a pas de redondance ou de dépendance linéaire entre les fonctions $q_j(x,y,z)$.

Etant donné un ensemble de points $\{\mathbf{x_i}\}, 1 \leq i \leq N$, supposons que l'on souhaite trouver le modèle de ce type réalisant le minimum de la fonction coût :

$$\sum_{i=1}^{N} (a^T q(x_i,y_i,z_i))^2$$

Ceci revient à considérer la métrique $d(\mathbf{x}) = a^T q(x,y,z)$.

> Remarque
> Pour le cas du plan, on peut utiliser le vecteur $q(x,y,z) = (1,x,y,z)^T$, et la métrique d correspond à la distance euclidienne du point $\mathbf{x} = (x,y,z)^T$ au modèle. Par contre, dans le cas général (c'est-à-dire tous les cas sauf celui du plan !), cette métrique ne correspond plus à la distance euclidienne du point $\mathbf{x}$ au modèle.

Comme le vecteur a est défini à un facteur multiplicatif près, on peut imposer la contrainte

$$\|a\|_2^2 = \sum_{j=1}^{r} a_j^2 = 1$$

Ceci revient à dire que l'on cherche un vecteur a unitaire dans $\mathbb{R}^r$.

Avec cette contrainte, la fonction coût peut alors s'écrire :

$$a^T \big[\sum_{i=1}^{N} q(x_i,y_i,z_i) q(x_i,y_i,z_i)^T \big] a$$

En introduisant la matrice symétrique réelle positive

$$M = \big[\sum_{i=1}^{N} q(x_i,y_i,z_i) q(x_i,y_i,z_i)^T \big]$$

a correspond au vecteur propre associé à la plus petite valeur propre de M. Pour réaliser le calcul, on peut utiliser la même méthode que pour le plan (décomposition de Householder). Notons que la complexité algorithmique de l'algorithme de décomposition de Householder est en $O(r^3)$.

Simplification en présence de terme constant Dans les cas où le vecteur $q(x,y,z)$ contient la fonction constante 1, on peut simplifier le problème d'une dimension. En effet, pour ces cas-là, le problème est séparable, au sens où le paramètre correspondant à la fonction 1 peut être déterminé analytiquement. Le principe est alors celui du cas du plan vu ci avant. Supposons que l'on ait

$$q(x,y,z) = (1, \bar{q}(x,y,z)^T)^T$$

avec $\bar{q}(x,y,z) = (q_2(x,y,z), \ldots, q_r(x,y,z))^T$. En introduisant

$$\bar{q_m} = \frac{1}{N} \sum_{i=1}^{N} (q_2(x_i,y_i,z_i), \ldots, q_r(x_i,y_i,z_i))^T$$

et $\bar{a} = (a_2, \ldots, a_r)^T$, on montre que, à l'optimum, on a :

$$a_1 = \bar{q_m}\, \bar{a}^T$$

La fonction à mimiser devient alors

$$\bar{a}^T \big[\sum_{i=1}^{N} (\bar{q} - \bar{q_m})(\bar{q} - \bar{q_m})^T \big] \bar{a}$$

En introduisant la matrice symétrique réelle positive

$$\bar{M} = \big[\sum_{i=1}^{N} (\bar{q} - \bar{q_m})(\bar{q} - \bar{q_m})^T \big]$$

on peut estimer $\bar{a}$ comme étant le vecteur propre unitaire (de $\mathbb{R}^{r-1}$ associé à la plus petite valeur propre de M. Outre le fait que la contrainte sur a est différente, l'un des intérêts ici est de baisser la dimension du problème de r à $r-1$. En effet, ceci est intéressant car la complexité algorithmique passe de $O(r^3)$ à $O((r-1)^3)$.

C.1.4 Surface algébrique

Ce formalisme s'adapte en théorie à toute surface algébrique, c'est-à-dire à tout modèle du type $P(x,y,z) = 0$ avec P polynôme. En effet, il suffit de décomposer P suivant une base polynomiale pour se ramener au cas précédent.

Notons cependant qu'en pratique, lorsque certains termes du vecteur $q(x,y,z)$ sont beaucoup plus grands que d'autres (ce qui est le cas avec des polynômes de degrés élevés), ceci a une influence directe sur le conditionnement de la matrice M introduite. Par conséquent, des problèmes numériques peuvent intervenir dans certains cas.

C.1.5 Cercle 2D

Le cercle 2D est un cas particulier de courbe algébrique du plan. Il suffit de choisir $q(x,y) = (1,x,y,(x^2+y^2))$. On obtient ainsi les coefficients a_1 et a_2, a_3, a_4 qui définissent l'équation

$$a1 + a_2 x + a_3 y + a_4 (x^2 + y^2) = 0$$

On en déduit directement, lorsque $a_4 \neq 0$, les paramètres du cercle : son centre

$$C = -\frac{1}{2a_4} \begin{pmatrix} a_2 \\ a_3 \end{pmatrix}$$

et son rayon

$$R = \frac{\sqrt{a_2^2 + a_3^2 - 4a_1a_4}}{2|a_4|}$$

C.1.6 Sphère

Le principe est tout à fait semblable au cas du cercle 2D. La base polynomiale s'écrit maintenant $q(x,y,z) = (1,x,y,z,(x^2+y^2+z^2))^T$. On obtient les paramètres a_1 puis a_2,a_3,a_4,a_5 qui définissent l'équation

$$a1 + a_2x + a_3y + a_4z + a_5(x^2+y^2+z^2) = 0$$

On en déduit directement, lorsque $a_5 \neq 0$, les paramètres de la sphère : son centre

$$\mathbf{c} = -\frac{1}{2a_5}\begin{pmatrix} a_2 \\ a_3 \\ a_4 \end{pmatrix}$$

et son rayon

$$R = \frac{\sqrt{a_2^2 + a_3^2 + a_4^2 - 4a_1a_5}}{2|a_5|}$$

Remarque
Notons que comme pour le cas du plan, il est simple de chercher la sphère (ou la quadrique, etc.) de moindres carrés contrainte à passer par un point.

C.2 Modèles du type $z = a^T q(x,y)$

Ces surfaces sont un cas particulier des précédentes, qui ont traditionnellement beaucoup été utilisées en vision, pour le traitement de scènes 2.5D. Ces représentations peuvent également servir dans un repère local pour estimer les courbures locales (voir annexe B).

Le formalisme suit ici celui de la régression linéaire, et est traité dans de nombreux ouvrages.

La minimisation de

$$F = \sum_{i=1}^{N} [z_i - a^T q(x_i,y_i)]^2$$

équivaut à minimiser

$$F = a^T \left[\sum_{i=1}^{N} q(x_i,y_i)q(x_i,y_i)^T\right] a - 2a^T \left[\sum_{i=1}^{N} q(x_i,y_i)\right]$$

A l'optimum

$$a = M^{-1}V$$

où

$$M = \sum_{i=1}^{N} q(x_i,y_i)q(x_i,y_i)^T$$

est une matrice symétrique réelle, positive et

$$V = \sum_{i=1}^{N} q(x_i,y_i)$$

On est donc ramené à l'inversion d'un matrice définie positive, ce qui se fait par décomposition de Cholesky en un produit d'une matrice triangulaire inférieure et de sa transposée. Voir par exemple les fonctions `cholsl` et `choldc` de [PTVF92b]. La complexité algorithmique de ce procédé est en $O(p^3)$ (où p est la dimension du vecteur q).

Annexe D

Construction d'une primitive à partir d'un quorum de points

D.1 Modèles du type $a^T q(x,y,z) = 0$

D.1.1 Cas général

On suppose que l'on a

$$a^T q(\mathbf{x}) = 0$$

avec $q(\mathbf{x}) = (q_1(x,y,z), ..., q_r(x,y,z))$ et a vecteur de $\mathbb{R}^r$. Notons que l'on définit la même surface si l'on normalise a par $\|a\|_2 = \sqrt{a^T a}$. Il suffit donc de $r-1$ points non-dégénérés pour déterminer le vecteur paramètre a. On choisit un ensemble «non-dégénéré» (au sens défini plus loin) de $r-1$ points $\mathbf{x_i}$ de $\mathbb{R}^3$ qui se trouvent sur cette surface. Ceci donne $r-1$ équations $a^T q(\mathbf{x_i}) = 0$, soit encore :

$$a_1 q_1(\mathbf{x_i}) + \ldots + a_r q_r(\mathbf{x_i}) = 0 \, (1 \leq i \leq r-1)$$

Si on suppose qu'un autre point $\mathbf{x}$ appartient à cette surface, on a également :

$$a_1 q_1(\mathbf{x}) + \ldots + a_r q_r(\mathbf{x}) = 0$$

Or, cette dépendance linéaire peut s'écrire comme un déterminant nul :

$$\begin{vmatrix} q_1(\mathbf{x_1}) & \ldots & q_r(\mathbf{x_1}) \\ \vdots & & \vdots \\ q_1(\mathbf{x_{r-1}}) & \ldots & q_r(\mathbf{x_{r-1}}) \\ q_1(\mathbf{x}) & \ldots & q_r(\mathbf{x}) \end{vmatrix} = 0$$

On obtient les coefficients $(a_i)_{1 \leq i \leq r}$ en développant le déterminant par rapport à la dernière ligne. On note à ce niveau $\Delta_{q_i(\mathbf{x})}$ le déterminant mineur associé au terme $q_i(\mathbf{x})$ dans le développement du déterminant. Ces termes sont utilisés dans la suite de cette annexe. On peut ensuite souhaiter normaliser ces coefficients en divisant par $\|a\|_2 = \sqrt{a^T a}$.

Ceci permet notamment de traiter le cas des surfaces algébriques Une surface algébrique est une surface définie par une équation implicite polynomiale, c'est-à-dire :

$$P(x,y,z) = 0$$

où P est un polynôme. En décomposant P suivant une base polynomiale $(q_1,\ldots,q_r)$, on peut écrire l'équation d'une surface algébrique S_a sous la forme :

$$a^T q(\mathbf{x}) = 0$$

ce qui ramène au cas général. Ce formalisme permet notamment de traiter l'ajustement des modèles suivants :

- Cercle 2D . $q = (1, x, y, x^2 + y^2)$
  Quorum : 3 points 2D
- Conique 2D. $q = (1, x, y, x^2, xy, y^2)$
  Quorum : 5 points 2D
- Plan. $q = (1, x, y, z)$
  Quorum : 3 points 3D
- Sphère. $q = (1, x, y, z, x^2 + y^2 + z^2)$
  Quorum : 4 points 3D
- Quadrique. $q = (1, x, y, z, x^2, xy, xz, y^2, yz, z^2)$
  Quorum : 9 points 3D

Remarque
En théorie, on pourrait traiter toute surface algébrique de cette façon. En pratique, il est à noter que la matrice obtenue peut dans certains cas être mal conditionnée (certains termes étant beaucoup plus grands que d'autres dans la matrice), ce qui peut conduire à des problèmes numériques.

Remarque
On peut appliquer cette méthode à des classes de surfaces plus générales. Il suffit que la distance s'exprime comme une fonction linéaire en les paramètres, ou encore comme une combinaison linéaire de fonctions qui ne dépendent pas des paramètres. Les fonctions peuvent faire intervenir des termes trigonométriques, exponentiels, ondelettes, etc.

D.1.2 Plan

Quorum : 3 points non-alignés.

A partir de trois points $\mathbf{x_1}$, $\mathbf{x_2}$ et $\mathbf{x_3}$ non alignés de $\mathbb{R}^3$, on définit le plan passant par les trois points par : un point sur le plan, disons $\mathbf{x_1}$; une normale unitaire $\mathbf{n}$

$$\mathbf{n} = \frac{(\mathbf{x_2} - \mathbf{x_1}) \times (\mathbf{x_3} - \mathbf{x_1})}{\|(\mathbf{x_2} - \mathbf{x_1}) \times (\mathbf{x_3} - \mathbf{x_1})\|}$$

Remarque
si besoin, on peut également ajuster un plan contraint à passer par un point donné. Ce cas est traité avec la même méthode, en remplaçant le point $\mathbf{x_1}$ par le point donné (deux autres points seulement sont alors suffisants pour définir un plan possible).

D.1.3 Cercle 2D

Quorum : 3 points 2D non-alignés

L'équation issue du développement du déterminant vu dans le cas général s'écrit ici :

$$-\Delta_1 + x\Delta_x - y\Delta_y + (x^2+y^2)\Delta_{x^2+y^2} = 0$$

Si $\Delta_{x^2+y^2}$ est non-nul, les paramètres du cercle sont

$$C = \frac{1}{2\Delta_{x^2+y^2}} \begin{pmatrix} -\Delta_x \\ \Delta_y \end{pmatrix}$$

et

$$R = \sqrt{c_x^2 + c_y^2 + \frac{\Delta_1}{\Delta_{x^2+y^2}}} = \frac{\sqrt{\Delta_x^2 + \Delta_y^2 + 4\Delta_1\Delta_{x^2+y^2}}}{2|\Delta_{x^2+y^2}|}$$

D.1.4 Sphère

Quorum : 4 points non-coplanaires
Le centre **c** est donc donné par :

$$\mathbf{c} = \frac{1}{2\Delta_{x^2+y^2+z^2}} \begin{pmatrix} \Delta_x \\ -\Delta_y \\ \Delta_z \end{pmatrix}$$

et le rayon est

$$R = \sqrt{c_x^2 + c_y^2 + c_z^2 - \frac{4\Delta_1}{\Delta_{x^2+y^2+z^2}}}$$

soit

$$R = \frac{\sqrt{\Delta_x^2 + \Delta_y^2 + \Delta_z^2 - 4\Delta_1\Delta_{x^2+y^2+z^2}}}{2|\Delta_{x^2+y^2+z^2}|}$$

D.2 Modèles du type $z = a^T q(x,y)$

Soit une surface telle que $z = a^T q(x,y)$ où $(x,y) \mapsto q(x,y)$ est une application de $\mathbb{R}^2$ dans $\mathbb{R}^r$, définie par $q = (q_1, \ldots, q_r)^T$, où $q_1, \ldots, q_r$ forment une famille libre de polynômes en x et y. On choisit alors un ensemble «non-dégénéré» (au sens défini plus loin) de r points $\mathbf{x_i}$ de $\mathbb{R}^3$ qui se trouvent sur cette surface. Ceci donne r équations $z_i = a^T q(x_i,y_i) = q(x_i,y_i)^T a$, ce qui peut s'écrire sous forme matricielle :

$$\begin{pmatrix} q_1(x_1,y_1) & \ldots & q_r(x_1,y_1) \\ \vdots & & \vdots \\ q_1(x_r,y_r) & \ldots & q_r(x_r,y_r) \end{pmatrix} a = \begin{pmatrix} z_1 \\ \vdots \\ z_r \end{pmatrix}$$

Le cas «dégénéré» mentionné plus haut correspond donc à r points qui annulent le déterminant

$$det(q(x_1,y_1), \ldots, q(x_r,y_r))$$

Dans le cas où ce déterminant n'est pas nul, on résout le système linéaire ci-dessus, et le vecteur solution a s'écrit :

$$a = \begin{pmatrix} q_1(x_1,y_1) & \ldots & q_r(x_1,y_1) \\ \vdots & & \vdots \\ q_1(x_r,y_r) & \ldots & q_r(x_r,y_r) \end{pmatrix}^{-1} \begin{pmatrix} z_1 \\ \vdots \\ z_r \end{pmatrix}$$

Quelques cas particuliers importants :

- Surface biquadratique $z = a_0x^2 + a_1y^2$
  Quorum : 2 points non dégénérés
- Surface biquadratique $z = a_0x^2 + a_1xy + a_2y^2$
  Quorum : 3 points non dégénérés
- Surface biquadratique $z = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2$
  Quorum : 6 points non dégénérés
  Ces cas sont en particulier intéressants pour l'estimation des courbures locales(annexe B).

D.3 Cercle 3D

Quorum : 3 points non-alignés de $\mathbb{R}^3$

On calcule tout d'abord le plan passant par les trois points, qui est défini par le point $\mathbf{x_1}$ et la normale unitaire

$$\mathbf{u} = \frac{(\mathbf{x_2} - \mathbf{x_1}) \times (\mathbf{x_3} - \mathbf{x_1})}{\|(\mathbf{x_2} - \mathbf{x_1}) \times (\mathbf{x_3} - \mathbf{x_1})\|}$$

qui est bien définie lorsque les 3 points ne sont pas alignés.

On définit, à partir des angles définissant le vecteur $\mathbf{u}$ en coordonnées sphériques, le trièdre orthonormé direct $(\mathbf{u}, \mathbf{v}, \mathbf{w})$.

On raisonne ensuite dans ce repère : si on note

$$x'_i = \mathbf{x_i}.\mathbf{v}$$
$$y'_i = \mathbf{x_i}.\mathbf{w}$$

alors l'équation du cercle 2D (dans le plan $(\mathbf{v}, \mathbf{w})$) s'écrit :

$$\begin{vmatrix} 1 & x'_1 & y'_1 & x'^2_1 + y'^2_1 \\ 1 & x'_2 & y'_2 & x'^2_2 + y'^2_2 \\ 1 & x'_3 & y'_3 & x'^2_3 + y'^2_3 \\ 1 & x' & y' & x'^2 + y'^2 \end{vmatrix} = 0$$

soit

$$\begin{vmatrix} 1 & \mathbf{x_1}.\mathbf{v} & \mathbf{x_1}.\mathbf{w} & \mathbf{x_1^2} - (\mathbf{x_1}.\mathbf{u})^2 \\ 1 & \mathbf{x_2}.\mathbf{v} & \mathbf{x_2}.\mathbf{w} & \mathbf{x_2^2} - (\mathbf{x_2}.\mathbf{u})^2 \\ 1 & \mathbf{x_3}.\mathbf{v} & \mathbf{x_3}.\mathbf{w} & \mathbf{x_3^2} - (\mathbf{x_3}.\mathbf{u})^2 \\ 1 & \mathbf{x}.\mathbf{v} & \mathbf{x}.\mathbf{w} & \mathbf{x^2} - (\mathbf{x}.\mathbf{u})^2 \end{vmatrix} = 0$$

Donc le centre du cercle est :

$$\mathbf{c} = (\mathbf{x_1}.\mathbf{u})\mathbf{u} + \frac{1}{2\Delta_{x'^2+y'^2}}[-\Delta_{x'}\mathbf{v} + \Delta_{y'}\mathbf{w}]$$

et le rayon est

$$R = \frac{\sqrt{\Delta^2_{x'} + \Delta^2_{y'} + 4\Delta_1\Delta_{x'^2+y'^2}}}{2|\Delta_{x'^2+y'^2}|}$$

D.4 «Cylindre-rotule»

Quorum : 2 points.

Il est relativement simple de trouver les paramètres à partir de la donnée de 2 points. Ceci conduit à la résolution d'une équation du second degré, ce qui mène à des solutions analytiques.

Résolution

Les 2 points $\mathbf{x}_i$ vérifient

$$d(\mathbf{x}_i, \text{Axe})^2 = r^2$$

soit

$$(\mathbf{x_i} - \mathbf{p}).(\mathbf{x_i} - \mathbf{p}) - ((\mathbf{x_i} - \mathbf{p}).\mathbf{u})^2 = r^2$$

ou encore

$$(\mathbf{x_i} - \mathbf{p}).\mathbf{u} = \varepsilon_i \sqrt{(\mathbf{x_i} - \mathbf{p}).(\mathbf{x_i} - \mathbf{p}) - r^2}$$

avec $\varepsilon_i = \pm 1$

Théoriquement, le terme $(\mathbf{x_i} - \mathbf{p}).(\mathbf{x_i} - \mathbf{p}) - r^2$ est positif ; en pratique on prend sa valeur absolue.

L'équation devient donc :

$$(\mathbf{x_i} - \mathbf{p}).\mathbf{u} = \varepsilon_i \sqrt{|(\mathbf{x_i} - \mathbf{p}).(\mathbf{x_i} - \mathbf{p}) - r^2|}$$

On obtient donc le système :

$$\begin{aligned} a_1x + a_2y + a_3z &= b && \text{(D.1)} \\ c_1x + c_2y + c_3z &= d && \text{(D.2)} \\ x^2 + y^2 + z^2 &= 1 && \text{(D.3)} \end{aligned}$$

avec
$\mathbf{u} = (x, y, z)^T$
$\mathbf{a} = (a_1, a_2, a_3)^T = \mathbf{x_1} - \mathbf{p}$
$\mathbf{c} = (c_1, c_2, c_3)^T = \mathbf{x_2} - \mathbf{p}$
$b = \varepsilon_1 \sqrt{|(\mathbf{x_1} - \mathbf{p}).(\mathbf{x_1} - \mathbf{p}) - r^2|}$
$d = \varepsilon_2 \sqrt{|(\mathbf{x_2} - \mathbf{p}).(\mathbf{x_2} - \mathbf{p}) - r^2|}$
avec $\varepsilon_1, \varepsilon_2 = \pm 1$

Si $a_3 \neq 0$ l'équation D.1 permet d'exprimer z en fonction de x et y dans les deux autres équations :

$$\begin{aligned} a_3^2x^2 + a_3^2y^2 + (b - a_1x - a_2y)^2 &= a_3^2 && \text{(D.4)} \\ (c_1a_3 - a_1c_3)x + (c_2a_3 - a_2c_3)y &= a_3d - c_3b && \text{(D.5)} \\ && && \text{(D.6)} \end{aligned}$$

soit

$$(a_3^2 + a_1^2)x^2 + 2a_1a_2xy + (a_3^2 + a_2^2)y^2 - 2ba_1x - 2ba_2y + b^2 - a_3^2 = 0 \qquad \text{(D.7)}$$

et si $(c_2a_3 - a_2c_3 \neq 0)$:

$$y = \frac{(a_3d - c_3b) - (c_1a_3 - a_1c_3)x}{c_2a_3 - a_2c_3} \qquad \text{(D.8)}$$

d'où

$$\begin{gathered} [(a_3^2 + a_1^2) - \tfrac{2a_1a_2(c_1a_3 - a_1c_3)}{c_2a_3 - a_2c_3} + \tfrac{(a_3^2 + a_2^2)(c_1a_3 - a_1c_3)^2}{(c_2a_3 - a_2c_3)^2}]x^2 \\ + [\tfrac{2a_1a_2(da_3 - bc_3)}{c_2a_3 - a_2c_3} - \tfrac{2(da_3 - bc_3)(c_1a_3 - a_1c_3)}{(c_2a_3 - a_2c_3)^2} - 2ba_1 + \tfrac{2ba_2(c_1a_3 - a_1c_3)}{c_2a_3 - a_2c_3}]x \\ + [\tfrac{(a_3^2 + a_2^2)(da_3 - bc_3)^2}{(c_2a_3 - a_2c_3)^2} - \tfrac{2ba_2(da_3 - bc_3)}{c_2a_3 - a_2c_3} + b^2 - a_3^2] = 0 \end{gathered}$$

soit $Ax^2 + Bx + C = 0$. On obtient donc les 2 solutions (éventuelles) par :

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

(dans le cas où $B^2 - 4AC$ est positif et A est non-nul).

Une fois la ou les valeurs de x trouvée(s), on obtient y par l'équation D.8, et z par

$$z = \frac{b - a_1 x - a_2 y}{a_3}$$

Les termes ε_1 et ε_2 introduits peuvent prendre les valeurs -1 et $+1$, ce qui fait potentiellement 4 cas. Or, deux de ce cas se déduisent des deux autres par symétrie $(\mathbf{u}, -\mathbf{u})$. Ensuite pour chacune des valeurs de $(\varepsilon_1, \varepsilon_2)$, il existe potentiellement deux solutions. Ce qui donne quatre solutions possibles.

Dans la pratique, dans un contexte d'ajustement ou d'extraction, seul l'un des quatre cas correspond (éventuellement) à la solution cherchée. Par conséquent, on évaluera la fonction coût sur les quatre cylindres en gardant le cylindre de coût minimum.

> Remarque
> Il n'est déjà plus aussi aisé de résoudre le cas d'un «cylindre-rotule» dont le rayon ne serait plus fixé, ou encore le cas du «cylindre sécant de rayon fixe», et a fortiori le cas du «cylindre sécant».

D.5 Cylindre

Quorum : ensemble non-dégénéré de 5 points (au sens défini plus loin).

On peut définir un cylindre par 5 paramètres libres. Ceci nous mène à considérer que 5 points «non-dégénérés» suffisent pour déterminer complètement un cylindre quelconque.

Notons tout d'abord qu'il n'existe pas d'expression algébrique pour décrire la classe des cylindres (l'expression algébrique la plus simple étant celle des quadriques). On pourrait bien entendu chercher la quadrique à partir de 9 points et ensuite voir si celle-ci est un cylindre. Le problème est que le cylindre correspond à un cas singulier de quadrique, qui sera donc très peu probable en pratique (particulièrement en présence de bruit sur les points).

Aussi l'idée était-elle ici de s'intéresser au cas du cylindre spécifiquement. Bien que d'apparence simple, le problème de construire le(s) cylindre(s) défini(s) par 5 points de $\mathbb{R}^3$ est un problème complexe et n'a semble-t-il été traité que très récemment [Tre00, OBPT01].

En particulier, ce problème n'a dans le cas général pas de solution analytique, et on se ramène donc à une résolution numérique.

Notre résolution du problème a consisté à partir des 5 équations provenant de la nullité des distances de chacun des points au cylindre, et de simplifier ce système au maximum (sous forme symbolique). Le détail des calculs effectués est présenté dans [Cha02]. Nous avons ainsi obtenu un système polynomial de 3 équations à 3 inconnues, qui est présenté dans la suite. Le système que nous obtenons ici est de degré inférieur à celui obtenu dans [OBPT01], et donc plus simple à traiter numériquement.

Système initial

Expression du système

Soit un cylindre défini par un vecteur directeur unitaire **u**, un point **P** sur l'axe et un rayon r.

Un point X appartient à ce cylindre si

$$d(X,Axe)^2 = r^2$$

soit

$$PX^2 - (\mathbf{PX}.\mathbf{u})^2 = r^2$$

5 points X_i sur ce cylindre vérifient le système :

$$\begin{cases} d(X_i,Axe) = r(1 \leq i \leq 5) \\ P : \text{point le plus proche de 0 sur l'Axe} \end{cases}$$

que l'on peut écrire

$$\begin{cases} PX_i^2 - (\mathbf{PX_i}.\mathbf{u})^2 = r^2(1 \leq i \leq 5) \\ \mathbf{OP} \perp \mathbf{u}(\text{sauf si P=0}) \\ \|\mathbf{u}\|^2 = 1 \end{cases}$$

soit

$$\begin{cases} PX_i^2 - (\mathbf{PX_i}.\mathbf{u})^2 = r^2(1 \leq i \leq 5) \\ \mathbf{OP}.\mathbf{u} = 0 \\ \|\mathbf{u}\|^2 = 1 \end{cases}$$

$$PX_1^2 - (\mathbf{PX_1}.\mathbf{u})^2 = r^2 \tag{D.9}$$
$$PX_2^2 - (\mathbf{PX_2}.\mathbf{u})^2 = r^2 \tag{D.10}$$
$$PX_3^2 - (\mathbf{PX_3}.\mathbf{u})^2 = r^2 \tag{D.11}$$
$$PX_4^2 - (\mathbf{PX_4}.\mathbf{u})^2 = r^2 \tag{D.12}$$
$$PX_5^2 - (\mathbf{PX_5}.\mathbf{u})^2 = r^2 \tag{D.13}$$
$$\mathbf{OP}.\mathbf{u} = 0 \tag{D.14}$$
$$\mathbf{u}.\mathbf{u} = 1 \tag{D.15}$$

Ceci forme un système de 7 équations polynomiales de degré total 4, 2 en p, 2 en u, 2 en r, à 7 inconnues.

Système final obtenu après calculs

Système en u obtenu Lorsque les points X_1, X_2, X_3, X_4 ne sont pas coplanaires, on a :

– Notation Vectorielle

$$\begin{cases} u^T M_{2345} u - c_{2345} = 0 \\ u_x[u^T M_{234}^{yz} u] + u_y[u^T M_{234}^{xz} u] + u_z[u^T M_{234}^{xy} u] - u^T V_{234} = 0 \\ u^T u - 1 = 0 \end{cases}$$

(où $u_x, u_y, u_z \in [-1,1]$) avec

$$
\begin{array}{rcl}
M_{2345} & = & -\Delta_{345}K(X_2,X_1)+\Delta_{245}K(X_3,X_1) \\
 & & -\Delta_{235}K(X_4,X_1)+\Delta_{234}K(X_5,X_1) \\
c_{2345} & = & -\Delta_{345}k(X_2,X_1)+\Delta_{245}k(X_3,X_1) \\
 & & -\Delta_{235}k(X_4,X_1)+\Delta_{234}k(X_5,X_1) \\
M^{yz}_{234} & = & \Delta^{yz}_{34}K(X_2,X_1)-\Delta^{yz}_{24}K(X_3,X_1)+\Delta^{yz}_{23}K(X_4,X_1) \\
M^{xz}_{234} & = & -\Delta^{xz}_{34}K(X_2,X_1)+\Delta^{xz}_{24}K(X_3,X_1)-\Delta^{xz}_{23}K(X_4,X_1) \\
M^{xy}_{234} & = & \Delta^{xy}_{34}K(X_2,X_1)-\Delta^{xy}_{24}K(X_3,X_1)+\Delta^{xy}_{23}K(X_4,X_1) \\
V_{234} & = & \begin{pmatrix} \Delta^{yz}_{34} & -\Delta^{yz}_{24} & \Delta^{yz}_{23} \\ -\Delta^{xz}_{34} & \Delta^{xz}_{24} & -\Delta^{xz}_{23} \\ \Delta^{xy}_{34} & -\Delta^{xy}_{24} & \Delta^{xy}_{23} \end{pmatrix} \begin{pmatrix} k(X_2,X_1) \\ k(X_3,X_1) \\ k(X_4,X_1) \end{pmatrix} \\
\Delta_{ijk} & = & \begin{vmatrix} x_i-x_1 & x_j-x_1 & x_k-x_1 \\ y_i-y_1 & y_j-y_1 & y_k-y_1 \\ z_i-z_1 & z_j-z_1 & z_k-z_1 \end{vmatrix} \\
\Delta^{xy}_{ij} & = & \begin{vmatrix} x_i-x_1 & x_j-x_1 \\ y_i-y_1 & y_j-y_1 \end{vmatrix} \text{ (idem } \Delta^{xz}_{ij} \text{ et } \Delta^{yz}_{ij}\text{).} \\
k(X,Y) & = & (X+Y)^T(X-Y) \\
K(X,Y) & = & (X+Y)(X-Y)^T
\end{array}
$$

– Notation scalaire polynomiale
Si on note $u=[x,y,z]^T$ pour simplifier les équations, on obtient les équations en x, y, z suivantes :
équation 1

$$
\begin{array}{l}
x^2(M_{2345}(0,0)) \\
+xy(M_{2345}(0,1)+M_{2345}(1,0)) \\
+xz(M_{2345}(0,2)+M_{2345}(2,0)) \\
+y^2(M_{2345}(1,1)) \\
+yz(M_{2345}(1,2)+M_{2345}(2,1)) \\
+z^2(M_{2345}(2,2)) \\
+1(-c_{2345}) \\
=0
\end{array}
$$

equation 2 :

$$
\begin{aligned}
&x^3(M^{yz}_{234}(0,0))\\
&+x^2y(M^{yz}_{234}(0,1)+M^{yz}_{234}(1,0)+M^{xz}_{234}(0,0))\\
&+x^2z(M^{yz}_{234}(0,2)+M^{yz}_{234}(2,0)+M^{xy}_{234}(0,0))\\
&+xy^2(M^{yz}_{234}(1,1)+M^{xz}_{234}(0,1)+M^{xz}_{234}(1,0))\\
&+xyz(M^{yz}_{234}(1,2)+M^{yz}_{234}(2,1)+M^{xz}_{234}(0,2)+M^{xz}_{234}(2,0)+M^{xy}_{234}(0,1)+M^{xy}_{234}(1,0))\\
&+xz^2(M^{yz}_{234}(2,2)+M^{xy}_{234}(0,2)+M^{xy}_{234}(2,0))\\
&+y^3(M^{xz}_{234}(1,1))\\
&+y^2z(M^{xz}_{234}(1,2)+M^{xz}_{234}(2,1)+M^{xy}_{234}(1,1))\\
&+yz^2(M^{xz}_{234}(2,2)+M^{xy}_{234}(1,2)+M^{xy}_{234}(2,1))\\
&+z^3(M^{xy}_{234}(2,2))\\
&+x(-V_{234}.X)\\
&+y(-V_{234}.Y)\\
&+z(-V_{234}.Z)\\
&=0
\end{aligned}
$$

equation3

$$x^2+y^2+z^2-1=0$$

Calcul des autres paramètres Une fois une valeur de u trouvée, on obtient les autres paramètres comme suit :

$$P=\frac{1}{2}((M-X_1\mathbf{1}^T)^T)^{-1}V=\frac{1}{2\Delta_{234}}\left[-\begin{pmatrix}u^TM^{yz}_{234}u\\u^TM^{xz}_{234}u\\u^TM^{xy}_{234}u\end{pmatrix}+V_{234}\right]$$

et

$$r=\sqrt{PX_1^2-(\mathbf{PX_1}.\mathbf{u})^2}$$

On peut obtenir ici jusqu'à 6 cylindres différents. Dans le cadre d'un ajustement ou d'une extraction, on testera les 6 cylindres sur les n points et on gardera le cylindre de moindre fonction coût.

Annexe E

Quelques mots sur l'implémentation

Le logiciel Cool a été implémenté en C++. La programmation orientée objet fournit un cadre naturel pour les primitives contraintes : la classe d'une primitive contrainte donnée dérive de la classe de la primitive libre correspondante(Figure E.1).

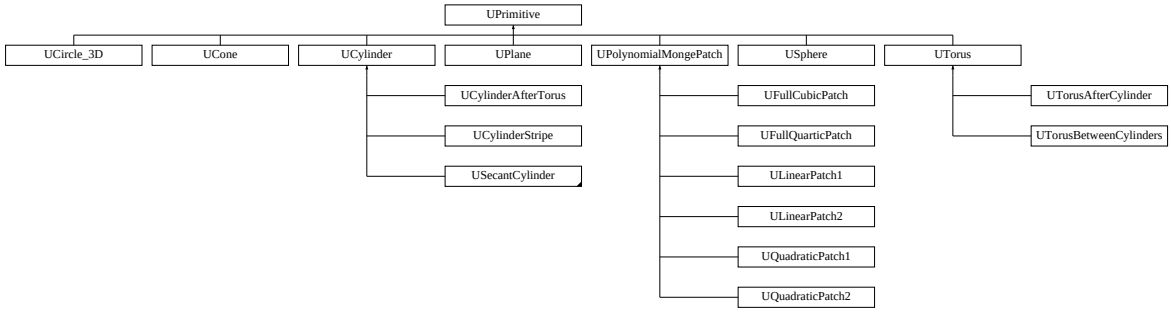


FIG. E.1 – Diagramme d'héritage de la classe UPrimitive

La classe «UPrimitive» contient comme champ un sous-nuage de points (du type ASubCloud), si bien que toute primitive possède un sous-nuage de points associé, sur lequel s'effectue l'ajustement.

La classe «ACloud» correspond à un nuage où sont stockées les coordonnées x,y,z des points. La classe «ASubCloud», qui est la plus utilisée, décrit un sous-nuage : elle contient simplement un pointeur vers un «ACloud», et une liste d'indices entiers.

Une scène est représentée par un seul nuage du type «ACloud» (contenant les coordonnées x,y,z de tous les points du nuage), un sous-nuage de points restant «mRemainingSubCloud», une liste de sous-nuages «mSubCloudList», et une liste de primitives «mPrimList».

Annexe F

Questionnaire des tests utilisateurs

Extracteur de lignes de tuyauterie
Questionnaire final (à l'issue des évaluations sur cas concrets)

1. Quelle est votre stratégie globale de segmentation d'un nuage pour isoler les lignes de tuyauterie ?
   Quelle est votre démarche générale : segmentation de tous les tuyaux de la scène, ou segmentation d'une seule ligne en ignorant les autres ?
2. Les résultats donnés par l'extracteur vous semblent-ils exploitables ? Inexploitables ?
3. En quoi l'extracteur vous paraît-il intéressant par rapport à la segmentation manuelle :
   (a) plus rapide ? moins rapide ?
   (b) plus juste ? moins juste ?
   (c) plus pénible ? moins pénible ?
   (d) Dans quel cas vous semble-t-il plus intéressant / moins intéressant que la segmentation manuelle ?
4. Quels problèmes avez-vous rencontrés sur les exemples traités ?
   (a) Quels sont les points à améliorer en priorité ?
   (b) la rapidité de calcul
   (c) la justesse des résultats (critère d'arrêt)
   (d) autres points
5. Plaçons-nous dans les cas réels (nuage de points de densité et de bruit correspondant aux productions du CNEPE).
   (a) Dans le cas d'un logiciel de segmentation idéal (celui de Thomas finalisé, complété, etc.), pensez-vous qu'il soit important que l'utilisateur intervienne au cours de l'exécution de l'algorithme, quel que soit le degré d'automatisation de la segmentation ? Si oui, à quels «moments» du traitement ?
   (b) Dans l'état actuel du logiciel, avec les limites et les failles qu'il comporte, où pensez-vous que l'utilisateur doive intervenir pour «rattraper le coup», pour guider le logiciel en somme ?
6. Lors d'un arrêt de l'algorithme, pensez-vous intéressant/inintéressant que l'utilisateur puisse intervenir, par exemple pour demander que l'algorithme poursuive la segmentation, que l'algorithme effectue un changement de diamètre, etc. ?
7. En présence de quels éléments faudrait-il que la propagation s'arrête ?
8. En présence de quels éléments faudrait-il que la propagation se poursuive ?
9. Concernant l'arrêt de la propagation dans la segmentation automatique actuelle, quel comportement est souhaitable ?
   (a) en cas d'arrêt précoce : accepter l'arrêt ou demander une poursuite de la segmentation ?
   (b) en cas d'arrêt tardif : refuser le résultat ou l'accepter ?
10. Dans des cas difficiles (nuage très peu dense, nuage très bruité), pour l'initialisation de l'algorithme, pensez-vous que l'utilisateur puisse cliquer deux points (sachant que dans ce cas la robustesse de l'algorithme sera augmentée) ?

Bibliographie

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *SIGGRAPH'98*, pages 415–421, 1998.

[AF96] J.M. Arnaudiès and H. Fraysse. *Cours de Mathématiques 4 : Algèbre bilinéaire et géométrie*. Dunod, Paris, 1996.

[AFRW97] A.P. Ashbrook, R.B. Fisher, C. Robertson, and N. Werghi. Segmentation of range data into rigid subsets using surface patches. In *BMVC'97, British Machine Vision Conference*, pages 530–539, Essex, UK, September 1997.

[Aka74] H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, AC-19(6) :716–723, December 1974.

[Ald94] M. A. Alder. Inference of syntax for point sets. In E.S. Gelsema and L. N. Kanal, editors, *Pattern recognition in practise IV*, New York : Elsevier Science, pages 45 – 58, 1994.

[Alg95] M.-E. Algorri. *Mesh generation and simplification for surface recsonstruction of unstructured points*. PhD thesis, ENST Paris, September 1995.

[AW98] E. Angelopoulou and L.B. Wolff. sign of Gaussian curvature from curve orientation in photometric space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10) :1056–1066, October 1998.

[Bah97] A. Bahi. *Segmentation de surfaces représentées par des nuages de points non organisés*. PhD thesis, Université Claude Bernard Lyon 1, 1997.

[Bar94] S. Barré. La courbure de gauss. *Journal de mathématiques des élèves de l'ENS de Lyon*, 1(1), 1994.

[BC94a] J. Berkmann and T. Caelli. Computation of surface geometry and segmentation using covariance techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 :1114–1116, 1994.

[BC94b] J. Berkmann and T. Caelli. On the relationship between surface covariance and differential geometry. In 0. Ying-Lie, A. Toet, D. Foster, H. Heijmans, and P. Meer, editors, *Shape in Picture : Mathematical Description of Shape in Grey-Level Images*, volume 126 of *NATO Asi Series. Series F, Computer and System Sciences.*, pages 343–352. Springer-Verlag, 1994.

[Bes99] P. Besl. CAD/CAM applications of 3D scanners. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, October 1999. tutorial.

[BF81] R.C. Bolles and M.A. Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI, Int. Joint. Conf. on Artificial Intelligence*, pages 637–643, Vancouver, Canada, 1981.

[BG99] M.E. Bock and C. Guerra. A geometric approach to the segmentation of range images. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, pages 261–269, October 1999.

[BJ88] P.J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2) :167–192, March 1988.

[BJ98] R. Ben-Jemaa. *Traitement de données 3D denses acquises sur des objets réels complexes*. PhD thesis, ENST Paris, 1998.

[BKV+02] P. Benkö, G. Kos, T. Varady, L. Andor, and R. Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19, 2002.

[BMG94] K.L. Boyer, M.J. Mirza, and G. Ganguly. The robust sequential estimator : A general-approach and its application to surface organization in range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10) :987–1001, October 1994.

[Bou94a] P. Boulanger. *Extraction multiéchelle d'éléments géométriques*. PhD thesis, Ecole Polytechnique, Université de Montréal, 1994.

[Bou94b] R. Boumaza. *Perception multisensorielle pour la reconnaissance d'objets tridimensionnels*. PhD thesis, CNRS-LAAS Toulouse, 1994.

[Bou99] P. Boulanger. Knowledge representation and analysis of range data. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, October 1999. tutorial.

[BS97] K. Bubna and C. V. Stewart. Model selection and surface merging in reconstruction algorithms. Technical report, Rensselaer Polytechnic Institute, Troy(NY), 1997.

[BS98] K. Bubna and C. V. Stewart. Model selection and surface merging in reconstruction algorithms. In *ICCV'98, 6th Int. Conf. on Computer Vision*, Bombay, 1998.

[BTG95] E. Bittar, N. Tsingos, and M.-P. Gascuel. Automatic reconstruction of unstructured 3D data : combining a medial axis and implicit surfaces. In *Eurographics'95*, September 1995.

[BY95] J-D. Boissonnat and M. Yvinec. *Géométrie Algorithmique*. Ediscience international, Paris, 1995.

[Car00] L. Carraro. Introduction à la régression. Cours de 2ème année, ENSMSE, 2000.

[CB00] R. Chaine and S. Bouakaz. Analyse surfacique de données 3-d non structurées : une approche basée sur les graphes. In *RFIA, Reconnaissance des Formes et Intelligence Artificielle*, volume 3, page 37, Paris, February 2000.

[CGL01] Th. Chaperon, F. Goulette, and C. Laurgeau. Extracting cylinders in full 3D data using a random sampling method and the Gaussian image. In T.Ertl, B. Girod, G.Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling and Visualization (VMV'01)*, Stuttgart, November 2001.

[CGL02] Th. Chaperon, F. Goulette, and C. Laurgeau. Extraction de cylindres dans des scènes 3D par une méthode d'échantillonnage aléatoire utilisant la sphère de gauss. In *RFIA'02, 13ème congrès francophone de Reconnaissance des Formes et Intelligence Artificielle*, Angers, January 2002. AFRIF-AFIA.

[Cha00] R. Chaine. *Segmentation d'ensembles non organisés de points 3-D d'une surface : propagation anisotrope d'étiquettes basée sur les graphes*. PhD thesis, Université Claude Bernard - Lyon 1, January 2000.

[Cha02] T. Chaperon. A note on the construction of right circular cylinders through five 3D points. Technical report, ENSMP, Centre de Robotique, 2002. A paraître.

[CHH99] O. Carmichael, D. Huber, and M. Hebert. Large data sets and confusing scenes in 3-d surface matching and recognition. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, pages 358–367, October 1999.

[CHW96] C-S. Chen, Y-P. Hung, and J-L. Wu. Model-based object recognition using range images by combining morphological feature extraction and geometric hashing. In *ICPR'96, Int. Conf. on Pattern Recognition*, pages 565–569, 1996.

[CLB95] L. Chaté, C. Lartigue, and P. Bourdet. Reconnaissance de contours et de modèles de surfaces 3D à partir de données issues de numérisation de formes. In *4 èmes Assises Européennes du Prototypage Rapide, Paris*, October 1995.

[CLJ67] I.M. Chakravarti, R.G. Laha, and J.Roy. *Handbook of methods of applied statistics*. Wiley, 1967.

[CM91] J. M. Chassery and A. Montanvert. *Géométrie discrète en analyse d'images*. Hermès, Paris, 1991.

[CM98] D. Comaniciu and P. Meer. Distribution free decomposition of multivariate data. In *SPR'98*, 1998.

[Con96] V. Conan. *Recalage de données 3D à partir d'une reconstruction de surface spline par filtrage numérique*. PhD thesis, ENSMP, Paris, December 1996.

[Dav92] A. Davignon. *Segmentation des images 3D, des discontinuités aux surfaces*. PhD thesis, Ecole Centrale de Paris, June 1992.

[Dav97] E. R. Davies. *Machine Vision : theory, algorithms, practicalities*. Academic Press, 2nd edition, 1997.

[dC76] M. P. do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall, 1976.

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

[Del01] B. Delyon. Régression. Cours de DESS, IRMAR Université Rennes-I, November 2001.

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

[Die97] J. Diebolt. Tests non paramétriques. Notes de cours DEA, IMAG, 1996-97.

[DP97] T. Darrell and A. P. Pentland. Cooperative robust estimation layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5) :474 – 487, May 1997.

[DS66] N.R. Draper and H. Smith. *Applied regression analysis*. Wiley Series in Probability and mathematical statistics, 1966.

[DS86] R.B. D'Agostino and M.A. Stephens. *Goodness-of-fit techniques*, volume 68 of *Statistics, textbooks and monographs*. Marcel Dekker, New York and Basel, 1986.

[DT96] F.W. DePiero and M.M. Trivedi. Real-time range image segmentation using adaptive kernels and Kalman filtering. In *ICPR'96, Int. Conf. on Pattern Recognition*, volume C, pages 573–577, 1996.

[ER93] H. Erkel-Rousse. Introduction à l'économétrie du modèle linéaire. Cours de l'ENSAE, 1993.

[Fau93] O. Faugeras. *Three dimensional Computer Vision : a Geometric Viewpoint*. MIT Press, 1993.

[FEF97] A. W. Fitzgibbon, D. W. Eggert, and R. B. Fisher. High-level CAD model acquisition from range images. *Computer-Aided Design*, 29(4) :321–330, 1997.

[FF93] A. W. Fitzgibbon and R. B. Fisher. Invariant fitting of arbitrary single-extremum surfaces. In J. Illingworth, editor, *British Machine Vision Conference*, pages 569–578, Sheffield, UK, 1993. University of Surrey, BMVA Press.

[FFE97] R. B. Fisher, A. W. Fitzgibbon, and D. W. Eggert. Extracting surface patches from complete range description. In *3DIM'97, International Conference on Recent Advances in 3-D digital imaging and modeling*, May 1997.

[Fit97] A. W. Fitzgibbon. *Stable segmentation of 2D curves*. PhD thesis, University of Edinburgh, 1997.

[FLW93] F.P. Ferrie, J. Lagarde, and P. Whaite. Darboux frames, snakes and super-quadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8) :771–784, 1993.

[Gal01] J. Gallice, editor. *Images de profondeur*. Traité IC2, Série Traitement du signal et de l'image. Hermès, 2001.

[GB96] M.A. Garcia and L. Basanez. Fast extraction of surface primitives from range images. In *ICPR'96, Int. Conf. on Pattern Recognition*, volume C, pages 568–572, 1996.

[GM93] S. Ghosal and R. Mehrotra. Segmentation of range images : an orthogonal moment-based integrated approach. *IEEE Transactions on Robotics and Automation*, 9(4) :385–399, August 1993.

[GM97] G. Guy and G. Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy, 3D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11), November 1997.

[Gou97] F. Goulette. *Quelques outils de géométrie différentielle pour la construction automatique de modèles CAO*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, March 1997.

[Gou99] F. Goulette. *Modélisation 3D automatique*. Sciences Mathématiques et Informatique. Presses de l'Ecole des Mines de Paris, Paris, November 1999.

[Guy95] G. Guy. Recovering surfaces, 3-d intersections, and 3-d junctions using perceptual constraints. Technical report, University of Southern California, 1995.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2) :71–77, July 1992.

[HDD+94] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH*, pages 295–302, July 1994.

[HFG00] G. Hermosillo, O. Faugeras, and J. Gomes. Cortex unfolding using level set methods. In *RFIA, Reconnaissance des Formes et Intelligence Artificielle*, volume 2, page 39, Paris, February 2000.

[HGB95] A. Hoover, D. Goldgof, and K. W. Bowyer. Extracting a valid boundary representation from a segmented range image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9), September 1995.

[HGB98a] A. Hoover, D. Goldgof, and K. W. Bowyer. Dynamic-scale model construction from range imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12) :1352 – 1357, December 1998.

[HGB98b] A. Hoover, D. B. Goldgof, and K. Bowyer. The space envelope : a representation for 3D scenes. *Computer Vision and Image Understanding (CVIU)*, 69(3) :310–329, March 1998.

[HHJJ95] M. Hebert, R. Hoffman, A. Johnson, and J.Osborn. Sensor-based interior modeling. In *6th Topical Meeting on Robotics and Remote Systems (ANS'95)*, pages 731–737. American Nuclear Society, February 1995.

[HI97] A. Hilton and J. Illingworth. Multi-resolution geometric fusion. In *3DIM'97, 2nd Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, Ottawa, 1997.

[HID95] M. Hebert, K. Ikeuchi, and H. Delingette. A spherical representation for recognition of free-form surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7) :681–690, July 1995.

[HJBJ+96] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7), July 1996.

[Hor86] B.K.P. Horn, editor. *Robot vision*. MIT Press, Mc Graw-Hill, 1986.

[HSIW96] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *ECCV'96, European Conf. on Computer Vision*, 1996.

[JB94] X.Y. Jiang and H. Bunke. Fast segmentation of range images by scan line grouping. *IAPR Conference on Machine Vision Applications*, 7(2) :115–122, 1994.

[JH97] A. E. Johnson and M. Hebert. Recognizing objects by matching oriented points. In *3DIM'97, Ottawa*, International Conference on Recent Advances in 3-D digital imaging and modeling, 1997.

[JHOH97] A. E. Johnson, R. Hoffman, J. Osborn, and M. Hebert. A system for semi-automatic modeling of complex environments. In *DIM'97, Ottawa*, International Conference on Recent Advances in 3-D digital imaging and modeling, 1997.

[Jia00] X. Jiang. A decomposition approach to geometric fitting. In *IAPR Workshop on Machine Vision Applications*, pages 467 – 470, Tokyo, 2000.

[JOM00] F. Jouzel, C. Olivier, and A. El Matouat. choice of the number of component clusters in mixture models by information criteria. In *RFIA, Reconnaissance des Formes et Intelligence Artificielle*, volume 1, page 149, Paris, February 2000.

[Kan93] K. Kanatani. *Geometric computation for machine vision*. Oxford Science, 1993.

[Kan00] K. Kanatani. Model selection in statistical inference and geometric fitting. In *Workshop on information-based induction sciences (IBIS2000)*, Izu, Japan, July 2000.

[Kan01] K. Kanatani. Model selection for geometric fitting : geometric AIC and geometric MDL. June 2001.

[Lau72] P. J. Laurent. *Approximation et optimisation*. Hermann, Paris, 1972.

[LC98] Z. Lei and D. B. Cooper. Linear programming fitting of implicit of implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2) :212–217, February 1998.

[Lei93] F. Leitner. *Segmentation dynamique d'images tridimensionnelles*. PhD thesis, TIMC-IMAG, INPG, Grenoble, September 1993.

[Lem97] C. Lemaire. *Triangulation de Delaunay et arbres multidimensionnels*. PhD thesis, ENSMSE, Saint-Etienne, December 1997.

[LGB95] A. Leonardis, A. Gupta, and R. Bajcsy. Segmentation of range images as the search for geometric parametric models. *International Journal of Computer Vision*, 14(3) :253–277, April 1995.

[LGS99] N. Loménie, L. Gallo, and G. Stamon. Structuration of unorganised clouds of points and detection of obstacles. In *Vision Interface*, Trois-Rivières, Canada, May 1999.

[LJS97] A. Leonardis, A. Jaklic, and F. Solina. Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11), November 1997.

[LMM97] G. Lukacs, R.R. Martin, and A. D. Marshall. Geometric least-squares fitting of spheres, cylinders, cones and tori. Technical Report 1068, EU Copernicus RECCAD Project, Budapest, May 1997. Martin R.R. and Varady T. eds.

[LMM98] G. Lukacs, R. Martin, and A. D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In *ECCV'98, 5th European Conf. on Computer Vision*, volume 1 of *LNCS 1406*, pages 671 – 686, Freiburg, 1998.

[LNdC95] C. Laurgeau, F. Nashashibi, and J. M. Martins da Cruz. Systèmes de modélisation géométrique 3D de l'environnement. Technical report, ENSMP, Paris, June 1995. rapport effectué pour le CEA.

[LT90] P. Liang and Todhunter. Representation and recognition of surface shapes in range images : a differential geometric approach. *Computer Vision, Graphics, and Image Processing*, 52 :78–109, 1990.

[MA97] R. A. McLaughlin and M. D. Alder. The Hough tranform versus the upwrite. TR97-02, Centre for Intelligent Information Processing Systems, Dept. Electrical & Electronic Engineering, The University of Western Australia(Nedlands), 1997.

[MA98] R. A. McLaughlin and M. D. Alder. The Hough tranform versus the UpWrite. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4), April 1998.

[McL00] R. A. McLaughlin. *Intelligent algorithms for finding curves and surfaces in real world data*. PhD thesis, University of Western australia, 2000.

[MEN01] MENSI. *3Dipsos version 2.4, manuel utilisateur*, April 2001.

[Mey02] A. Meyer. *Segmentation de nuage de points suivant les lignes caractéristiques*. PhD thesis, INSA Rouen, 2002.

[MLM01] D. Marshall, G. Lukacs, and R. Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 :304–314, 2001.

[MV97] A. M. McIvor and R. J Valkenburg. A comparison of local surface geometry estimation methods. *Machine Vision and Applications*, 10 :17–26, 1997.

[MY95] T. Masuda and N. Yokoya. A robust method for registration and segmentation of multiple range images. *Computer Vision and Image Understanding : CVIU*, 61(3) :295–307, May 1995.

[Nat98] E. Natonek. Fast range image segmentation for servicing robots. In *IEEE International Conference on Robotics and Automation*, pages 406–411, Leuven, Belgium, May 1998.

[NFJ93] T.S. Newman, P.J. Flynn, and A.K. Jain. Model-based classification of quadric surfaces. *Computer Vision, Graphics, and Image Processing (CVGIP) : Image Understanding*, 58(2) :235–249, October 1993.

[OBPT01] O.Devillers, B.Mourrain, F.P. Preparata, and P. Trebuchet. On circular cylinders by four or five points in space. Technical Report 4195, INRIA, Sophia-Antipolis, France, February 2001.

[Ost95a] G. Osty. Méthodes de reconnaissance de caractéristiques géométriques de profils 2d numérisés. mémoire bibliographique. Master's thesis, ENS Cachan LURPA, 1995.

[Ost95b] G. Osty. Reconnaissance des caractéristiques du contour 3D de pièces numérisées : la méthode par codage directionnel. Master's thesis, ENS Cachan LURPA, 1995.

[Ost02] G. Osty. *Extraction de particularités sur données issues de numérisation 3D : partitionnement de grands nuages de points*. PhD thesis, ENS de Cachan, 2002.

[PBJB98] M. W. Powell, K. W Bowyer, X. Jiang, and H. Bunke. Comparing curved-surface range image segmenters. In *ICCV'98, 6th International Conference on Computer Vision*, pages 286 – 291, Bombay, 1998.

[PDH+97] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *3DIM'97, Ottawa*, International Conference on Recent Advances in 3-D digital imaging and modeling, pages 205 –211, May 1997.

[PF96] M. Pilu and R. B. Fisher. Part segmentation from 2D edge images by the MDL criterion. In *British Machine Vision Conference, Edinburgh (BMVC'96)*, September 1996.

[Pon88] J. Ponce. *Représentation des objets tridimensionnels*. PhD thesis, Université Paris 11, Orsay, December 1988.

[Pra87] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21(4) :145 – 152, July 1987.

[PS85] F. P. Preparata and M. I. Shamos. *Computational geometry, an introduction*. Springer-Verlag, 1985.

[PTVF92a] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes, Example Book (C)*. Cambridge University Press, Second edition, 1992.

[PTVF92b] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Second edition, 1992.

[RAS97] M.R. Reed, P.K. Allen, and I. Stamos. Automatic model acquisition from range images with view planning. In *CVPR'97*, pages 72–77, 1997.

[RC88] M. Rioux and L. Cournoyer. The NRCC three-dimensional image date files. Technical Report 29077, National Research Council Canada, Ottawa, June 1988.

[RFWA99] C. Robertson, R.B. Fisher, N. Werghi, and A.P. Ashbrook. An improved algorithm to extract surfaces from complete range description. In *World Manufacturing Conference WMC'99 (ISMT'99)*, ICSC Academic Press, Durham, pages 587 – 591, September 1999.

[RL92] G. Roth and M.D. Levine. Geometric primitive extraction using a genetic algorithm. Technical Report TR-CIM-CIM-92-14, Computer vision and robotics lab., McGill University, Montreal, October 1992.

[RL93] G. Roth and M. D. Levine. Extracting geometric primitives. *CVGIP : Image Understanding*, 58(1) :1–22, 1993.

[RL94] G. Roth and M. D. Levine. geometric primitive extraction using a genetic algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9) :901 – 905, 1994.

[RW95] P.L. Rosin and G.A.W. West. Nonparametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12) :1140–1153, December 1995.

[Sav00] J-M. Savignat. *Approximation diffuse Hermite et ses applications*. PhD thesis, ENSMP, Fontainebleau, October 2000.

[SB95] N. S. Sapidis and P. J. Besl. Direct construction of polynomial surfaces from dense range images through region growing. *ACM Transactions on Graphics*, 14(2) :171–200, April 1995.

[Shi87] Y. Shirai. *Three-Dimensional Computer Vision*. Symbolic Computation. Springer-Verlag, 1987.

[SL95] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4), April 1995.

[Sny92] J. M. Snyder. Interval analysis for computer graphics. *SIGGRAPH*, pages 121–130, July 1992.

[SOR99] J-M. Savignat, O.Stab, and A. Rassineux. Reconnaissance des surfaces par analyse de la courbure. Technical report, ENSMP, 1999.

[SPPH99] R. Sacchi, J.F. Poliakoff, P.D.Thomas, and K-H. Häfele. Curvature estimation for segmentation of triangulated surfaces. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, pages 535–543, October 1999.

[ST92] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2) :185–194, July 1992.

[Tau91] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13 :1115–1138, 1991.

[TC89] Trouvay and Cauvin, editors. *Matériel Pétrole, Petroleum Material*. 1989.

[TCS+94] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. J. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3) :287–303, 1994.

[TF95] E. Trucco and R. B. Fisher. Experiments in curvature-based segmentation of range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2), February 1995.

[TM98] C-K. Tang and G. Medioni. Inference of integrated surface, curve, and junction descriptions from sparse 3D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11) :1206–1223, 1998.

[TM99] C-K. Tang and G. Medioni. Robust estimation of curvature information from noisy 3D data for shape description. In *ICCV'99, 7th International Conference on Computer Vision*, Corfu, Greece, September 1999.

[Tre00] P. Trebuchet. Cylindres de révolution passant par 4 ou 5 points. In *Journées de Géométrie Algorithmique*, Luminy, France, October 2000.

[VMJ97] T. Varady, R.R. Martin, and Cox Jordan. Reverse engineering of geometric models - an introduction. *Computer-Aided Design*, 29(4) :255–268, 1997.

[VT00] G. Vosselman and J.W.H. Tangelder. 3D reconstruction of industrial installations by constrained fitting of CAD models to images. In Ch. Perwass G. Sommer, N. Krueger, editor, *Mustererkennung 2000*, series Informatik aktuell, pages 285–292. Springer Verlag, 2000.

[WFAR99] N. Werghi, R. B. Fisher, A. Ahsbrook, and C. Robertson. Faithful recovering of quadric surfaces from 3D range data. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, pages 280–289, October 1999.

[WI95] M. D. Wheeler and K. Ikeuchi. Sensor modeling, probabilistic hypothesis generation, and robust localization for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3) :252–265, March 1995.

[WL95] K. Wu and M. D. Levine. 3D part segmentation : A new physics-based approach. In *IEEE proceedings on Computer Vision and Pattern Recognition*, pages 311–317, 1995.

[WL97] K. Wu and M. D. Levine. 3D part segmentation using simulated electrical charge distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11), November 1997.

[WSI98] M. D. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3D objects from multiple range images. In *ICCV'98, 6th International Conference on Computer Vision*, pages 917–924, Bombay, 1998.

[YBK94] X. Yu, T.D. Bui, and A. Kryźak. Robust estimation for range image segmentation and reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5) :530–538, May 1994.

[YS99] Y. Yemez and F. Schmitt. Progressive multilevel meshes from octree particles. In *3DIM'99, 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, pages 290–299, October 1999.

[YT94] T. Yoshimi and F. Tomita. Robust curvature vectors calculation from range data using ISL method. In *MVA'94 : IAPR Workshop on Machine Vision Applications*, pages 506–509, Kawasaki, Japan, 1994.

[Zha95] Z. Zhang. Parameter estimation techniques : a tutorial with application to conic fitting. Technical Report 2676, INRIA, Sophia Antipolis, France, October 1995.

[Zha96] Z. Zhang. Parameter estimation techniques : a tutorial with application to conic fitting. *International Journal of Image and Vision Computing*, 15(1) :59–76, January 1996.